Database Security XII
IFIP - The International Federation for Information Processing

IFIP was founded in 1960 under the auspices of UNESCO, following the First World Computer Congress held in Paris the previous year. An umbrella organization for societies working in information processing, IFIP's aim is two-fold: to support information processing within its member countries and to encourage technology transfer to developing nations. As its mission statement clearly states,

IFIP's mission is to be the leading, truly international, apolitical organization which encourages and assists in the development, exploitation and application of information technology for the benefit of all people.

IFIP is a non-profitmaking organization, run almost solely by 2500 volunteers. It operates through a number of technical committees, which organize events and publications. IFIP's events range from an international congress to local seminars, but the most important are:

- The IFIP World Computer Congress, held every second year;
- open conferences;
- working conferences.

The flagship event is the IFIP World Computer Congress, at which both invited and contributed papers are presented. Contributed papers are rigorously refereed and the rejection rate is high.

As with the Congress, participation in the open conferences is open to all and papers may be invited or submitted. Again, submitted papers are stringently refereed.

The working conferences are structured differently. They are usually run by a working group and attendance is small and by invitation only. Their purpose is to create an atmosphere conducive to innovation and development. Refereeing is less rigorous and papers are subjected to extensive group discussion.

Publications arising from IFIP events vary. The papers presented at the IFIP World Computer Congress and at open conferences are published as conference proceedings, while the results of the working conferences are often published as collections of selected and edited papers.

Any national society whose primary activity is in information may apply to become a full member of IFIP, although full membership is restricted to one society per country. Full members are entitled to vote at the annual General Assembly. National societies preferring a less committed involvement may apply for associate or corresponding membership. Associate members enjoy the same benefits as full members, but without voting rights. Corresponding members are not represented in IFIP bodies. Affiliated membership is open to non-national societies, and individual and honorary membership schemes are also offered.
Database Security XII

Status and Prospects

IFIP TC11 WG11.3 Twelfth International Working Conference on Database Security, July 15-17, 1998, Chalkidiki, Greece

Edited by

Sushil Jajodia
George Mason University
Contents

List of Figures ix

List of Tables xi

Preface xiii

Contributing Authors xv

Part I Invited Talks

1 E-Commerce Security: No Silver Bullet 3
   Anup K. Ghosh

2 Technical Enforcement of Informational Assurances 17
   Joachim Biskup

Part II Workflow

3 Analyzing the Safety of Workflow Authorization Models 43
   Wei-Kuang Huang and Vijayalakshmi Atluri

4 Rules and Patterns for Security in Workflow Systems 59
   Silvana Castano and Maria Grazia Fugini

Part III Privacy

5 An information-flow model for Privacy (InfoPriv) 77
Part IV  Policy Modeling

6  Security Policies in Replicated and Autonomous Databases  93
Ehud Gudes and Martin S. Olivier

7  Programmable Security for Object-Oriented Systems  109
John Hale, Mauricio Papa, and Sujeet Shenoi

Part V  Mediation

8  Secure Mediation: Requirements and Design  127
Joachim Biskup, Ulrich Flegel, and Yücel Karabulut

Part VI  Information Warfare Defense

9  Reconstructing the Database after Electronic Attacks  143
Brajendra Panda and Joseph Giordano

Part VII  Multilevel Security

10  Version Management in the STAR MLS Database System  159
Ramprasad Sripada and Thomas F. Keefe

11  SACADDOS: a support tool to manage multilevel documents  173
Jérôme Carrère, Frédéric Cuppens, and Claire Saurel

Part VIII  Role-based Access Controls

12  Using Role-Templates for Handling Recurring Role Structures  191
W. Essmayr, E. Kapsammer, R.R. Wagner, and A M. Tjoa

13  Role Based Security and Java  205
D. Smarkusky, S. Demurjian, Sr., M. Bastarrica, and T. C. Ting

Part IX  Mobile Databases

14  Security Issues in Mobile Database Access  223
Astrid Lubinski

Part X  Inference

15 Bayesian Methods Applied to the Database Inference Problem 237
LiWu Chang and Ira S. Moskowitz

16 The Design and Implementation of a Data Level Database Inference Detection System 253
Raymond W. Yip and Karl N. Levitt

Part XI  Panel

17 Security and Privacy Issues for the World Wide Web: Panel Discussion 269
Bhavani Thuraisingham, Sushil Jajodia, Pierangela Samarati, John Dobson, and Martin Olivier

Part XII  Discussion Summary

18 Workshop Summary 287
Vijayalakshmi Atluri and David L. Spooner

Index 295
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>A CTPN representation of WAM for a Workflow with one task</td>
<td>53</td>
</tr>
<tr>
<td>3.2</td>
<td>A CTPN representation of the check processing example</td>
<td>53</td>
</tr>
<tr>
<td>4.1</td>
<td>An example of Document Preparation workflow</td>
<td>61</td>
</tr>
<tr>
<td>4.2</td>
<td>The organizational model of WIDE</td>
<td>64</td>
</tr>
<tr>
<td>4.3</td>
<td>Specification of a trigger enforcing the binding of duties constraint</td>
<td>65</td>
</tr>
<tr>
<td>4.4</td>
<td>The binding0fDuties authorization pattern</td>
<td>67</td>
</tr>
<tr>
<td>4.5</td>
<td>The separation0fDuties authorization pattern</td>
<td>69</td>
</tr>
<tr>
<td>4.6</td>
<td>The number0fRoles authorization pattern</td>
<td>70</td>
</tr>
<tr>
<td>4.7</td>
<td>Example of pattern instantiation - the evaluationAgent trigger</td>
<td>71</td>
</tr>
<tr>
<td>5.1</td>
<td>Potential information flow between entities.</td>
<td>80</td>
</tr>
<tr>
<td>5.2</td>
<td>Storing of an entity's information in another entity.</td>
<td>82</td>
</tr>
<tr>
<td>5.3</td>
<td>Direct (potential) information flow.</td>
<td>82</td>
</tr>
<tr>
<td>5.4</td>
<td>Negative (potential) information flow.</td>
<td>83</td>
</tr>
<tr>
<td>5.5</td>
<td>An information sanitiser.</td>
<td>84</td>
</tr>
<tr>
<td>5.6</td>
<td>Changing of values by means of information flow.</td>
<td>87</td>
</tr>
<tr>
<td>5.7</td>
<td>Algorithm to implement the dynamic aspects of InfoPriv.</td>
<td>88</td>
</tr>
<tr>
<td>6.1</td>
<td>The Model for Global Queries and Autonomous Authorization Schemas</td>
<td>96</td>
</tr>
<tr>
<td>7.1</td>
<td>Figure 2. Access control definitions.</td>
<td>112</td>
</tr>
<tr>
<td>8.1</td>
<td>Design of mediator-based information integration</td>
<td>134</td>
</tr>
<tr>
<td>9.1</td>
<td>Damage Assessment in $H$ using a Directed Graph.</td>
<td>147</td>
</tr>
<tr>
<td>9.2</td>
<td>A More Complex Graph on Damage Assessment.</td>
<td>148</td>
</tr>
<tr>
<td>10.1</td>
<td>Version Chain</td>
<td>161</td>
</tr>
<tr>
<td>10.2</td>
<td>Architecture of Star-DBS</td>
<td>165</td>
</tr>
<tr>
<td>10.3</td>
<td>Distribution of On-page Cache Size</td>
<td>168</td>
</tr>
<tr>
<td>10.4</td>
<td>Disk I/O Reduction as a function of Access Skew</td>
<td>169</td>
</tr>
<tr>
<td>11.1</td>
<td>Initial planning (solid line) and updated planning (dashed line)</td>
<td>182</td>
</tr>
<tr>
<td>11.2</td>
<td>Initial planning (solid line) and updated planning (dashed line)</td>
<td>183</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>11.3</td>
<td>SACADDOS logical architecture</td>
<td>186</td>
</tr>
<tr>
<td>12.1</td>
<td>Elements and relationships of the basic RBAC model.</td>
<td>193</td>
</tr>
<tr>
<td>12.2</td>
<td>Example role-hierarchy.</td>
<td>194</td>
</tr>
<tr>
<td>12.3</td>
<td>Roles, Template-Roles and Role-Templates.</td>
<td>196</td>
</tr>
<tr>
<td>12.4</td>
<td>Embedding of template-roles</td>
<td>198</td>
</tr>
<tr>
<td>12.5</td>
<td>Example for embedding of template-roles</td>
<td>199</td>
</tr>
<tr>
<td>12.6</td>
<td>Relationship between two role-templates.</td>
<td>200</td>
</tr>
<tr>
<td>12.7</td>
<td>1-to-0..1 relationship between role-templates.</td>
<td>200</td>
</tr>
<tr>
<td>12.8</td>
<td>1-to-0..* relationship between role-templates.</td>
<td>201</td>
</tr>
<tr>
<td>12.9</td>
<td>1..<em>-to-0..</em> relationship between role-templates.</td>
<td>201</td>
</tr>
<tr>
<td>12.10</td>
<td>Specialization of a role-template.</td>
<td>202</td>
</tr>
<tr>
<td>14.1</td>
<td>General Mobile Scenario</td>
<td>224</td>
</tr>
<tr>
<td>14.2</td>
<td>Basic Adaptation Concept</td>
<td>231</td>
</tr>
<tr>
<td>15.1</td>
<td>A Simple Bayesian Network</td>
<td>245</td>
</tr>
<tr>
<td>16.1</td>
<td>The inference function.</td>
<td>261</td>
</tr>
<tr>
<td>16.2</td>
<td>Effect of the Number of Attributes in the Database.</td>
<td>266</td>
</tr>
<tr>
<td>16.3</td>
<td>Effect of the Number of Return Tuples of Queries.</td>
<td>266</td>
</tr>
<tr>
<td>16.4</td>
<td>Effect of the Number of Projected Attributes of Queries.</td>
<td>266</td>
</tr>
<tr>
<td>16.5</td>
<td>Effect of the Size of the Selection Criteria of Queries.</td>
<td>266</td>
</tr>
<tr>
<td>16.6</td>
<td>Effect of the Number of Tuples in the Database.</td>
<td>266</td>
</tr>
<tr>
<td>16.7</td>
<td>Effect of the Number of Queries Issued.</td>
<td>266</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>Data Dependency and Damage Assessment in $H.$</td>
<td>146</td>
</tr>
<tr>
<td>15.1</td>
<td>Simple Relational Database with an Unknown Value</td>
<td>238</td>
</tr>
<tr>
<td>15.2</td>
<td>Contingency Table</td>
<td>239</td>
</tr>
<tr>
<td>15.3</td>
<td>Simple Relational Database with Two Unknown Values</td>
<td>245</td>
</tr>
<tr>
<td>15.4</td>
<td>Missing values over multiple tuples</td>
<td>248</td>
</tr>
</tbody>
</table>
This volume is an outgrowth of the papers that were presented at the Twelfth IFIP WG 11.3 Working Conference on Database Security, held July 15-17, 1998 in Chalkidiki, Greece. Authors had an opportunity to update and revise their papers following the meeting. It is offered both to document progress and to provide researchers with a broad perspective of the recent developments in the field.

Thanks are due to the following people for making the conference a success:

The authors of the papers submitted to the conference

The reviewers of the papers

Jia-Ling Lin for her help with managing the reviews

Joachim Biskup and Anup Ghosh for invited talks

Bhavani Thuraisingham for organizing the panel

George Pangalos, the General Chair, and Ioannis Mavridis for local arrangements and preparation of preliminary proceedings

Ralph Wachter of Office of Naval Research and Carl Landwehr for the travel funds

Sujeet Shenoi for managing the travel funds

David Spooner, the chair of the IFIP WG 11.3

Thanks are also due to Jia-Ling Lin who assisted with every aspect of preparing this book, from collection of manuscripts to dealing with all matters related to \LaTeX{} style files.

Sushil Jajodia
Vijayalakshmi Atluri is an Assistant Professor of Computer Information Systems in the MS/IS Department at Rutgers University. Her mailing address is 180 University Avenue, Newark, NJ 07102, and her home page and email addresses are http://cimic.rutgers.edu/~atluri and atluri@andromeda.rutgers.edu, respectively.

M. Cecilia Bastarrica is a doctoral candidate in the Department of Computer Science & Engineering, The University of Connecticut. Her mailing address is the Department of Computer Science and Engineering, 191 Auditorium Road, U-155, The University of Connecticut, Storrs, CT 06269-3-155 and her email address is cecilia@enr.uconn.edu.

Joachim Biskup is currently a professor of information systems and security at the University of Dortmund. His mailing address is August-Schmidt-Straße 12, D-44221 Dortmund, Germany and his home page and email addresses are ls6-www.cs.uni-dortmund.de/issi/personnel/Joachim.Biskup.html and biskup@ls6.cs.uni-dortmund.de, respectively.

Jérôme Carrère is currently a graduate student at the ONERA-CERT. His mailing address is 2 Av. E. Belin 31055, Toulouse Cedex, France and his email address is carrere@cert.fr.

Silvana Castano is currently an associate professor of computer science at University of Milano, Italy. Her mailing address is Dipartimento di Scienze dell’Informazione, Universita’ di Milano, via Comelico 39, 20135 MILANO - ITALY. The email adress is castano@dsi.unimi.it.

LiWu Chang is currently a computer scientist at the US Naval Research Laboratory. His mailing address is Mail Code 5540, Naval Research Labora-
tory, Washington, DC, 20375, USA, and his home page and email addresses are http://www.itd.nrl.navy.mil/ITD/5540/personnel/lchang.html and lchang@itd.nrl.navy.mil, respectively.

Frédéric Cuppens is currently a Research Engineer at the ONERA-CERT. His mailing address is 2 Av. E. Belin 31055, Toulouse Cedex, France and his email address is cuppens@cert.fr.

Steven A. Demurjian, Sr. is an associate professor of computer science and engineering at the University of Connecticut. His mailing address is the Department of Computer Science and Engineering, 191 Auditorium Road, U-155, The University of Connecticut, Storrs, CT 06269-3-155 and his homepage and email addresses are www. engr.uconn.edu/~ steve and steve@engr. uconn.edu, respectively.

John Dobson is Professor of Information Management at the University of Newcastle, United Kingdom. He can be reached at Department of Computer Science, University of Newcastle, Newcastle NE1 7RU, U.K.; telephone (+44) 191 222 8228; email John.Dobson@ncl.ac.uk.

Lucas Dreyer is currently an analyst/programmer for Eskom in South Africa. His mailing address is 1 Lancaster Street, Flat 5, Westdene, Johannesburg, 2092, South Africa and his email address is lucas.dreyer@eskom.co.za.

Wolfgang Essmayr is currently a PhD student and research assistant at the Institute for Applied Knowledge Processing (FAW). His mailing address is Hauptstrasse 99, FAW - Softwarepark Hagenberg, A-4232 Hagenberg, Austria, and his home page and email addresses are www.faw.uni-linz.ac.at/people/we and we@faw.uni-linz.ac.at, respectively.

Ulrich Flegel is currently a research assistant at the University of Dortmund. His mailing address is August-Schmidt-Straße 12, D-44221 Dortmund, Germany and his email address is flegel@ls6.cs.uni-dortmund.de.

Maria Grazia Fugini is currently an associate professor of computer science at Politecnico di Milano, Italy. Her mailing address is Politecnico di Milano, Dip. Elettronica e Informazione, Piazza da Vinci, 32, I-20133 Milano ITALY, and her home page and email addresses are: fugini@elet.polimi.it and http://www.elet.polimi.it/people/fugini respectively.
Anup K. Ghosh is currently a senior research scientist with Reliable Software Technologies. His mailing address is Reliable Software Technologies, 21515 Ridgetop Circle, #250, Sterling, VA 20166, USA. His Web page is www.rstcorp.com/~anup and he can be reached by email at anup.ghosh@computer.org.

Joseph Giordano is a technical advisor in the Defensive Information Warfare branch of the US Air Force Research Laboratory, Rome Research Site. His mailing address is AFRL/IFGB, 525 Brooks Road, Rome, NY 13441-4505, USA and his email address is giordanoj@r1.af.mil.

Ehud Gudes is a professor of Computer Science at Ben-Gurion University in Israel with interests in Database security and Data mining. His mailing addresses and Homepage are: Dept. of Mathematics and Computer Science, Ben-Gurion University, Beer-Sheva, 84105, Israel Email: ehud@cs.bgu.ac.il, Homepage: http://www.cs.bgu.ac.il/~ehud/.

John Hale is an assistant professor of computer science at Washington State University. His mailing address is School of EECS, Washington State University, Pullman, WA 99164, USA. His home page and email addresses are http://www.eecs.wsu.edu/~hale and hale@eecs.wsu.edu, respectively.

Wei-Kuang Huang is an assistant professor in the department of Operation and Information Management at University of Connecticut. His mailing address is 368 Fairfield Road, U-41M, Storrs, CT, 06269-2041, and his email address is whuang@sba.uconn.edu.

Sushil Jajodia is Professor and Chairman of Department of Information and Software Engineering and Director of Center for Secure Information Systems at the George Mason University. His mailing address is Mail Stop 4A4, George Mason University, Fairfax, VA 22030-4444, USA, and his home page and email addresses are isse.gmu.edu/~csis and jajodia@gmu.edu, respectively.

Elisabeth Kapsammer is currently a PhD student and research assistant at the Institute for Applied Knowledge Processing (FAW). Her mailing address is Hauptstrasse 99, FAW - Softwarepark Hagenberg, A-4232 Hagenberg, Austria, and her home page and email addresses are www.faw.uni-linz.ac.at/people/ek and ek@faw.uni-linz.ac.at, respectively.

Yücel Karabulut is currently a research assistant at the University of Dortmund. His mailing address is August-Schmidt-Straße 12, D-44221 Dortmund,
Germany and his home page and email addresses are ls6-www.cs.uni-dortmund.de/~karabulu and karabulu@ls6.cs.uni-dortmund.de, respectively.

**Thomas F. Keefe** is currently an associate professor in the Department of Computer Science and Engineering at Penn. State. His mailing address is 220 Pond Laboratory, Dept. of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802. His home page and email address are www.cse.psu.edu/~keefe and keefe@cse.psu.edu respectively.

**Karl Levitt** is a professor of the Department of Computer Science at University of California, Davis. His mailing address is Department of Computer Science, University of California, Davis, One Shields Avenue, CA 95616, USA, and his home page and email address are www.cs.ucdavis.edu/people/faculty/default.html#levitt and levitt@cs.ucdavis.edu.

**Astrid Lubinski** is involved in the "Mobile Visualization"-project at the University of Rostock, with special interest in mobile database access. Her addresses are University of Rostock, Computer Science Dept., D-18051 Rostock, Germany for mail, lubinski@informatik.uni-rostock.de for e-mail and wwwdb.informatik.uni-rostock.de/~lubinski for more information at her home page, respectively.

**Ira S. Moskowitz** is currently a mathematician at the US Naval Research Laboratory. His mailing address is Mail Code 5540, Naval Research Laboratory, Washington, DC, 20375, USA, and his home page and email addresses are http://www.itd.nrl.navy.mil/ITD/5540/personnel/moskowitz.html and moskowitz@itd.nrl.navy.mil, respectively.

**Martin Olivier** is an associate professor in the Department of Computer Science at the Rand Afrikaans University in Johannesburg, South Africa. His mailing address is Department of Computer Science, Rand Afrikaans University, PO Box 524, Auckland Park, Johannesburg, 2006 South Africa. His e-mail address is molivier@rkw.rau.ac.za. More information is available on his Web page: http://www.rau.ac.za/staff/molivier.

**Brajendra Panda** is an assistant professor in Computer Science at the University of North Dakota. His mailing address is P.O. Box 9015, University of North Dakota, Grand Forks, ND 58202-9015, USA, and his home page and email addresses are www.cs.und.edu/~panda and panda@cs.und.edu, respectively.
Mauricio Papa is a Ph.D. candidate in computer science at the University of Tulsa. His mailing address is Department of Computer Science, University of Tulsa, 600 S. College Avenue, Tulsa, OK, 74104, USA. His email address is papama@murph.cs.utulsa.edu.

Pierangela Samarati is a Computer Scientist in the Computer Science Laboratory at SRI. She is currently on leave from University of Milan, where she is an Assistant Professor at the Computer Science Department. Samarati’s permanent address is: Dipartimento di Scienze dell’Informazione, Universita’ di Milano, via Comelico, 39/41, 20135 Milano, Italy. E-mail: samarati@dsi.unimi.it. Web: http://www.csl.sri.com/~samarati.

Claire Saurel is currently a Research Engineer at the ONERA-CERT. Her mailing address is 2 Av. E. Belin 31055, Toulouse Cedex, France and her email address is saurel@cert.fr.

Sujeet Shenoi is an associate professor of computer science at the University of Tulsa. His mailing address is Department of Computer Science, University of Tulsa, 600 S. College Avenue, Tulsa, OK 74104, USA. His email address is sujeet@utulsa.edu.

Debra Smarkusky is a doctoral candidate in the Department of Computer Science & Engineering, The University of Connecticut. Her mailing address is the Department of Computer Science and Engineering, 191 Auditorium Road, U-155, The University of Connecticut, Storrs, CT 06269-3-155 and her email address is debs@engr.uconn.edu.

David L. Spooner is a professor in the Computer Science Department at Rensselaer Polytechnic Institute and a member of the Faculty of Information Technology. His mailing address is Computer Science Department, Rensselaer Polytechnic Institute, Troy, New York 12180-3590. His home page and email address are, respectively, www.cs.rpi.edu/~spoonerd/ and spoonerd@cs.rpi.edu.

Ramprasad Sripada is a software designer in transaction processing group at Tandem Computers Inc. His mailing address is Tandem Computers Inc, 10100 N. Tantau Ave Loc 251-18, Cupertino, CA 95014, USA. His e-mail address is ram.sripada@tandem.com.

Bhavani Thuraisingham is a department head and a senior principal engineer at the MITRE Corporation, 202 Burlington Rd, Bedford, MA 01730. Her email address is thura@mitre.org.
T. C. Ting is a professor of computer science and engineering and the Director of Precision Manufacturing Institute at the University of Connecticut. His mailing address is the Department of Computer Science and Engineering, 191 Auditorium Road, U-155, The University of Connecticut, Storrs, CT 06269-3-155 and his email address is ting@ engr. uconn.edu.

A Min Tjoa is currently a professor of software technology and director of the Institute of Software Technology at the Technical University of Vienna. His mailing address is Resselgasse 3/188, Technical University of Vienna, A-1040 Vienna, Austria and his home page and email addresses are www ifs.tuwien.ac.at and tjoa@ifs.tuwien.ac.at, respectively.

Roland R. Wagner is currently a professor of information systems and director of the Institute for Applied Knowledge Processing (FAW) at the Johannes Kepler University of Linz. His mailing address is Altenbergerstrasse 69, University of Linz, A-4040 Linz, Austria and his home page and email addresses are www.faw.uni-linz.ac.at and rwagner@faw.uni-linz.ac.at, respectively.

Raymond Yip is a Ph.D. candidate of the Department of Computer Science at University of California, Davis. His mailing address is Department of Computer Science, University of California, Davis, One Shields Avenue, CA 95616, USA, and his home page and email address are seclab.cs.ucdavis.edu/~yip and yip@cs.ucdavis.edu.
1 E-COMMERCE SECURITY: NO SILVER BULLET
Anup K. Ghosh

1.1 INTRODUCTION

Electronic commerce has come out of its infancy, re-buffed its nay-sayers, and is now a multi-billion dollar industry. The success of early adopters in electronic commerce has now led the pragmatist herd in participating in on-line commerce. It is now a staple of advertising to include a World Wide Web address for consumers to learn more about a company or product and even to perform on-line transactions.

With the convergence of businesses to the Web and with the growth of on-line commerce, new threats to corporate assets have emerged through vulnerabilities in computer security. It is important to note that in the physical world, security and trust in commerce have evolved over centuries through a system of expectations, contracts, and a judicial system for enforcement. When depositing money with a bank, traditionally people expected that their money is adequately protected in bank vaults. In reality, most money is stored on electronic ledgers with wire transfers used for payment settlement between parties. However, the perception of security together with depositors’ insurance provides trust in the banking system for consumers.

On the Internet, new forms of currency are evolving to support electronic payment. Some payment systems involve supporting the existing infrastructure for credit and debit transactions, while other evolving payment systems provide payment by means of digital currency [3]. Whether the solution is new cryptographic protocols for assuring the integrity of payment, or simply building an Internet interface to existing payment infrastructures, trust in Internet-based
payment systems is required to build the consumer base for electronic commerce.

In physical world transactions, rarely do we ask for secure phone lines for placing orders, rarely must we provide proof-positive identification when giving out credit card numbers, rarely are non-repudiation systems employed, and rarely do we worry about our credit card slips we give to waiters in restaurants. It is reasonable therefore to ask why we must go to extraordinary lengths to secure e-commerce systems. One simple reason is perception. Security violations in Internet-based systems have received much notoriety in the popular press which, in turn, feeds the media frenzy over every new Internet security violation. As a result, a general paranoia of insecurity in e-commerce transactions has gripped the consumer public. Aside from perception, there are several technical reasons why electronic commerce must have stronger requirements on security than traditional forms of commerce [8]. First, and most importantly, is the inter-networking of computer systems. This topic is discussed shortly in this introduction. Second, the storage of sensitive data in repositories or databases makes e-commerce systems ideal targets. For instance, hacking an on-line firm’s database that holds all its customers’ credit card numbers is more profitable than dumpster diving for credit card receipts. Third, the lack of forensic evidence in computer crimes makes detection, capture, and prosecution more difficult. Good and regular auditing of computer usage is rarely practiced. Legal cases against computer crimes depend on auditing practices, audit trails, and the ability to demonstrate malice. Fourth, the ability to write programs to automate computer crimes provides a higher return on investment for computer criminals than physically committing the crime on site. Once written, hacking tools are distributed widely among “underground” networks and used by junior hackers that often do not know how the exploit scripts work, let alone how to write them. Finally, computer crimes can be committed thousands of miles from the crime scene in almost complete anonymity. The lack of a physical evidence trail at the scene of the crime makes detection and prosecution of computer crimes more difficult than ordinary white collar crime and reduces the risks for perpetrators of computer crime.

Today, the Internet is the medium of choice for electronic commerce. The Internet was not designed to be secure. Rather, the Internet was designed to support interoperability between heterogeneous computer platforms using a common protocol for communication. Providing a simple common set of protocols (TCP/IP) enabled maximum connectivity to the Internet without imposing undue burdens on each platform. In contrast, if the Internet protocol required each computer to support an encryption standard, a digital signature standard, or a key exchange standard, the Internet never would have gotten off the ground. The Internet protocol is a clear example of trading off security for flexibility. With the availability of the Internet to each desktop, the exposure of confidential or proprietary resources to the Internet is a threat to corporations. While encryption technology is very effective in providing privacy in Internet communications, it does little to close holes in the network through which
intruders gain access to sensitive assets. Firewalls are the most effective strategy for preventing unauthorized access to network services. However, even firewalls are sensitive to data-driven attacks through legitimate network services [2]. These topics are discussed in more details in Section 1.2.

Currently, a plethora of encryption protocols exists for securing data transactions over the Internet, including Secure Sockets Layer (SSL), Secure HTTP (S-HTTP), Secure MIME (S/MIME), Secure Electronic Transaction (SET), CyberCash’s Secure Internet Payment System, and DigiCash’s e-cash. The vendors that support and sell implementations of these protocols would like consumers and businesses to think that e-commerce is secure when using these protocols. They are only partially correct. The truth is there is no silver bullet to e-commerce security. Securing the data transaction via encryption protocols provides privacy for data sent over the Internet. It does not protect a company’s e-commerce server system from attack. It does not provide end users protection against malicious mobile code downloaded from rogue Web sites. Any e-commerce transaction is processed by a number of different components, any of which may be a weak link in the security of the transaction. The security of the system is only as strong as its weakest link. Computer criminals are unlikely to attempt to attack even weak encryption protocols (e.g., 48-bit encryption) when breaking into network servers is so much easier. Thus, the security of the components executing e-commerce transactions should be relatively uniform in strength. If one component is significantly stronger than others, then the weaker components are more likely to be attacked and the system compromised.

Recognizing that the security of the data transport in e-commerce systems is significantly stronger than other components in e-commerce systems, the weak links in e-commerce security are highlighted in the rest of this paper including client software such as Web browsers, server software including network services and the operating system, and CGI scripts.

1.2 WEAK LINKS

Electronic commerce systems are often implemented as a three-tiered architecture consisting of client software, network server software, and back-end databases. In addition, a middleware layer exists between network servers and the back-end databases that processes e-commerce transactions and updates the databases. Vulnerabilities in any of these software components can compromise the security of the entire enterprise.

The most serious risks posed to e-commerce security today are borne by the merchants themselves. Consumers’ liability due to credit card fraud is often limited. For instance, if a consumer’s credit card numbers are stolen and fraudulently used, the consumer’s liability is restricted to U.S. $50 by most credit lenders. On the other hand, the amount of credit card fraud that occurs on an annual basis totals in the billions of dollars and is accepted as the cost of business. In spite of the staggering losses due to credit card fraud, the banking industry is not complacent about fraud. Rather, the strategy used to address
credit card fraud is risk management. Fraud risks are identified and costs for mitigating them calculated. When the return on investment is sufficient, legal enforcement is applied and preventative mechanisms are deployed. Risk management is equally effective in dealing with threats to computer security. No commercial system will ever be 100 percent secure to all possible threats. Rather, the benefit of preventing computer crimes from occurring must be weighed against the cost of protecting digital assets from attack.

The liability for merchants due to computer crimes is rising significantly each year. A joint study by the Computer Security Institute (CSI) and the U.S. Federal Bureau of Investigations (FBI) in 1998 found that the total financial losses reported by corporations over the previous year rose 36% from a similar study in 1997 to $136 million [7]. Over 64% of respondents in the study reported computer security breaches within the past 12 months, representing a 16% increase in computer security violations over the previous year’s findings. Finally, the most serious losses occurred through unauthorized access by insiders. This last point underscores the need for maintaining internal host security at the merchant site in spite of unknown threats from the Internet.

Before discussing the weak links in the merchant side, the risks posed to end users are first described.

1.3 WEB CLIENTS

The vast majority of all vertical market (business to consumer) transactions are performed using Web browsers as the front-end. Web browsers today pose risks to end users’ security and privacy. The greatest threat to end users’ security and privacy is simply lack of knowledge about the risks of using Web browsers to visit untrusted sites. In addition, a large amount of privacy concerns can be alleviated by simply knowing what personal information is captured by Web sites when surfing the Web. Additional steps can be taken from preventing personal information from being released as described in this section. Similarly, hazards imposed by executable content can be addressed by disabling their execution from untrusted Web sites by configuring the browser appropriately.

Casting executable content aside for the moment, flaws in Web browsers themselves can cause security problems for end users. Employees who use browsers to “surf the Net” may potentially compromise the security of the corporate systems. The first issue companies must wrestle with is whether or not to trust the Web browser itself. Most browsers are given the privilege to execute programs locally, to write to user disks, to upload and download files and programs from the Internet. The consumer must trust that the browser software is not performing any malicious actions such as corporate espionage on a file system.

As an example of client browser vulnerabilities, consider how users of Microsoft’s Internet Explorer (IE) version 3.0 can be tricked into executing any program on their machines at the behest of a remote server [5]. The discovery of this bug in the Internet Explorer has resulted in careful scrutiny of the Internet Explorer software, resulting in yet more security bug findings by groups
at MIT and the University of Maryland [4]. The bug allows a Web page master to embed “shortcuts” from a Web page to a program anywhere on a user’s host machine. A shortcut is a method for executing a program on a Win32 machine from the desktop. The security problem is that a computer user might hit a link on someone else’s Web page (residing on a remote server) that causes the execution of a program that resides on the user’s local machine. What this means is that someone else whom you don’t necessarily know or trust can cause certain programs to execute on your desktop. What is particularly insidious about this bug is that it requires no executable content to be downloaded, such as ActiveX controls or Java applets, and the program can be executed in spite of the highest level of security set in the IE browser. The latter point is not surprising since the security levels can only prevent executable content from executing on the user’s local machine, rather than preventing a shortcut to another program on the desktop.

The bug is a direct result of integrating the IE browser with the Windows desktop. While the ability to create a shortcut from a Web page to a user’s desktop may have been viewed as a “feature” during the software design, it is in fact a bug that can be used against a user to violate the security of the user’s machine. The bug allows a Web page writer to include “.URL” and “.LNK” files on a Web page. These files are shortcuts to executing programs that may exist on the user’s Win32 machine. For this attack to work, the programs must exist on the desktop. There are, however, many programs that exist on a Windows desktop that can be used maliciously. For example, a shortcut link can be made to the Windows COMMAND.COM executable program to execute DOS commands that can modify the file system. With some degree of sophistication, the problem can be made much worse. By combining shortcut links with client-side executable scripts such as VBScript or JavaScript, malicious commands or programs can be downloaded into the IE cache as a batch file and then executed in sequence. This problem has since been corrected in subsequent release versions of the Internet Explorer.

1.3.1 Privacy concerns

Privacy issues in e-commerce are becoming highly visible in the media as well as in the U.S. Congress. Simply by surfing to different Web sites, consumers are giving out personal information about themselves. For instance, Web sites can and often do collect information about their Web site visitors including their name, email address, their Internet domain, machine name, platform type, and browser type. Some issuers of digital certificates, which are used to authenticate users to Web sites, shamelessly ask for and incorporate personal information such as age, gender, and profession in the digital certificates read by Web sites. This information, in turn, is used for directed marketing as well as profiling of visitors.

Cookies are another Web technology that can be used to intrude on end users’ privacy. Cookies are data that are sent back and forth between the Web server and client to maintain state between Web connections. Cookies
are stored on the user's own disk and retrieved the next time the user visits the same site. As such, Web sites can maintain persistent information about individual's browsing and shopping habits without knowledge or approval of the individual. The best known example of this strategy is by the company Doubleclick (ad.doubleclick.net). Doubleclick uses cookies to profile an individual's browsing habits. These browsing habits, in turn, are used for directed marketing to users who hit Web sites that use Doubleclick's services. Any time a user hits a site using Doubleclick, cookies are set to update the user's profile. For instance, if the user hits a search engine site and types the keywords "computer security", the user's individual profile will be updated to reflect that user's interest and on the page that is subsequently returned, banners of computer security vendors may be displayed. While directed marketing can be argued as benefiting both consumer and vendor, the databases built up on individual shopping/browsing habits can also be construed as an invasion of privacy. The safest way to prevent organizations to collect personal information is to browse through a proxy that removes the personal information. The anonymizer site (www.anonymizer.com) provides this service free of charge. Web proxies can be configured in house to do the same.

1.3.2 Executable content

Executable content poses privacy and security risks to end users, too. Java applets, ActiveX controls, Javascripts, and VBscripts are all examples of executable content. Others include PostScript files, multi-media files for browser plug-ins (e.g., .avi and .wav files), and mail attachments such as MS Word files. All of these forms of executable content are often downloaded or shared in e-commerce activities. Recently, the introduction of push technology into Web browsers has opened new vulnerabilities in the desktop by introducing scheduled content delivery including executable content. Executable content is simply another term for a computer program. Whenever computer programs are downloaded and executed on end users' machines, they execute with the privilege of the end user. Therefore, it is possible to program an ActiveX control that is placed on a Web page to download to a user's machine and read their mail files and send this information back to the Web server. The notorious Computer Chaos Club out of Germany demonstrated the ability to download a seemingly benign ActiveX control that in fact scheduled electronic transfers of funds from the user's account (set up by Quicken personal financial software) to a numbered Swiss bank account [1]. Similarly, it is possible to write JavaScripts to spy on all Web pages a user visits and send this information back to the Web site that sent the JavaScript.

1.3.2.1 Java applets. Java applets are mobile Java programs. That is, Java applets can be automatically downloaded from any Web page and run within the user's Web browser. Because the browser runs with the privilege of the user, the potential exists for Java applets to gain access to sensitive files on the user's desktop or to even execute commands with the user's full privileges.
Java applets should not be confused with Java applications, which like any other full-featured program, have unrestricted access to system resources. Because Java applets automatically download and execute on the user’s machine when its hosting Web site is hit, Java applets are considered untrusted code that must be carefully constrained. For this reason, the inventors of Java created a "sandbox" for Java applets in which Java applets may safely execute without posing risks to the user’s security or privacy.

The Java sandbox poses a technological solution to constraining potentially malicious applet behavior. For instance, Java applets are not permitted to access the local file system. Also, Java applets are not allowed to make network connections except back to the originating site, nor can they listen to network connections made to the user’s machine. The Java sandbox is enforced by three technologies: the bytecode verifier, the applet class loader, and the security manager [6]. The three technologies work in concert to prevent an applet from abusing its restricted privileges. Because each provides a different function, a flaw in any one can break the whole sandbox. For this reason, not only must their design be solid, but their implementations must be correct. The complexity of the functions that each technology provides makes correct implementations a difficult goal to attain in practice.

The Java security problems found to date have been a direct result of flaws in the implementations of the three components of the Java sandbox. Despite the efforts of JavaSoft in creating a sandbox, the Java security model has been broken on more than one occasion [6]. The Java security model depends on the enforcement of type safety in the language. Dynamic class loading in Java applets makes static type checking infeasible. Hence, the necessity for the three-pronged approach to the sandbox. Attack applets that are able to break type safety are effectively able to break out of the sandbox and completely compromise the system. Type safety flaws in the Java Virtual Machine (JVM) have largely been found by researchers in laboratories and since corrected by the vendor. Not surprisingly, flaws in the Java security model are usually found with each new release of the Java Developers Kit (JDK). With the release of JDK 1.2, a new security model for Java applets based on code signing is supported. This model effectively opens the sandbox to allow cryptographically signed applets to access system resources. If an applet has the correct signature to access the file system, for instance, it may be allowed to read or write files. Unsigned applets will still be restricted by the sandbox model. The problems the code signing model introduces are that every site must create, implement, and administer its own security policy for applets. Requiring sites to develop and administer their own security policies has proven to be impractical to date.

1.3.2.2 ActiveX controls. In contrast to the Java security model, ActiveX controls rely on a trust-based model for preventing malicious controls from executing. An ActiveX control is simply a program wrapped in a pre-specified interface that the Internet Explorer browser can execute. The program executes with the full rights and privileges of the browser. As such any ActiveX
control can access any files on the user’s machine, can delete, steal, or modify these files, and can execute commands on the user’s machine. There are no constraints on the behavior of ActiveX controls. The ActiveX Exploder site (www.halcyon.com/mclain/ActiveX) illustrates this property well. The ActiveX control automatically downloaded, installed, and executed from this site will shut down a Windows machine.

The only technology imposed on ActiveX controls to prevent potentially malicious behavior is the control that requires user approval before installing the control. Prior to downloading and installing a new ActiveX control, a dialog box is popped up in the user interface. If the ActiveX control has a signed certificate, the certificate can be displayed to show which organization or individual is endorsing the control. If the user trusts the endorser, then the control will be downloaded and execution will begin. However, there is no technology to prevent a malicious control at this point to violate the security or privacy of the end user. As a result, the security model is totally trust-based. Users must make their own decisions on whether the control is trustworthy or not. Caution must be executed before agreeing to install and execute an ActiveX control.

**1.3.2.3 Push technology.** The final type of executable content to be considered here is known as push technology. Push technology turns the Web paradigm on its head. Web surfers are used to finding a Web site and requesting information. The information is pulled into the user’s browser. With push technology, users still have to determine which Web sites they want information from, but once selected, the Web sites take matters into their own hands and push information to the browser without the user’s prodding. Web sites who push active content are similar to their counterparts in the TV and radio industry. Essentially, these sites broadcast their content. Users need only “tune” their browsers to their channel. Hence, the concept of “active channels”, now being pushed by Microsoft in the Internet Explorer 4.0. The idea is to get the latest updates on information without having to request it, since presumably you will not know when to request updated information.

The first well-known adopters of push technology came in the form of PointCast and Marimba. PointCast Network is a program that exploits push technology to distribute news over the Internet. PointCast broadcasts news, stock updates, sports scores, weather, and other dynamic content on a seemingly continuous basis.

Unlike the prevalent pull paradigm of the Web, push technology works on the principle of passive acceptance of data. That is, the client always accepts data pushed from the content provider, without control over what data is being sent. In the pull model, a client actively requests data from a Web site. Push technology, on the other hand, requires this decision to be made once. That is, the user subscribes to a channel (Web site) once and from that point on any and all content that matches your personal filter is downloaded. Bear in mind, the customizations are not geared around filtering out viruses. This
gives the subscribed sites a great deal of leverage to send any data of their choosing. For example, a Web site can send not only updates of news, but also active content, digital images, plug-ins, and even software patches to update the network client on-the-fly. Since the client often belongs to one of the subscribers (e.g., PointCast and Microsoft), the client can be programmed to serve any number of functions. The client can be an interpreter to execute commands sent from broadcasters. For the more paranoid of mind, the network client can be used to spy on user's networked drives and send this data back over a network socket. How difficult would such an attack be? Remember the network client will have full system privileges as any other program running on your desktop. Also remember client approval is granted *a priori* via the subscription for downloading content over an active channel. Consider that the client (e.g., the IE 4.0) can download, install and execute executable content such as ActiveX controls at any point in the future. This means that it is possible to write an ActiveX control, or even a trusted Java applet to download to targeted clients (subscribers) and perform nefarious functions such as spying on their hard drives or even deleting files. Is this a stretch of the imagination? Perhaps. But the mechanism will be technologically built in to your desktop machine.

Other security concerns over push technology center around the updates of software. Network clients that support push technology can immediately update themselves with each new patch or each new release version of the software. This technique by itself can go a long way towards making networked machines more secure. Every time a software flaw is found in the network client, the network client can reach back to the vendor, download the patch, install it, and fortify itself against known attacks. One downside of the technique is the fact that the network client is downloading executables that can alter its functionality. The question is how safe are these executables? Is it possible that they could be downloaded from a rogue organization posing as the vendor? The answer is yes. Domain name spoofing is a well-known Internet attack. The attack works by fooling a DNS server to resolve a network address to an incorrect IP address belonging to the perpetrator. The perpetrator could then download its own version of the software modified to perform its objectives, such as spying on your hard drive. Can this attack be prevented? Yes. Using digital signatures, all executables can be signed to provide proof positive of the identity of the software publisher and to determine if the software has been corrupted in transit. This system is not perfect, however. The system is based on trust. You must trust each of your content providers to not download any malicious content. Even with digital signatures, a "trusted" organization can still exploit the push technology for its own gain at the expense of selected targets. Since downloading of content occurs at scheduled intervals, rather than at the behest of the end user, this malicious content can be downloaded and executed while the user is asleep at night or on a coffee break. This leaves the end user unaware of what happened and the content can erase all traces of any nefarious activity since it is given full access to the system.
Are these reasons enough to not use push technology? Not unless you are using it on an enterprise-critical or mission-critical machine where the compromise of your digital assets could result in severe consequences. It is important to note that at the time of this writing, no attacks through push technology are known. The most important step users can take is to educate themselves of the risks and manage them appropriately.

1.4 NETWORK SERVERS

Clearly, a host of security and privacy issues are raised by the Web browsers that everyone now uses. Education is the best antidote to the risks of executable content to users. As discussed earlier, secure data transaction protocols can provide strong privacy for data transported in on-line sessions. The weak links in e-commerce security are on both ends of the network connection. Gene Spafford, a computer security researcher at Purdue University, made an interesting comment on the disparity between data transaction security and the security of the client and network server software:

Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit-card information from someone living in a cardboard box to someone living on a park bench.

In the analogy of on-line commerce, users live in an environment as secure as a park bench, while the network servers are as secure as a cardboard box in the physical world. Clearly, if someone really wanted to steal a credit card number, it would be foolish to attack the armored car rather than either the cardboard box or the park bench.

Network server security is one of the most important components to secure of all e-commerce system components. The reason is that network servers are the gateway from the untrusted Internet to a company’s proprietary digital assets. As such, they must be guarded against the types of threats posed by malicious computer hackers.

The most widely used technology for protecting network servers and internal digital assets is the firewall. Firewalls are the first line of defense against external attacks. A firewall is placed between the computer network to be protected and the network that is considered to be a security threat. Though firewalls are typically used to isolate local area networks within a company from the Internet, firewalls are also used to partition, isolate, and control access between internal corporate networks. Firewalls are usually a combination of filtering routers and application proxies that run on a dedicated machine.

Firewalls provide control over which network services are offered to the Internet or the external network at-large. An easy way to secure a network from external threats is simply to disconnect all access to and from the Internet. Since some Internet services such as mail, Web access, and FTP are essential in today’s corporations to do business, disconnecting from the Internet is not a feasible nor a strategic option. On the other hand, simply connecting all internal machines to the Internet without forethought to computer security can place corporate assets at risk. Firewalls are a compromise solution between
these two extreme positions of security and insecurity. Even though rigorous access controls can be imposed on the types of network services offered, there is the potential for attacks to be waged against a corporate network through errors in configuration of the firewall, around the firewall through backdoors, or even through legitimate requests over network services offered through the firewall.

While firewalls are useful for thwarting attacks launched through unintended network services, there is little that firewalls can do to prevent data-driven attacks through legitimate requests made to offered services. These types of attacks use the legitimate grammar of the protocol and creative license to trick software on the inside of the firewall to act on behalf of a remote user in violation of the security policy of the site.

Firewalls are ineffective at thwarting data-driven types of attacks through legitimate network service requests. One class of attacks exploits weaknesses in network applications running on a server. For example, sendmail is one of the most commonly used mail servers used on Unix machines. Throughout its long history (sendmail is now on version 8 approaching version 9) sendmail has been rife with errors that have resulted in security vulnerabilities. For example, in the past when sendmail was compiled in "debug" mode, it allowed untrusted outside users unrestricted access to the system. Even now, security-related bugs in sendmail are usually discovered with each subsequent release version. The problem is not that sendmail is poorly written, rather, the size and complexity of the sendmail program make a bug-free implementation a near impossibility.

Firewalls can do little to prevent program errors in an application server from being exploited through legitimate requests to the server. They can, however, limit the extent of the damage. A firewall proxy can create an artificially small file system around an executing application server. By creating this "jail cell" around a server, if the server program is compromised by an outside request, then the extent of damage that can be caused by the intrusion is limited to the scope of the jail cell. In the case of sendmail, a data-driven attack that is able to obtain shell access on the server through a bug in sendmail will only be able to access files and/or programs in the file system that is defined by the jail cell. Of course, any mail that is within the scope of the jail cell may be vulnerable to eavesdropping by a subverted sendmail program. The key to addressing the firewall's vulnerability to data-driven attacks is to stay on top of the latest holes found in server-side software and to patch the software as fixes are released.

Network servers are vulnerable to external threats due to errors in configuration, flaws in the server software and interface scripts, inappropriate access controls to the back-end databases, and security holes in the operating system.

---

1The Unix chroot command is used to define the "jail cell".
2Bugtraq (www.netspace.org/lsv-archive/bugtraq.html) is the premier forum for reporting security-related software bugs.
that underlies the network server. The setup and configuration of a network server can be complex and, similar to firewalls, simple errors in configuring the network server may have drastic security implications. Most network servers consist of network services such as a Web server, a mail server, and sometimes other network services such as file transfer protocol (FTP), and news (NNTP). Configuring these services securely is a formidable task even for experienced administrators. Most of the problems in security of corporate systems are a direct result of errors in configuration. The rest are flaws in the actual software source code. Configuration errors can lead to privilege escalation where an unauthorized and untrusted user gains a level of privilege for unauthorized access to corporate information systems. As an example, consider a system administrator who installs and configures a Web server. The system administrator knows that the server must start up as the super user — the user account with highest privileges — in order to listen to port 80, the standard port for Web requests. Without realizing the security implications, the system administrator also sets the executing privilege of the Web server to the super user. Now, any actions the server takes will have the force of the super user. This means that if an attacker is able to subvert the server or any of the programs the server calls, the attacker will now have the privilege to read, modify, delete, or create any file on the system.

Software is mostly configured to meet the functional requirements of the organization, e.g., providing access to corporate intranets from remote logins, rather than configured to meet requirements of corporate security policy. Most network software that is installed out-of-the-box is configured by default to provide maximum functionality, rather than security. Unless configured to meet a company's own security policy, the network services will probably be vulnerable to attack. The default configurations of network servers are known as the **deadly defaults**. Therefore, the firewall, the network servers, the middleware, and access to the back-end databases must all be configured to uphold each site's own security policy. In [3], common errors in configuration of Web servers that are exploited for security breaches are described.

### 1.5 SERVER-SIDE MIDDLEWARE

Aside from the network server, perhaps a more dangerous form of software that has emerged in on-line applications is the Common Gateway Interface (CGI) script. CGI scripts and other middleware are server-side programs that execute when called by the Web server in response to a Web request. Simple CGI scripts may increment a counter each time a Web page is accessed. Others may support customer feedback via mail. More sophisticated CGI scripts perform online transaction processing tasks required of on-line commercial transactions. For example, a CGI script may submit a customer query to an on-line database to find out the customer's investment portfolio balance.

Because CGI scripts execute in response to a remote user's request and typically process user input directly, the danger exists for a user to be able to manipulate the CGI script into giving system privileges to the untrusted
user. This is particularly true for electronic commerce, where in order for any transaction to occur, user input is necessary, an application must be executed, and files must be updated. It is the sheer power of CGI to execute interesting applications that makes it so dangerous to corporate security.

CGI scripts are written in interpreted languages such as Perl and Python or in compiled languages such as C. Perl is popular for CGI scripts because of the ability to rapidly construct applications and the ability to parse text from user input easily. However, languages such as Perl also provide a great deal of power for executing system commands that can be exploited by malicious users. For example, certain Perl commands such as \texttt{eval()}, \texttt{system()}, backquotes ('), pipes, and \texttt{exec()} can potentially result in system commands being executed on the network server host at the discretion of a unknown and untrusted remote user. Using these commands in CGI scripts is especially dangerous because unexpected malicious user input from remote systems can easily turn commonly used Perl commands into vehicles for intrusions.

Several steps can be taken to mitigate the dangers of CGI scripts. First, users should not be allowed to place their own CGI scripts on the Web server. Users are much less likely to test and verify that their scripts do not pose a security hazard, especially if they do not have the technology to perform security analysis. System administrators must be aware of stray CGI scripts that get placed on the server. These scripts can often be a backdoor that hackers (or potentially malicious internal users) leave behind to allow unauthorized entry into a system. The Web server should be configured such that CGI programs can only be executed from a single directory (with appropriate access control). If configured successfully, this measure can reduce the threat of users creating CGI scripts in their home directories. Even CGI scripts that are distributed with Web servers, downloaded from the Internet, or purchased commercially, should be viewed with suspicion. More to the point, all CGI scripts should be tested rigorously for security holes.

Scripting or interpreted languages such as Perl should be avoided. While compiled languages such as C can be equally hazardous, the scripting languages make it easier for users to unintentionally code dangerous constructs. Even if the system administrator decides that a CGI script is safe, it is wise to keep the source code for the CGI scripts hidden from the outside world. If a person outside the organization can download the source, then the source can be analyzed for vulnerabilities and potentially exploited later. Finally, every CGI program on the server must be accounted for in terms of its purpose, origin, and modifications. If the program does not serve a business function of the Web server it should be removed. This will eliminate most of the demo CGI scripts that are distributed with the Web server software. Once a stable set of CGI programs is established, a digital hash of the program (using MD5, for example) executables should be made. This will allow any modifications of the programs to be detected in the future by comparing subsequent hashes with the original digital hash.
1.6 CONCLUSIONS

Electronic commerce systems are vulnerable to malicious attack through many different software components. This paper describes vulnerabilities in Web clients, firewalls, network servers, server-side middleware, and databases that can be exploited in compromising e-commerce security. In current e-commerce systems, the lion’s share of security work has been focused on data transaction protocols. While these protocol implementations are not free from errors by any means, they represent the most secure component in e-commerce systems today. As a result, a disproportionate amount of attacks against e-commerce systems have been focused against server-side systems and users' client software.

Acknowledgments

This work has been supported by the Advanced Technology Program of NIST under Cooperative Agreement # 70NANB7H3049.

References


Abstract: Dealing with informational assurances we have to consider the full complexity of the information society. In a narrower sense informational assurances comprise informational rights, the related legal and social rules as well as the enforcing technical mechanisms. The right of privacy, understood as informational self-determination, is taken as an important example. Starting from a discussion of present shortcomings in technically enforcing this right, we outline some recent developments in the German and European legislation concerning privacy, teleservices and digital signatures. Also some selected mechanisms for improving the technical enforcement are evaluated, including federated system structure and local security autonomy, cryptographic protocols enabling cooperation under threats, and the tamper resistant hardware foundation. Finally, we advocate the shift from the traditional paradigm of reference books implemented as centralized databases to the new paradigm of communicating personal data agents. The new paradigm is devised to enhance the data subject’s means to technically enforce the interests concerning privacy.

2.1 INTRODUCTION

Over the last decades the vision of what is called the “information society” has evolved. Some features of this vision have already become reality, others are still nebulous and open for the future. Both sides, the technical innovations in the past and the further developments in the future, challenge our communities: the past technical achievements strongly require new and adapted social foundations, and the ongoing technical projects demand a careful social design. Democratic societies already started to become aware of these challenges and partially responded to them. Two mainstreams of activities can be identified,
privacy protection and evaluation criteria for critical computing systems. Each stream has its own weaknesses, and even both streams together cannot cope with all issues.

2.1.1 Privacy protection

The first sort of activities concerns individual privacy. Here fundamental human rights of individuals are sought to be protected against the assumed overwhelming informational power of public institutions and private companies. The basic legislation decrees so-called “informational self-determination”, i.e., in principle, each individual citizen can freely decide on whom he gives what part of his data and on what kind of processing his data he is willing to agree. According to this principle, an individual should retain full control over processing and disseminating his data. However, this principle is questioned by conflicting social goals, technical difficulties and the lack of effective and efficient technical enforcement mechanisms.

Examples of conflicting social goals are public security, law enforcement, national defence, social and health services, scientific research, freedom of press, participation in public decision, or trade interests. Basically, legislators dealt with such conflicts in two ways: the basic privacy law simply declares that some agencies or institutions are exempted from the principle, or the basic law refers to additional, so-called sector-specific laws each of which regulates the conflicts for some restricted domain. Critics, however, argue that there are too many global exemptions and that sector-specific laws do not cover all relevant domains and lack coherence.

Technical difficulties mainly group around the following four observations. 1. Once an individual has disclosed some of his data (understood as knowledge about him), deliberately or under legal compulsion, this data (understood as some digits) is processed within a computing system that is under the control of someone else. While, ideally, a subject is entitled to control his data (knowledge), this data (digits) is not physically available to him but just only to those agents against whom, among others, his privacy should be protected. 2. The correlation between data as knowledge and its encoding as digits is inherently difficult to monitor. In some cases it is even deliberately blurred, for instance by a cryptographic encipherment. 3. Digital data can be easily duplicated and may be spurious. 4. Much data (as knowledge) is not merely personal but deals with social relationships with other individuals within the real world, for instance data about matrimonial or childhood status or about medical treatments. Accordingly, also within the computing system this data (as digits) is not unambiguously connected to a personal file but may be spread across the files of all the persons involved, or the data even disguises as pointers or related technical concepts.

Legislators appear to have dealt with the first three technical difficulties only rather weakly. Basically, the first observation is treated by penalties and some supervision, the second one by a somehow sophisticated though not technically elaborated definition of “personal data” (as any information relating
to an identified or identifiable natural person), and the third one by a technical appendix to the basic privacy law which states some high-level, declarative rules of well-controlled data processing.

The fourth observation on the relationships seems to be completely ignored, and in fact it may also be seen as already resulting from another kind of conflicting social interests. Whereas the conflicts mentioned above are between a weak individual and a powerful institution, the conflicts inherent in social relationships may also arise between individuals of equal strength. Moreover, even without any conflicting interests, the problem of how to represent real world relationships within the formalism of a computing system has been intensively studied in the field of data modelling but not generally been solved.

The lack of technical enforcement mechanisms for the principle of privacy is mainly due to the problems already discussed before: without a socially agreed settlement of conflicts we cannot construct fair technical enforcement mechanism; the postulated ideal control and the actual physical control are separated; the semantics of digitally stored data with respect to the outside world are rarely captured algorithmically; and the physical possibilities of manipulating and duplicating digital data cannot be fully controlled using only traditional data processing techniques but would strongly require to employ new technologies like cryptography.

2.1.2 Evaluation criteria and computing security agencies

The second sort of activities responding to the challenges to the information society is directed to assist organizations in running their computing systems in a secure way. Here the organizational needs and interests are sought to be protected against accidental or malicious misbehaviour of people, or of system components devised by them. In contrast to the activities on privacy protection there is no general legislation, but the states founded specialized new computing security agencies, which, at least at the beginning, happened to be closely related to previous or existing security agencies. The computing security agencies are supposed to publish evaluation criteria for secure computing systems (see for instance [55, 17, 16, 18]), to evaluate products against these criteria (or to supervise other institutions carrying out such evaluations), and to give advice to organizations of public interest concerning security in computing systems.

In the early stages military needs and their interest in strict confidentiality (against the assumed enemy) dominated. Accordingly, early evaluation criteria were strongly influenced by the Bell-LaPadula model of restricting and controlling the data flow within a computing system. But already then, not only confidentiality but also availability of data and other computing resources as well as their integrity have been recognized as important security goals. All goals, however, were mainly treated from the perspective of a centralized, strictly hierarchical organization running centralized computers for a specific unquestioned purpose.

As time went by, both the scope of the evaluation criteria broadened and computing technologies changed dramatically. In the public and commercial
sector integrity and availability were preferred to confidentiality, and computer networks and workstations suffer from vulnerabilities and offer options which differ from those of traditional mainframes. The computing security agencies have reacted with and still work on a series of amended evaluation criteria, the latest are the so-called Common Criteria.

The Common Criteria [18] can be characterized by two important features. Firstly, they take into account the international flavour of the “information society”, i.e. its participants live in a globalized informational environment and thus the security of their computing systems have to be evaluated according to a broad international perspective. And secondly, within that globalized environment there are many different needs and interests around, i.e. the various participants should be supported to define their specific protection profiles and security targets.

Although many shortcomings of the restricted purely military oriented point of view on centralized computing systems have been eliminated, the Common Criteria still do not adequately face the large variety of today's computing systems, and they do not appropriately cover the evolving reality of everyone's computing environment with publicly available international telecommunication, Internet, personal computers and digital service providers.

2.1.3 Unsolved issues

Roughly speaking, the weaknesses of both streams of activities can be summarized as follows.

- The privacy-oriented stream deals with the legal issue of fundamental rights of individuals, but it does not seriously consider to balance all other rights involved, and it does not thoroughly take care of the technical enforcement of its requirements.

- And the evaluation-criteria-oriented stream emphasizes technical enforcement of security but it largely ignores social and legal issues within a democratic society.

Even worse, though the technical guidelines of evaluation criteria can also be helpful to operating computing systems which manage personal data, they are not at all tailored according to the technical enforcement of privacy.

In order to respond to the challenges of the evolving information society, we need a much broader approach: A political discussion aiming at a balanced social solution of and a coherent legislation on all aspects of digitally processing information and of digitally delivering services. And a scientific development aiming at the actual technical enforcement of the social and legal rules we will have agreed upon.

In particular, the political discussion has to consider all parties involved and their possibly mutual conflicting rights and interests, and it has to be consistent with the actual and future state of technologies. And science has to elaborate on all sensible social options in order to provide suggestions for their effective and efficient technical implementation.
The rest of this paper is devoted to direct the reader's attention to some recent contributions towards these goals. It is not intended to present a complete survey (that would be beyond the scope of the author's present resources) rather it concentrates on selected topics that the author believes to proceed in the right direction (and that the author is acquainted with from his actual experience).

2.2 INFORMATIONAL ASSURANCES

2.2.1 An outline of the information society

The "information society" comprises all individuals, participating in or being affected by electronic information processing, as well as their public institutions of any level and their private companies of any size. These individuals, institutions and companies are tied together by a historically achieved and further developing framework of informational and other rights and interests which, in some instances, might be shared or, in other circumstances, might be in conflict.

Seen from the perspective of this discussion, the information society is technologically based on public or private telecommunication services, on which computerized networks for all kinds of computers are run, for example ranging from personal computers over office workstations with local or specialized global servers to powerful mainframe computers. Such networks are used for a wide variety of purposes, in particular to exchange raw data, like email, to provide informational services of any kind, like daily news, video entertainment, event and transportation schedules or database records, and to support informational cooperation like home banking, electronic commerce or certifying digital documents.

Additionally, in response to the challenges mentioned above, the information society should be based on a coherent and balanced system of informational rights and socially agreed and legally founded rules as well as of mechanisms that support the participants in enforcing their issues. Such a system has been suggested to be called (in German) "informationelle Garantien" [31], and in the next subsection of this paper a further outline and elaboration will be presented under the keyword "informational assurances".

2.2.2 A framework of informational assurances

Dealing with informational assurances we have to consider the full complexity of the information society. In particular we have to uniformly cope with

- all its participants comprising both the individuals and the groups that are formed by them ranging from public institutions over civil associations to private companies,

- their informational rights,
their specific needs and wishes for information, informational services and informational cooperation,

their specific interests for such informational activities,

the mutual conflicts among such rights or interests,

the anticipated threats to such rights and interests,

the necessary basis of trust that is required for fulfilling such needs and wishes,

the social and legal rules for that trust,

and the technical mechanisms that can enforce such social and legal rules as a matter of routine in daily live.

Informational assurances in a narrower sense comprise the informational rights, the social and legal rules as well as the enforcing technical mechanisms.

By the very nature of the information society, nearly every individual, institution, association or company has to be treated as a participant. A participant may play an active role, or he might be only passively affected by the actions of other participants. In general, every participant will be involved in many ways.

Informational rights always arise with a double meaning. On the one hand, a participant is entitled to behave how he is named here: he has all civil rights to participate in the activities of the information society and to take advantage of them. On the other hand, if being an individual, a participant enjoys the fundamental human rights, including privacy in the sense of informational self-determination, and also otherwise he is the object of all kinds of protection that a state offers: in any case informational activities should not be harmful to him. Therefore many informational activities should be both enabled and restricted by law and its enforcement.

Based on general informational rights on participation, a participant can actively pursue his specific informational needs and wishes. His demands may be concerned with a wide range of activities, which can be roughly classified as follows: information as such (meaning that he is providing or collecting and processing any kind of data that seems relevant to his participation), informational services (meaning, for example, that he is asking for or delivering press services, electronic entertainment, database retrieval, etc.), or informational cooperation (meaning that he is involved, for example, in some role of electronic commerce, electronic voting, document certification, etc.).

Once a participant is involved in some informational activity, actively or passively, he is following several interests, which may vary considerably depending on the specific situation. I advocate that the goals commonly cited for defining computer security, namely availability, integrity, authenticity possibly with non-repudiation, confidentiality and others, should be understood first of all as specific interests of participants within an informational activity.
Both the general rights, based upon which participants are involved in some informational activity, and the specific interests of the participants involved may turn out to be conflicting. Indeed, they will be in conflict most of the time. The conflicts arise from the different active roles and passive affectednesses in an informational activity.

Each conflict may result in threats to rights or interests. In fact, in case of conflicting issues, one participant following his issue appears as threatening the conflicting issues of another participant. Additionally, we are also faced with threats resulting from the accidental or malicious misbehaviour of some participant. Such a troublemaker may be intentionally involved in the informational activity, or he may come more or less from outside, for instance misusing some computing facilities that are available for him because of his general rights of participation.

Although there are in general unavoidable conflicts and threats, informational activities, seen as purposely arisen interaction of participants, must be somehow based on trust. Ideally, a participant would prefer to trust only those other participants that he can exercise some kind of control over. Practically, however, the case of having direct control over others rarely occurs. Basically, there are two ways of solving this dilemma.

In the first way the assistance of further participants is required. They are intended to act as some kind of notary or arbitrator, which are to be trusted by the original, possibly mutual distrusting participants. In the second way the trust is shifted to some technical equipment, more precisely to the people delivering that equipment.

For any kind of trust, we need some social and legal rules. They are required either to establish trust, as, for example, in a notary or in the Technical Control Board, or to deter misbehaviour, or, if this fails, to deal with the consequences of misbehaviour. Such rules have to be enforced somehow. For hopefully rare cases, this task is the role of law courts.

For the routine cases of daily life in the information society, however, it appears desirable to shift most of the enforcement burden directly to technical mechanisms. By the design and tamper resistant construction of such technical enforcement mechanisms, it should be just technically infeasible to violate the rules, or, otherwise, the mechanisms should effectively provide sufficient documented evidence against a violator.

2.2.3 The interrelationships of political and technical aspects

It is worthwhile to note how the political aspects, dealing on one side with informational rights and on the other side with the social and legal rules for trust, are intimately intertwined with technical aspects, concerning on the one side informational activities and on the other side technical mechanisms to enforce rules. As a full discussion of the interrelationships of these aspects is beyond the scope of this paper, we only state some short observations.

In most cases informational rights are based on traditional fundamental human and civil rights. These traditional rights are reinterpreted and concretized
with respect to the new technical possibilities of informational activities. Some of these new possibilities, however, may not be appropriately captured by the traditional rights at all. In that cases, the fundamental human and civil rights have to be augmented by additional, newly stated informational rights. For example, the right of informational self-determination has been directly derived from fundamental human rights of self-determination (in Germany stated as Article 2(1) in connection with Article 1(1) of the Constitution). But certifying public verification keys for digital signatures and using digital pseudonyms need to be treated by some newly created right (that has to fit traditional rights, of course).

Informational activities are not merely technical, but also or first of all, depending on the point of view, they constitute social interactions. As such they require some trust among the participants and, additionally or substitutionally, in their social environment. This trust in turn has to be founded in social and legal rules.

Social and legal rules for trust should have a technical basis for ensuring that they are routinely manageable for the massively occurring and more or less only technically observable informational activities. Thus they require technical enforcement mechanisms.

Surely, such technical enforcement mechanisms may affect informational rights, and it may happen that they impact the originally wanted informational activities and the required social and legal rules.

Summarizing, we can see a feedback loop where political aspects are followed by technical aspects and vice versa. Each of these aspects occurs on two levels: on a high and more or less declarative level (informational rights, and informational activities, respectively), and on a lower and more or less implementational level (social and legal rules, and technical enforcement mechanisms, respectively). Of course, a closer inspection would show more detailed levels and additional feedbacks.

2.2.4 The health care example

The field of health care provides good examples of both the interactions of the various aspects and many subtle details. Here we can only shortly sketch some points. A comprehensive study [51] has been performed, for instance, by the SEISMED consortium within the Advanced Informatics in Medicine program of the European Union. Some more personal views on this topic are contained in [7, 8].

Everybody is supposed to share the fundamental right to take advantage of medical services. This right is complemented by the health care professionals' fundamental legal obligation to provide their services at their best. And there are important additional social and legal rules involved, for instance for professional secrecy, control on epidemics, freedom of medical research, cost effectiveness, or for health insurances. With the emergence of computer and telecommunication technologies an important part of health care procedures can now be considered as informational activities where nearly everybody is
involved in various active roles and passive affectednesses, not seldom even simultaneously. Thus, the fundamental rights of health care as well as the additional rules have to be adapted to the new situation in order to ensure that appropriate information technologies are selected and dependably operated in an agreed mode.

Since, whether following a conscious decision or just as a matter of fact, the information technologies involved tend to be open federated systems of more or less autonomously participating components, the technical basis for the adapted rights and rules should be incorporated in the components themselves, as far as possible at all. Thus we are faced with the challenge to provide technical enforcement mechanisms that are located in the federal components and are under the physical control of their owners, i.e. of the human interest holders. Apparently, cryptography that is based on personal tamper resistant hardware devices and trustworthy certification procedures appears to be indispensable. But also the collection and maintenance of personal data should be reconsidered in order to substitute today's centralized data repositories by networks of communicating personal data agents whenever the social or legal rules ask for personal control exercised by the affected data subjects.

In fact, I strongly argue that we can comply with strong versions of many rules only by combining cryptography with personal data agents. Certainly, if this vision was realized, in turn we would need new rules for the anticipated personal computing, in particular for ensuring the availability of data that is socially or contractually required. This need would arise because the first technical problem with respect to privacy protection, as stated in Section 2.1.1, would be converted into its symmetric counterpart. Whereas now a data subject is concerned about some other participant having actual control on his data, in the vision that other participant would worry to get the subject's data actually transmitted when required at some point of time.

2.2.5 The situation in Germany

In Germany, legislation on privacy started with the "Bundesdatenschutzgesetz" (Data Protection Act) [14] which was declared in 1977 and essentially amended in 1990. The amendment was based on a sentence [15] of the Bundesverfassungsgericht (German Constitutional Court), which postulated the informational self-determination as part of the fundamental human rights. Accordingly, the law specifies that the processing of personal data is admissible only if at least one of the following conditions is met: the affected person has willingly agreed on the processing or that law or some other legal regulation allows it. There are some sector-specific legal regulations, in particular the so-called "Sozialgesetzbuch", which covers processing of personal data within the system of social security and health care. The underlying idea is to balance the fundamental right of informational self-determination with the practical needs of efficient and cost-effective daily life procedures.

While legislation on privacy has exhibited the tendency to be protective, i.e. to restrict the data processing, which has already evolved anyway in the past,
recent legislation on teleservices and digital signatures [25] is more directed to enable future good practice.

The Teledienstegesetz (Teleservices Act), as Article 1 of the Informations- und Kommunikationsdienste-Gesetz (Information and Communication Services Act) [25], aims at establishing “uniform economic conditions for the various applications of information and communication services”, like for example telebanking, Internet access or electronic commerce. This law states that teleservices can be freely offered, subject that the service complies with general legal rules, and it limits the service providers’ responsibility for the information content on their own part, thereby mostly excluding responsibility for mediated parts.

The Teledienstegesetz is complemented by a sector-specific data protection law, the Teledienstedatenschutzgesetz (Teleservices Data Protection Act), as Article 2 of the Informations- und Kommunikationsdienste-Gesetz (Information and Communication Services Act) [25]. Among other features, it obliges service providers to offer clients using the services anonymously or under pseudonyms. Thus, besides the traditional aspect of privacy concerning the confidentiality of personal data and its actual protection, the law takes care of a second aspect of privacy, namely of non-observability of personal behaviour. However, presumably the relevant obligations, as stated in the law, will be rather weak in practice, because anonymity and pseudonyms are required only under the proviso that these features are “technically feasible” and that they can be reasonably expected.

The Signaturgesetz (Digital Signature Act), as Article 3 of the Informations- und Kommunikationsdienste-Gesetz (Information and Communication Services Act) [25], mostly deals with a legal and organizational framework for establishing trust in using digital signatures. In particular, it defines rules for licensing certification authorities and for their procedures to provide evidence for a relationship between some natural or juristic person and a public verification key for digital signatures. It obliges the certification authority to reliably identify that person that may demand to get certificates under a pseudonym. Interestingly, the law also contains an article on “technical components”. Its intention is, basically, that the purpose of digital signatures is actually met by the computing systems run by the certification authority. Therefore it requires that the technical components are sufficiently tested according to the state of technology and are approved by some institute acting on behalf of the licensing authority.

The Signaturgesetz is complemented by a so-called Signaturverordnung (Digital Signature Ordinance), SigV, [24] which among others details the requirements of the law pertaining the technical components. These requirements state all the nice features that you expect for digital signatures related to key generation, storage of a secret signature key and controlling access to it, adequate determination of data to be signed, secure register of certificates, and correct time-stamps. The ordinance also tells how to get assured about such properties, namely on the one side by a catalogue of suitable security measures
to be published in the Federal Gazette, and one the other side by an evaluation according to the evaluation criteria, as discussed in Section 2.1.2.

It must be emphasized that the law is expected to be widespread applied in both the public and commercial sector, and thus to substantially enhance future informational cooperation, but literally it is only some kind of proposition to the participants of the “information society”, and it does not exclude any other means to make their cooperation trustworthy. Thus only future practice will finally show how the participants will behave and how courts will decide on disputes.

It should be clear from the preceding paragraphs that the sketched approach to dealing with legally binding electronic statements are an important step towards complying with a framework of informational assurances, as presented in Section 2.2.2. In particular, the subtle interrelationships between political and technical aspect are dealt with by connecting the legal rules directly to technical enforcement mechanisms. Thus in this field we can see a promising attempt to address the unsolved issues, identified in Section 2.1.3 with respect to privacy and technical enforcement.

The German computing security agency is called “Bundesamt für Sicherheit in der Informationstechnik” (Federal Agency for Security in Information Technology), BSI (cf. http://www.bsi.de). It was founded in 1990 as an authority in the portfolio of the Ministry of Interior. The surveillance tasks defined in the Digital Signature Act (and other tasks) are part of the duties of the “Regulierungsbehörde für Telekommunikation und Post” (Regulatory Authority for the Telecommunications and Posts), RegTP, (cf. http://www.regtp.de) which was founded in 1998 in the portfolio of the Ministry of Economics as some kind of successor of the former Ministry of Posts.

2.2.6 The situation in Europe

The member states of the European Union (EU) have rather different traditions in dealing with data protection and related legal issues, see for instance [53, 36, 54], comprising, say, the German perspective of the individual’s right of informational self-determination as well as the Swedish point of view that citizens must be able to control their local administration and thus should be allowed to inspect the files of the administration.

Originally founded as a community emphasizing a common and free market, the EU is recently evolving towards a political union as demonstrated by the agreement signed in Maastricht in 1992. Accordingly the EU now has to deal with the fundamental rights and needs of its citizens, too, and thus also with privacy, informational self-determination and related concepts. Furthermore, these rights and needs of individuals have to be balanced with conflicting goals, in particular with the original trade interests within Europe and with sovereign rights of the member states concerning national and public security.

As one of the results, the EU finally accepted a Directive on Data Protection [27] in 1995. Although announced to support a high level of protection, and being an important first step for Europe indeed, nevertheless the directive ap-
pears as a (too weak) compromise between the diverging pressures of national governments to maintain national law traditions and to exempt important areas of data processing from restrictions. See for instance [35, 54] for critical remarks.

After years of stagnation, as seen from the outside—or apparently more likely of confrontations, as seen from the inside—the European Commission recently came up with several documents [28, 29, 30] on informational services and cooperation, in particular with a communication on a “European Framework for Digital Signatures and Encryption”, a communication on “The Need for Strengthened International Coordination”, and a proposal for a “Directive on Digital Signatures”. These documents emphasize the need of strong cryptography for supporting the citizens’ requirements on acting within the information society, in particular with respect to authentication, integrity and confidentiality. Though we cannot expect yet that all national governments will finally fully agree to the Commission’s points of view, there is presumably a high interest in declaring European mandatory directives as soon as possible. This expectation applies at least to the less controversial field of digital signature which are widely accepted to be crucial for electronic commerce. The field of encryption, though also identified as crucial for informational services and cooperation in general, may turn out not to be mature for a final European conclusion in the near future, unfortunately.

2.2.7 A notion of security

Within the framework of informational assurances, as sketched in Section 2.2.2, any formal notion of security for the technical enforcement mechanisms should be embedded in an overall reasoning about all relevant aspects and comply with the diversity of interests of the participants involved. The commonly used keywords for security —availability, integrity, authenticity possibly with non-repudiation, confidentiality and others— merely express such interests in a high level declarative way, and, accordingly, they have to be substantially refined for all of the participants’ views on a specific informational activity. In the next paragraphs, the author’s own approach [6, 7] towards defining an appropriate notion of security is shortly outlined; a more thorough discussion of this topic can be found, for instance, in [43]; a recent study to relate a broad perspective of security, so-called multilateral security, to evaluation criteria and certification can be found in [48, 49].

Basically, the approach follows the framework of informational assurances. The proposed formal notion of security results from capturing the process of designing a system that can be claimed to be secure. At the beginning of this process the participants of an informational activity are supposed to form a community. Each participant, or appropriate groups of them, expresses his specific needs and wishes for the computing system to be designed. Already on this level of abstraction, some conflicts among the participants’ demands and with respect to informational (or other) rights may arise. After appropriately resolving these conflicts, all further steps are based on the fundamental
assumption that the intended purpose of the system is legitimate and consistent. Accordingly, on this level, we can tentatively define: A system is secure iff it satisfies the intended purposes without violating relevant informational (or other) rights.

Then, in further refinement steps, all the concepts have to be detailed and formalized, the already introduced concepts as well as further ones like the participants’ interests and their anticipated threats or the trust in subsystems participants are willing to grant. We emphasize that all concepts are thought to be decentralized. Finally, at the end of the process, the definition of security roughly says that the final system meets the intended purposes, even if it is embedded in adversary environments, and it “does not do anything else” that has been considered to be harmful and has been explicitly forbidden therefor.

2.3 SELECTED MECHANISMS FOR TECHNICAL ENFORCEMENT

In Section 2.2.2 we considered informational assurances as comprising informational rights, social and legal rules as well as enforcing technical mechanisms. The technical enforcement mechanisms play a crucial rule in the routine cases of daily life, for the tremendous amount of technical informational events occurring within the information society can only be effectively controlled by means that are technical too. The technical enforcement mechanisms should make it technically infeasible to violate the social and legal rules, or otherwise, the mechanisms should effectively provide enough documented evidence against a violator. Then, the role of a human participant would be reduced to autonomously select technical enforcement mechanisms according to his rights and interests and the conflicts and threats anticipated by him, to control the selected mechanisms, and to use documented evidence of violations in (hopefully) exceptional and rare cases.

Evidently, these requirements appear to be difficult to meet, in particular because informational activities usually concern many participants with different expectations. There is some hope, however, to solve at least some aspects of this challenge by

- emphasizing federated (rather than centralized) system structures with a high degree of local security autonomy,

- employing cryptographic protocols (where cryptography is used as the discipline for enabling cooperation under threats), and

- using specialized hardware as tamper resistant foundation.

Of course, the indicated parts of such solutions refer to different layers of a computing system, and accordingly they would have to be carefully harmonized. Unfortunately, the author does not know about any comprehensive approach to such a solution. However, there are already a lot of proposals and subsystems for partial aspects available. In the rest of this section, some examples of them are shortly sketched, and their potential impact for a comprehensive solution is roughly indicated. The selection of the examples is strongly biased by the
author’s personal experience, and the readers are cordially invited to add their own insight and contributions.

2.3.1 Federated system structure and local security autonomy

2.3.1.1 The paradigm of communicating personal data agents. Currently we can identify two extreme kinds of storing personal data. The first one uses traditional, centralized, and (more or less) well-structured databases which gather and hold all the data that the database owners suppose to need as data consumers for their organizational purposes, actually right now or potentially in the future. The second one is the rapidly evolving, totally decentralized, and (more or less) unstructured World Wide Web, WWW, where individuals or institutions offer their data as data providers for anyone who might take advantage from it.

The privacy legislation, as discussed in Section 2.1.1, deals with the first kind only. It attempts to protect individuals, acting as data providers, against the database owners that have physical control over the stored data. Among others, the protection is based on postulating a restriction: any supply of personal data for a database is bound to a specific, well-described purpose, and afterwards the database owners are not allowed to use that data beyond the stated purpose. As mentioned before, in this scenario the data subjects are in a somehow weak position, since they do not dispose of technical mechanisms to enforce the postulated restriction.

In order to remedy this situation, many years ago we designed and prototyped the so-called “personal model of data” [3, 4, 5, 13] that anticipated some of the innovative services that are now feasible by the WWW (and some more indeed, including roles, decentralized access control, and set oriented query processing). The basic approach of the personal model of data, as well as of the WWW, is that an individual as data subject retains full technical control over the storage of his data, which is locally held on a computing system of his choice and under his supervision and auditing. Surely, a participant that traditionally would own and maintain a database for his purposes still needs personal data, and thus we have to provide the appropriate means for this requirement. In the personal model these means are roughly layered as follows.

Firstly, the participant must hold an “authority” which can be interpreted as the access right (in today’s CORBA terms a credential [39]) to execute the commands necessary to pursue the purpose in question. Secondly, the participant must be “acquainted” with the data subject in the sense that the participant knows the data subject’s unique identifier in the network (in today’s WWW terms the URL), in order to be able to direct the data request appropriately. Thirdly, we need the informational infrastructure which allows on-line procedures of the sort sketched here. And finally, after an autonomously performed access control and auditing action, the data subject’s computing system has to correctly react indeed by transmitting the requested data.

This shift from the traditional paradigm of reference books implemented as centralized databases to the proposed paradigm of communicating personal
data agents already happens everyday in some sectors of the information society. These sectors are characterized by the supposed joint interest of data providers and data consumers to cooperate, in particular by the consumer's expectation and trust that requested data is properly provided whenever the request is legitimate. In other sectors, the consumer's concern on the availability of data still prevails the data provider's concern on confidentiality and control, as postulated by the principle of informational self-determination.

Interestingly, for both paradigms the concern on integrity of data turns out to be subtly distributed over both sides, and it strongly depends on the mutual trust among data providers and consumers (and infrastructure providers). Indeed, in the framework of informational assurances, both for availability and integrity, the new paradigm would demand for new legal rules and technical enforcement mechanisms in order that the data providers and their computing systems always cooperate as expected.

Whereas within the paradigm of communicating personal data agents a data subject retains full technical control over the primary storage of his data, he would still be left with some of the problems related to using that data once it has been communicated. Hence that paradigm would have to be complemented by further legal rules and enforcement mechanisms. The legal rules should disallow to permanently store communicated personal data, at least in a large scale (while demanding its mandatory availability on necessary demand at the data subject's site). And we could develop technical enforcement mechanisms for controlling the usage of personal data by exploiting techniques that have been introduced for digital money and electronic commerce (cf. e.g. [12, 56]), for instance fingerprints against unauthorized passing of electronic goods or measures against double spending of coins (see Section 2.3.2 below).

Reviewing the observations concerning the technical difficulties with the principle of privacy, as discussed in Section 2.1.1, we see that the new paradigm could offer promising solutions to many of them, but the fourth difficulty related to data about social relationships would remain, unless we could additionally find new forms of cooperative data representation and access control.

2.3.1.2 Federated database systems and mediated information systems. Besides the two extreme kinds of storing personal data we see also further informational services, in particular federated database systems [37, 52] and mediated information systems [57, 58]. Both services introduce new layers between the data providers and the data consumers, in order to assist participants of the information society in dealing with the increasing scope and complexity of information management. Obviously these additional layers also challenge us with respect to informational assurances. Most work for federated database systems has been devoted to resolving the heterogeneity of access rights among the components allowing them to perform access control widely autonomously, see for example [2, 23, 26, 33, 34]. Some recent work for mediated information systems also deals with the necessary trust in the intermediate layers and related problems [9, 22]. The work in both fields is apparently done
under the implicit assumption of a relative small number of components. It would be necessary to explore to which extent the results can be scaled for the tremendous number of participants in a system of communicating personal data agents.

2.3.1.3 Trust in certificates. In federated systems participants want to autonomously decide on their trust in certificates, as they are required, among others, for the public keys, which are used for verifying digital signatures or for encrypting confidential messages. Like for any other problem of informational assurances we have to consider the impact of many viewpoints. Of course, one viewpoint is the status of legal regulations. As presented in Section 2.2.5 and Section 2.2.6, there is substantial progress with respect to digital signatures, but, unfortunately, due to political debates on the conflicting goals of national security and law enforcement, not for encryption yet. Another viewpoint is the design of actual systems. Here we see already established systems like "Pretty Good Privacy", PGP [59], for enabling participants to autonomously employ end-to-end cryptography, both for signatures and encryption, or system specifications like CORBA [39], for allowing autonomous access control in federated object systems.

At the bottom of any consideration, a specific user has to evaluate to what extent he is willing to trust a certificate. Since such a certificate may be generated by a chain of actions of diverse participants, whether along hierarchies or within a "web of trust", the task of trust evaluation is quite subtle. One may wonder whether this task can be technically supported at all, because it mostly deals with social relationships. On the other hand, the mass of daily electronic transactions could require to elaborate on a formal model to automate routine decisions. A specific proposal for such a model and a discussion of other approaches can be found in [38].

2.3.2 Some cryptographic protocols enabling cooperation under threats

Cryptography can be considered as the discipline in computing which aims at cooperation under threats. If used in a decentralized fashion, as enabled by the asymmetric cryptography, it allows individuals to technically enforce many of their informational interests, including confidentiality, detection of loss of integrity, authenticity and non-repudiation. It must be emphasized, however, that asymmetric cryptography must be firmly founded in both the (more or less social) trust in certificates for public keys, as discussed before, and in tamper resistant hardware devices, considered below in Section 2.3.3.

This presentation is not the place to survey cryptography what has excellently been done for instance by [50]. Here we only mention some selected work that aims at providing documented evidence on happened events and on anonymity. Both features are important for the technical enforcement of informational assurances, and it would be worthwhile to exploit their potentials for the specific problems of privacy, in particular within the paradigm of communicating personal data agents.
Fail-stop signature schemes [40, 42] are a new class of digital signature schemes, which improve previously known schemes in case that somebody (unexpectedly) succeeds in forging a signature. Of course, such an unhappy event should not occur, but, unfortunately, we cannot totally exclude the possibility. For the security (in terms of unforgeability) of all known schemes is based on unproven assumptions in the theory of computational complexity, in particular on the famous assumption $P \neq \text{NP}$. Now, if a forgery actually happened for a fail-stop signature scheme, then a claimed but not actual signer can prove that forgery by demonstrating that the complexity assumption has been broken. The innovative signing protocol just delivers the necessary evidence to convince a court about this fact, and thus such a protocol can strengthen the situation of a (socially weak) signer against a (socially powerful) verifier.

Informational cooperation may require to exchange digital goods or to digitally sign contracts. The exchange or signing scheme must be fair in the sense that, even if one of the participants misbehaves, either both participants or none of them obtain what they expected. The classical pessimistic way is to ask a third party for assistance but at the price of extra costs. Optimistic exchange and contract signing schemes [1, 44] reduce that costs in that the third party is not actively involved in the fault-less case. Thus optimistic schemes are more suitable for the daily routine cases but still provide enough evidence for solving disputes in hopefully exceptional cases. If we consider personal data as a digital good which is provided on the basis of contracts, we could exploit such schemes for the paradigm of communicating personal data agents.

As mentioned before, privacy implies that a data subject retains control over both the primary storage of personal data and its usage once it has been provided to some consumer. For ordinary digital goods like copyrighted documents or software as well as for personal data a particular challenge is to control unauthorized proliferation. Again the provider is interested in producing some non-repudiatable pieces of evidence that he has delivered a specific copy of the item to a specific receiver. Recent progress on asymmetric fingerprinting schemes [45, 46, 47] already achieves this goal for large electronic goods like pictures.

Fingerprinting cannot prevent the passing of data but can only deter participants to transmit data if not authorized. Technically enforced strict prevention has been studied in the framework of digital money in order to avoid the double spending of electronic coins. On-line prevention techniques restrict the availability of the informational services under consideration and the autonomy of the participants. Off-line prevention appears to require what has been called "wallets with observers" [21]. The "observer" is an electronic substitute of that participant that has an interest in controlling an electronic action of another participant. That substitute is physically implanted into the computing device of the participant to be controlled. In case of electronic coins, the observer is the substitute of the bank that wants to control its client. This scenario assumes that the controlling participant (the bank) can oblige the controlled participant (the client) to use the customized tamper resistant computing de-
vice with the implanted observer for the informational cooperation (spending a
coin). Surely this scenario is completely different from the privacy scenario
where everybody would act as a controlling participant regarding any receiver
of his data as a participant to be controlled. So, obviously we are still far away
from the ultimate goal of privacy.

The seminal work of [19, 20] introduced the possibility of anonymity and
digital pseudonyms for informational cooperation. Surely these features could
be very important for the field of health care when personal data is required to
be provided beyond the protected environment of professional secrecy, which
is constituted by the special relationship of a patient and a physician or other
persons directly caring for the patient. Then confidentiality and privacy re­
quirements on the one side could be maintained while also supporting other
interests like fair clearing procedures for health care providers and health in­
surances on the other side. A first study of feasibility [10, 11] has shown that
both features could be achieved indeed. Surely, before introducing the proposed
new schemes we would have to study the technical details as well as the social
implications more deeply.

Digital pseudonyms allow non-observability of a participant’s behaviour on
the application layer. On the communication layer, we would also like to protect
participants against observing their activities on the communication network,
i.e., their sending and receiving of messages. This goal can be achieved to
some extent by so-called mixes [19, 32]. In a mix communication network
each exchange node is organized as a mix where on each round messages are
gathered, cryptographically recoded (decrypted and reencrypted), resorted, and
retransmitted. As a result, adversary observers cannot trace messages travelling
through the network.

2.3.3 Tamper resistant hardware foundation

Any technical enforcement mechanisms have to be founded somehow within
the hardware. In particular, if a participant wants to enforce his interests by
some cryptographic scheme then his protection crucially depends on his reliable
control over generating, storing and using the secret keys. For this purpose he
needs a personal computing device, which in particular physically isolates the
cryptographic secrets. These devices must be tamper resistant in the sense that
they are physically protected against unauthorized attempts to read or modify
their contents, the secrets as well as their programs.

Again many viewpoints have to be considered, see e.g. [41] for a recent discus­sion. Among them are legal and social rules for manufacturing and distributing
such devices, and both rules and technical measures for dealing with loss or
theft of such devices. Since the devices are devoted to informational coopera­tion with other participants, also their potentially conflicting interests have
to be honoured. The “observers”, mentioned in the preceding Section 2.3.2,
are an example of providing a physically implemented electronic substitute of
those participants.
2.4 A SUMMARY WITH RESPECT TO PRIVACY

I advocate considering the issue of privacy within a more comprehensive framework of informational assurances, which take care of both restricting and enabling participation in the "information society". The informational assurances include technical mechanisms enforcing pertinent laws and related social and legal rules. As far as possible at all, technical mechanisms should be physically controlled by those participants whose interests are enforced.

Privacy, understood as informational self-determination, demands control over the primary storage, the transmission and the usage of personal data and over the knowledge about personal behaviour within the computing system under consideration.

Control concerning personal data seems to be best achievable if we treat personal data like any other electronic good in electronic commerce. Then personal data would be primarily stored in personal data agents, which communicate on demand. An agent transmits data if and only if required by the consumer and autonomously agreed on by the supplier who controls the agent. Transmission of personal data would be only one part of a more comprehensive electronic transaction of contract signing and fair exchange. Exploiting techniques developed for electronic commerce, like digital signatures, fingerprinting, "observers" and others, a data subject could be provided with technical means to control usage of his data once it has been transmitted, in particular by producing non-repudiatable pieces of evidence to deter the misuse of data.

Control over knowledge about personal behaviour require informational services which allow anonymity and digital pseudonyms. These services have to be offered on the application level and on the communication level. While on the application level an individual can have the direct disposal of his anonymous or pseudonymous credentials, on the communication level network providers can only be indirectly and socially supervised.

Acknowledgments

I sincerely thank U. Flegel, R. Menzel and T. Polle for carefully reading draft versions and helping to improve the presentation. I am also grateful to K. Rennenberg for valuable hints.

References


[27] European Commission, Directive on the protection of individuals with regard to the processing of personal data and the free movement of such data, 1995.


II  Workflow
ANALYZING THE SAFETY OF WORKFLOW AUTHORIZATION MODELS
Wei-Kuang Huang and Vijayalakshmi Atluri

Abstract: Workflow Management Systems (WFMS) are being widely used today by organizations to coordinate the execution of various applications representing their day-to-day tasks. To ensure that these tasks are executed by authorized users or processes (subjects), and to make sure that authorized subjects gain access on the required objects only during the execution of the specific task, granting and revoking of privileges need to be synchronized with the progression of the workflow through proper authorization mechanisms. Recently, Atluri and Huang have proposed a workflow authorization model (WAM) that provides such synchronization. This paper, first extends WAM to support roles and authorization constraints such as separation of duties. Second, it develops methodologies to analyze the safety of workflow authorization model when authorization constraints are imposed. The analysis is carried out by modeling WAM as a suitable Petri net (PN) and by utilizing the well-established analysis techniques of PNs.

3.1 INTRODUCTION

Workflow Management has emerged as the technology to automate the coordination of day-to-day activities (called tasks) of business processes. Today wide use of Workflow Management Systems (WFMS) can be found in a number of domains including office automation, finance and banking, healthcare, telecommunications, manufacturing and production. The various tasks in a workflow are executed by several users or programs according to the organizational rules relevant to the processes represented by the workflow. To ensure that these tasks are executed by authorized users or processes (subjects), proper authorization mechanisms must be in place. Moreover, to make sure that authorized
subjects gain access to the required objects only during the execution of the specific task, granting and revoking of privileges need to be synchronized with the progression of the workflow. Recently, Atluri and Huang have proposed a workflow authorization model (WAM) [2] that provides such synchronization.

It is common to many organizations to express security policies in terms of roles rather than users. For example, a nurse is authorized to administer a medication to a patient. Roles represent organizational agents intended to perform certain job functions within the organization. Users in turn are assigned appropriate roles based on their qualifications. Such role based authorization simplifies security administration. In addition, rules specifying separation of duties are imposed to reduce the risk of frauds by not allowing any individual to have sufficient authority within the system to perpetrate a fraud on his own. Such authorization constraints can be found in many application domains. For example, in a paper reviewing process, a person is never allowed to review his/her own paper, and a paper must be reviewed by at least three different individuals.

Although WAM is capable of providing the synchronization of authorization flow with the workflow and supports role-based authorization, it is not capable of supporting other essential requirements such as separation of duties. Although many research prototypes and commercial WFMS products are available today (e.g., Lotus Notes) that provide support for role-based authorization, they are not capable of modeling separation of duties. Since no proper support is provided, currently these constraints have to be implemented as ad hoc application code.

The significant work in this direction is due to Sandhu [10]. However, this research is not adequate to specify separation of duties in WFMS environment since it does not specify access control in terms of tasks. Recently, Bertino et al. [3], have identified several types of authorization constraints, including separation of duties. They have categorized constraints imposed on role and user assignments to tasks into three types: static, dynamic and hybrid constraints and have developed an approach to determine their consistency by first expressing the authorization constraints as clauses in a logic program. They have also proposed algorithms to check for the consistency of the constraints and to assign users and roles to tasks that constitute the workflow in a such a way that no constraints are violated. The primary emphasis of [3] is verification of the consistency of authorization constraints. However, since the predicates representing the constraints do not include the objects, it is not capable of distinguishing one workflow instance from the other. Thus, it is not capable of modeling inter-instance authorization constraints.

In this paper, we show how authorizations can be derived when constraints expressing separation of duties are enforced. This requires examining the current state of the set of authorizations as well as the state of the workflow. Our approach conducts a run-time evaluation of the state to authorize users to execute a task. The major distinction between our work and that proposed in [3] is that our WAM is capable of modeling separation of duties constraints on
multiple workflow instances thus is able to capture inter-instance constraints. This is because, authorization constraints are specified consisting of authorizations themselves as predicates. This will also allow enforcement of constraints based on authorizations derived by other means, but not necessarily through the workflow executions.

In this paper, we also provide a formal model based on Petri nets by enhancing the Color-Timed Petri Net proposed in [2] to model role-based access control with separation of duties. Representing WAM as a Petri net allows one to visually depict the workflow behavior through its graphical representation and to analyze its behavior through its rich set of analysis techniques. Analysis helps one to understand the implications of the authorization policies. Although each policy may appear innocent in isolation, their cumulative effect may lead to an undesirable authorization state [11]. (See section 3.3 for a definition of authorization state.) So for a given initial authorization state and a set of security policies specified by authorization rules, analysis requires determining all the reachable authorization states. This, known as the safety problem, first identified by Harrison, Ruzzo and Ullman [6], specifically can be stated as the following question: "Is there a reachable state in which a particular subject possesses a particular privilege for a specific object?" We develop methodologies to analyze the safety of workflow authorization model when authorization constraints such as separation of duties are imposed.

3.2 WORKFLOW AUTHORIZATION MODEL

To ensure that authorized subjects gain access on the required objects only during the execution of the specific task, WAM synchronizes authorization flow with the workflow by synchronizing granting and revoking of privileges with the initiation and completion of the tasks. To achieve this synchronization, WAM associates an Authorization Template (AT) with each task, which specifies the static parameters of the authorization that can be defined during the design of the workflow. When a task starts its execution, this AT is used to derive the actual authorization. In this section, we review WAM with its extension to incorporate role based authorizations.

A workflow \( W \) can be represented as a partially ordered set of tasks \( \{tw_1, tw_2 \ldots tw_n\} \), where each task \( tw_i \) in turn can be defined as a set \( OP_i \) of a partial or total order of operations \( \{op_1, op_2 \ldots op_n\} \) that involve manipulation of objects [5]. Processing of a task involves accessing certain objects by certain subjects with certain privileges. To execute a task \( tw_i \), relevant privileges on required objects have to be granted to appropriate subjects.

Let \( S = \{s_1, s_2 \ldots \} \) denote the set of subjects, \( O = \{o_1, o_2 \ldots \} \) the set of objects \( \Gamma = \{\gamma_1, \gamma_2 \ldots \} \) the set of objects types and \( R = \{r_1, r_2 \ldots \} \) the set of roles. The function \( F : O \rightarrow \Gamma \). That is, if \( F(o_i) = \gamma_j \), then \( o_i \) is of type \( \gamma_j \). \( G : S \rightarrow R. \) I.e., if \( G(s_i) = r_j \), then \( s_i \) is of role \( r_j \). Let \( PR \) denote a finite set of privileges. We use \( S_{r_i} \) to denote the set of subjects that belong to role \( r_i \) and \( O_{\gamma_i} \) to denote the set of objects of type \( \gamma_i \).
Definition 1 1. Time set $\mathcal{T} = \{\tau \in \mathcal{R.} \mid \tau \geq 0\}$
2. a time interval $\{[\tau_l, \tau_u] \in \mathcal{T} \times \mathcal{T} \mid \tau_l \leq \tau_u\}$ represents the set of all closed intervals.

According to definition 1 an interval is defined by its lower and upper bounds, $\tau_l$ and $\tau_u$, respectively, where each of $\tau_l$ and $\tau_u$ can either be a constant or an expression.

Definition 2 A task $t_{Wi}$ is defined as $(OP_i, \Gamma_{IN_i}, \Gamma_{OUT_i}, [\tau_l, \tau_u])$, where $OP_i$ is the set of operations to be performed in $t_{Wi}$, $\Gamma_{IN_i} \subseteq \Gamma$ is the set of object types allowed as inputs, $\Gamma_{OUT_i} \subseteq \Gamma$ is the set of object types expected as outputs, and $[\tau_l, \tau_u]$ is the time interval during which $t_{Wi}$ must be executed.

Here $[\tau_l, \tau_u]$ specifies the temporal constraint stating the lower and upper bounds of the time interval during which a task is allowed to be executed.

Definition 3 A task-instance $t_{Wi}$ is defined as: $(OPER_i, IN_i, OUT_i, [\tau_l, \tau_u])$ where $OPER_i$ is the set of operations performed during the execution of $t_{Wi}$, $IN_i$ is the set of input objects to $t_{Wi}$ such that $IN_i = \{x \in O \mid F(x) \in \Gamma_{IN_i}\}$, $OUT_i$ is the set of output objects from $t_{Wi}$ such that $OUT_i = \{x \in O \mid F(x) \in \Gamma_{OUT_i}\}$, and $[\tau_l, \tau_u]$ is the time interval during which $t_{Wi}$ has been executed.

Whenever a task is executed, a task-instance will be generated. Thus, a task $t_{Wi}$ may generate several $t_{Wi}$’s. $\tau_l$ and $\tau_u$ in the above definition indicate the time at which that particular task-instance has started and finished execution, respectively, whereas $[\tau_l, \tau_u]$ represent the time during which the task is allowed to be executed. Note that $[\tau_l, \tau_u]$ may differ from $[\tau_l, \tau_u]$. However, to ensure the temporal constraints, $[\tau_l, \tau_u]$ must be within $[\tau_l, \tau_u]$.

Definition 4 An authorization is a 4-tuple $A = (s, o, pr, [\tau_b, \tau_e])$, where subject $s$ is granted access on object $o$ with privilege $pr$ at time $\tau_b$ and is revoked at time $\tau_e$.

An authorization base $AB = \{A_1, A_2 \ldots\}$ is a finite set of authorizations. As workflow execution progresses, all authorizations that have been generated are added to the set $AB$.

Definition 5 Given a task $t_{Wi}$, an authorization template $AT(t_{Wi})$ is defined as a 4-tuple $AT(t_{Wi}) = ((r_i, -), (\gamma_i, -), pr_i, [\tau_l, \tau_u])$ where
(i) $(r_i, -)$ is a subject hole which can be filled by a subject $s_i$ where $G(s_i) = r_i$,
(ii) $(\gamma_i, -)$ is an object hole which can be filled by an object $o_i$ where $F(o_i) = \gamma_i$,
(iii) $pr_i$ is the privilege to be granted to $s_i$ on object $o_i$.
(iv) $[\tau_l, \tau_u]$ is the time interval during which the task must be executed.

In the definition for $AT(t_{Wi})$ (i) says that only subjects belonging to role $r_i$ is allowed to execute $t_{Wi}$ thus the subject hole $(r_i, -)$ allows only subjects that belong to role $r_i$, (ii) dictates that only objects of type $\gamma_i$ can be processed by
ANALYZING THE SAFETY OF WORKFLOW AUTHORIZATION MODELS

Thus the object hole \( (\gamma_i, -) \) allows objects of only type \( \gamma_i \), (iii) says that a subject requires a privilege \( pr_i \) on the objects that arrive at \( tw_i \) for processing, and (iv) says the default interval for the authorization template will be the valid time interval for the task.

Authorization templates are attached to the tasks in a workflow. A task \( tw_i \) may have more than one authorization template associated with it. More \( AT \)'s are required in cases where there are more than one type of object to be processed, or more than one subject is required to perform the processing. To distinguish the privileges in \( AT \) from those in \( A \), we often use \( pr(AT) \) to denote the privilege component of an authorization template \( AT \). An authorization template enables one to specify rules such as “Only a clerk is allowed to perform check preparation during time 10 and 50.” These can actually be stated during the design process by the workflow designer.

**Definition 6** [Authorization Derivation Rule]

Given an authorization template \( AT(tw_i) = ((r_i, -), (\gamma_i, -), pr_i, [\tau_i, \tau_{ui}]) \) of task \( tw_i \), an authorization \( A_i = (s_i, o_i, pr_i, [\tau_i, \tau_{ei}]) \) is derived as follows:

**Grant Rule:** Suppose object \( X \) is sent to subject \( y \) at \( \tau_{ai} \) to start \( tw_i \).

If \( X \in O_{\gamma_i} \) and \( y \in S_{r_i} \) and \( \tau_{ai} \leq \tau_{ui} \),

\[
s_i \leftarrow y, o_i \leftarrow x, pr_i \leftarrow pr(AT);
\]

\[
\tau_{ei} \leftarrow \tau_{ui},
\]

if \( \tau_{ai} \leq \tau_i, \tau_{bi} \leftarrow \tau_i; \) otherwise \( \tau_{bi} \leftarrow \tau_{ai}. \)

**Revoke Rule:** Suppose \( tw_i \) ends at \( \tau_{fi} \) at which point \( x \) leaves \( tw_i \).

If \( \tau_{fi} \leq \tau_{ui}, \tau_{ei} \leftarrow \tau_{fi}. \)

**Example 1** We explain now how authorizations are derived from the authorization templates with an example. Considering a check processing example [2] consisting of three tasks \( tw_1, tw_2 \) and \( tw_3 \) denoting prepare check, approve check and issue check, respectively. They can be expressed as follows:

\[
tw_1 = \{(\text{prepare check}), \{\text{check}\}, [10,50]\}
\]

\[
tw_2 = \{(\text{approve check}), \{\text{check}\}, [20,60]\}
\]

\[
tw_3 = \{(\text{issue check}), \{\text{check}\}, [40,80]\}
\]

Suppose the associated roles for performing these processes are clerk, manager and clerk, respectively. Assume Mary and John are clerks and Peter is the only manager. We define the following authorization templates.

\[
AT(tw_1) = ((\text{clerk}, -), (\text{check}, -), \text{prepare}, [10,50])
\]

\[
AT(tw_2) = ((\text{manager}, -), (\text{check}, -), \text{approve}, [20,60])
\]

\[
AT(tw_3) = ((\text{clerk}, -), (\text{check}, -), \text{issue}, [40,80])
\]

Now suppose the check \( ck1 \) for payment arrives at time 40 and John starts \( tw_1 \). Both subject and object are filled into the authorization template \( AT(tw_1) \), generating an authorization (John, \( ck1 \), prepare, [40,50]). Suppose John finishes \( tw_1 \) at 47, then the authorizations on \( ck1 \) are revoked for John by replacing the upper bound with 47, thus forming the authorization (John, \( ck1 \), prepare, [40,47]). Similarly, other authorizations will be derived as approve and issue tasks are executed.
3.3 WAM WITH SEPARATION OF DUTIES

Separation of duties can be expressed as constraints. In [3], Bertino et al. have identified several types of authorization constraints, including separation of duties. Similarly, we also express separation of duties as rules.

**Definition 7** Given an authorization template \( AT(t_{wi}) = ((r_i, -), (\gamma_i, -), pr_i, \tau_{wi}, \tau_{ui}) \), we define a set of potential authorizations, \( PA_i \), representing all possible authorizations that can be potentially derived from \( AT(t_{wi}) \). Each potential authorization \( pa \) in \( PA_i \) is a triple \((s_i, o_i, pr_i)\) such that \( s_i \in S_{r_i}, o_i \in O_{\gamma_i} \).

**Definition 8** Given an authorization \( A = (s, o, pr, [\tau_b, \tau_e]) \) in \( AB \), we define a non-temporal projection \( ANT \) of \( A \) as \( ANT = (s, o, pr) \). The non-temporal projection of \( AB, AB_{NT} = \{ANT_1, ANT_2 \ldots \} \).

In our formalism, we assume each constraint \( c_i \) is a logical expression of the form: \( q \leftarrow p \) where \( p \) is any logical expression consisting of \( ANT \) as literals and \( q \) is a single literal which is always either \( pa \) or \( \sim pa \) such that \( pa \in PA_i \) of some \( tw_i \). We also denote \( s(p) \) (or \( s(q) \)) as the set of subjects that are specified in \( pa \in PA_j \) (or \( pa \in PA_i \)). Enforcement of a constraint would either force or prohibit assignment of a specific user to a task. Therefore, separation of duties constraints fall into two categories. Note that this categorization is different from that of [3].

- **Exclusive type:** In this type, \( q \) is always of the form \( \sim pa \) where \( pa \in PA_j \) for some \( tw_j \).
- **Assertive type:** In this type, \( q \) is always of the form \( pa \) where \( pa \in PA_j \) for some \( tw_j \).

As an example of an exclusive constraint, consider once again the check processing example introduced in section 3.2. Suppose the business policy of the bank is such that it does not allow any single individual to both prepare and issue a check. This policy can be expressed as constraint \( c_1 \): A clerk who prepares a check cannot issue the same check. Formally, \( c_1 \) can be expressed as follows.

\[
c_1: (\forall x \in S_{clerk}, y \in O_{check}) \cdot (\sim (x, y, issue) \leftarrow (x, y, prepare))
\]

On the other hand, if the business policy states that a check can be issued only by an individual who prepares it. This can be formally stated as:

\[
c_2: (\forall x \in S_{clerk}, y \in O_{check}) \cdot ((x, y, issue) \leftarrow (x, y, prepare))
\]

Thus \( c_2 \) is an assertive type constraint. Based on whether a constraint is either assertive or exclusive type, certain users in the role of clerks are not eligible to execute the task of issuing a check. Thus, the set of eligible users for each task changes dynamically based on the current state of the authorization base. Moreover, the eligible users vary from one task-instance to another. Furthermore, only certain constraints play a role in deciding the eligibility of subject to execute a task. For example, \( c_1 \) affects the eligible set of subjects of \( tw_3 \) but not of \( tw_1 \) or \( tw_2 \) in our check processing example. Therefore we determine first the set of relevant constraints for each task \( tw_i \), denoted as \( C_{tw_i} \).
Definition 9 We define $C_{twi}$ as follows: $C_{twi} = \{c_j \mid c_j \in C \text{ which is of the form } q_i \leftarrow p_j \text{ and } q_i \in PA_i\}$.

Then, for each task, we define a set of eligible subjects, denoted as $S_{x}^{o}(o)$ with respect to object $o$.

Definition 10 Given an authorization template $AT(t_{wi}) = ((r_{i}, -), (\gamma_{i}, -), pr_{i}, [\tau_{i}, \tau_{ui}])$, we define the set of eligible subjects $S_{x}^{o}(o)$ as follows:

1. $S_{x}^{o}(o) = S_{r_{i}}$ if $C_{twi} = \emptyset$
2. $S_{x}^{o}(o) = S_{r_{i}} - s(q_{i})$ if $c_{i} : q_{i} \leftarrow p_{j} \in C_{twi}$ is an exclusive constraint and $p_{j}$ is true with respect to $ABNT$
3. $S_{x}^{o}(o) = s(q_{i})$ if $c_{i} : q_{i} \leftarrow p_{j} \in C_{twi}$ is an assertive constraint and $p_{j}$ is true with respect to $ABNT$

The above definition says that if the constraint specifying the separation of duties is of exclusive type, the set of eligible subjects is obtained by subtracting the disallowed subjects from the set of subjects playing the role assigned to execute the task. On the other hand, if the constraint is of assertive type, the set of eligible subjects is simply the set of subjects specified in $pa \in q$. If no constraints affect the task, then the set of eligible subjects is same as the set of subjects playing the role. An appropriate authorization must be generated from the authorization template at run time in such a way that the subject to execute the task must be chosen from the set of eligible subjects and only when the task receives an object with type specified in the authorization template.

The following authorization derivation rule ensures this requirement.

Definition 11 [Authorization Derivation Rule for extended WAM] Given an authorization template $AT(t_{wi}) = ((r_{i}, -), (\gamma_{i}, -), pr_{i}, [\tau_{i}, \tau_{ui}])$ of task $t_{wi}$, an authorization $A_{i} = (s_{i}, o_{i}, pr_{i}, [\tau_{bi}, \tau_{ei}])$ is derived as follows:

Grant Rule: Suppose object $x$ is sent to subject $y$ at $\tau_{ai}$ to start $t_{wi}$.
If $x \in O_{\gamma_{i}}$ and $y \in S_{x}^{o}(x)$ and $\tau_{ei} \leq \tau_{ui}$,

$s_{i} \leftarrow y, o_{i} \leftarrow x, pr_{i} \leftarrow pr(AT)$;
$\tau_{ei} \leftarrow \tau_{ui}$,
if $\tau_{ai} \leq \tau_{ii}, \tau_{bi} \leftarrow \tau_{ii}$; otherwise $\tau_{bi} \leftarrow \tau_{ai}$.

Revoke Rule: Suppose $w_{i}$ ends at $\tau_{fi}$ at which point $o_{i}$ leaves $t_{wi}$.
If $\tau_{fi} \leq \tau_{ui}, \tau_{ei} \leftarrow \tau_{fi}$.

Note that the only difference between the above definition and earlier derivation rule in definition 6 is the set from which a subject is chosen. Without the separation of duties constraints, a subject is chosen from the set of subjects playing the specified role, whereas with separation of duties constraints, a subject is chosen from the set of eligible subjects which may be a subset of the previous set.

Definition 12 Given a workflow $W$, we define a workflow authorization state $A^{\ell}(W)$ as the set of current non-temporal projection of authorizations derived during the execution of $W$. 
For instance, in example 1, a workflow authorization state \( A^w(W) \) could be \( \{(John, ck2, prepare), (Peter, ck1, approve)\} \), meaning that an authorization is granted to John for preparing \( ck2 \) and to Peter to approve \( ck1 \). In the following, we explain the process of deriving authorizations by taking an example.

**Example 2** Consider once again example 1. Suppose a separation of duties constraint is specified as \( (\forall x \in S_{clerk}, y \in O_{check}) \cdot (\sim (x, y, issue) \leftarrow (x, y, prepare)) \). Since \( S^e_1(ck1) = subject(dclerk) = \{John, Mary\} \), Mary \( \in S^e_1(ck1) \), and \( ck1 \in O_{check} \), after the execution of task \( tw_1 \), an authorization \((John, ck1, prepare, [40,47])\) is generated. Similarly, after the execution of task \( tw_2 \), an authorization \((Peter, ck1, approve, [47,54])\) is generated. As a result of the constraint, the eligible set of subjects authorized to execute \( tw_3 \) are evaluated as follows: \( S^e_3(ck1) = S_{clerk} - \{s(John, ck1, prepare)\} = \{Mary, John\} - \{John\} = \{Mary\} \). In other words, only Mary is allowed to execute \( tw_3 \). Thus the separation of duties constraint is satisfied.

### 3.4 SAFETY ANALYSIS OF WAM

In this section, we first present the Petri net representation of WAM through which we perform the safety analysis. Then we present the algorithm to test for the safety of WAM. We also report the implementation status of this module.

#### 3.4.1 Color Timed Petri Net (CTPN) - A Model to Represent Workflow Authorization Model

In this section, we present the **Color Timed Petri Net (CTPN) Model** [2] to represent the WAM proposed in the previous section. Our CTPN is an extension of colored and timed Petri nets [7], which in turn are extensions of an ordinary Petri net.

**Definition 13** A Color Timed Petri Net (CTPN) is a tuple \( CTPN = (PN, \Sigma, CR, E, IN, D, ts) \) where

1. \( PN = (P, T, F, M) \) is an ordinary Petri net.
2. \( \Sigma = \{\sigma_1, \sigma_2, \ldots\} \) is a finite set of colors (or types),
3. \( CR \) is a color function such that \( CR(p) \subseteq \Sigma \), and \( CR(m(p)) \subseteq CR(p) \),
4. \( E \), the arc function such that: \( \forall f(p, t), f(t, p) \in F, E_f \subseteq CR(p)_{MS} \),
5. \( IN \) is an interval function associated with a transition, i.e., \( IN : T \rightarrow T \times T \) such that \( IN(t_i) = [\tau_{li}, \tau_{ui}] \) where \( [\tau_{li}, \tau_{ui}] \in T \times T \) and \( t_i \in T \),
6. \( D \) is a delay function associated to a place. i.e., \( D : P \rightarrow T \times T \) such that \( D(p_j) = [\delta^m_j, \delta^M_j] \), where \( [\delta^m_j, \delta^M_j] \in T \times T \) and \( p_j \in P \), and
7. \( ts \) is a timestamp function such that \( ts(m(p_k)) = \alpha_k \in T \), which denotes the arrival time of the token to \( p_k \).

We represent a token as \((v, x)\) where \( v \) is the color of the token and \( x \) the timestamp. Whenever a token moves from one place to another through a fired transition, its timestamp is modified to the firing time of the transition.
We use $m(p)$ to denote the marking of place $p$. $m(p)$ is expressed as a multi-sets of tokens with respect to distinct colors. For example, $m(p) = g + r$ represents place $p$ containing a token of color $g$ and a token of color $r$, i.e., $CR(m(p)) = \{g, r\}$.

The above definition dictates that each token has a color which is defined in the color set $\Sigma$. Each place has a color set (i.e., denoted as $CR(p)$) attached to it which specifies the set of allowable colors of the tokens to enter the place. For a token to reside in a place, it must satisfy that the color of token is a member in the color set of the place. Each arc $f(p, t)$ or $f(t, p)$ is associated with a color set such that this set is contained in the multi-sets of $CR(p)$. A transformation of colors may occur during firing of a transition. The firing of a transition is determined by the firing rules and the transformation by the arc function $E$. The firing rules can be formally stated as follows:

**Definition 14** Given a transition $t_i$ such that $IN(t_i) = [\tau_{ti}, \tau_{ui}]$, $\forall p_j \in \bullet t_i$ and $\forall p_k \in t_i \bullet$, for any $p_j$ marked with tokens $(v_{j1} , x_{j1}) , (v_{j2} , x_{j2}) \ldots (v_{jn} , x_{jn})$,

1. $(v_{jl} , x_{jl})$ is said to be available only during the interval $[\delta_j^m + x_{jl}, \delta_j^M + x_{jl}]$,
2. $t_i$ is said to be enabled at time $x$ if $E_f(p_j,t_i) \leq m(p_j)$ and all tokens in $p_j$ are available at time $x$.
3. an enabled $t_i$ is firable if $\max \{ (\delta_j^m + x) \mid p_j \in \bullet t_i \} \leq \tau_{ui}$ and $\min \{ (\delta_j^M + x) \mid p_j \in \bullet t_i \} \geq \tau_{ti}$ is true. A firable transition may fire any time during the firable interval, $[\max \{\tau_{ti} , \max \{\delta_j^m + x \mid p_j \in \bullet t_i \}\} , \min \{\tau_{ui} , \min \{\delta_j^M + x \mid p_j \in \bullet t_i \}\}]$.
4. Suppose $t_i$ fires at $\tau_i$. Firing of $t_i$ results in a new marking $M'$ as follows: $m'(p_k) = m(p_k) + E_f(t_i,p_k)$ and $m'(p_j) = m(p_j) - E_f(p_j,t_k)$ and the timestamp of each element in $m'(p_k)$ is $\tau_i$.

A transition $t_i$ is enabled only if all of its input places $p_j$ contain at least as many available tokens of the type as that specified in the arc function $E_f(p_j,t_i)$ of the corresponding $f(p_j,t_i)$.

The delay associated with a place represents minimum $\delta^m(p)$ and maximum $\delta^M(p)$ delay a token is required to remain in that place after its arrival. The delay can be a constant $d$ where $\delta^m(p) = \delta^M(p)$. A token is said to be available only after the delay $D(p_j)$ has elapsed. On the other hand, the time interval associated with a transition states that it can fire only during this interval, irrespective of the tokens' timestamps in its input places. A transition is said to be enabled only if each of its input places has an available token.

A transition $t_i$ fires only if its enabling time falls within the specified time interval $IN(t_i)$. When more than one input place exists, the transition fires after the maximum delay of all the input places has elapsed. Both the time interval and the delay can be specified as variables instead of fixed values. Upon firing, a transition $t_i$ consumes as many tokens of colors from each of its input places $p_j$ as those specified in the corresponding $E_f(p_j,t_i)$ and deposits as many tokens with specified colors into each output place $p_k$ as those specified in the corresponding $E_f(t_i,p_k)$. That is, the arc function of $f(p_k,t_i)$ specifies the number of tokens of specified colors to be removed from $p_j$ when $t_i$ fires,
and the arc function $f(t_i, p_k)$ specifies the number of tokens of specified colors to be inserted into $p_k$ when $t_i$ fires.

### 3.4.2 CTPN Representation of WAM

In the following, we illustrate how each component of WAM is represented in the CTPN. Given a task $tw$ with $AT(tw) = ((r, -), (\gamma, -), pr, [\tau_l, \tau_u])$, execution of $tw$ generates an authorization $A = (s, o, pr, [\tau_l, \tau_u])$.

1. A role $r$ is represented as a place with an associated color set that contains all subjects. (2) A object type $\gamma$ is represented as a place with a color set that contains all objects. (3) A subject, an object and a (subject, object) pair are represented by a token with respective color. (4) A subject assigned to a role $r$ is represented as a token deposited in a place $r$. (5) An object of type $\gamma$ is represented as a token in place $\gamma$. (6) A privilege $pr$ to perform the task is represented by a place with a color set expecting tokens of (subject, object) type. An arc is connected from grant transition to the privilege place and another from the privilege place to the revoke transition with the arc function of expected (subject, object) pair. (7) The grant and revocation processes are represented as input (grant) and output (revoke) transitions of the place representing the privilege, respectively. (8) A subject hole $(r, -)$ is represented as an input arc from place $r$ to the grant transition with an arc function specifying the subjects of role $r$. (9) An object hole $(\gamma, -)$ is represented as an input arc from place $\gamma$ to the grant transition with an arc function specifying the objects of type $\gamma$. (10) The time interval $[\tau_l, \tau_u]$ associated with the grant or revoke transitions denotes the specified interval during which the authorization is valid. (11) An authorization corresponds to a filled privilege place. (12) The time interval $[\tau_l, \tau_u]$ in an authorization is the time interval during which a token resides in place $pr$ (the difference between $\tau_l$ and $\tau_u$ is the duration for executing $tw$, denoted as $D(pr)$). (13) A constraint $c : q_i \leftarrow p_j \in C_{tw_i}$ can be represented by a subnet as follows: For each $A_j$ in $p_j$ create a place $C$ with an input arc $(f_1)$ from the grant transition of $pr_j$ in $p_j$ and an output arc $(f_2)$ to the grant transition of $pr_i$ in $q_i$, both with arc function $(x, o)$ where $x$ and $o$ are two variables. If $c_i$ is of assertive type, the arc function from the place representing $r_i$ is same as $x$ in $f_2$, otherwise if $c_i$ is of exclusive type, it will be a different variable than $x$. (14) A relative time constraint $d$ between two tasks can be associated as a delay $d$ at the constraint place.

Figure 3.1 shows the CTPN representation of the authorization model for a workflow consisting of role $r$, object type $\gamma$ and a task $tw$. $s_1 \ldots s_m$ are subjects assigned to role $r$ and $o_1 \ldots o_n$ denote objects of type $\gamma$. Ensuring that only objects of specified type and subjects of specified role are assigned a privilege $pr$ is reflected by the input arc functions to the grant transition of $pr$. The granting of an authorization is represented by the firing of $t_s$ when an object of type $\gamma$ arrives with an authorized subject from role $r$, thereby starting the execution of $tw$. Then a token of color $(x, o)$ is placed in $pr$. The authorization is thus derived based on the value (or color) and timestamp of the input tokens. Also note that the privilege $pr$ is not granted for $x$ on $o$ until $t_s$ fires and a token
with \((x,o)\) is deposited in \(pr\). The token resides in \(pr\) until \(t_f\) fires (i.e., as long as the task is executed). This firing removes the token from \(pr\) thereby revoking the authorization from the subject. Figure 3.2 shows the CTPN representation for the check processing example. The \(start\) and \(finish\) transitions represent the initiation and completion of each workflow instance.

### 3.4.3 Approach to Conduct Safety Analysis

In this section, we demonstrate how CTPN representation lends itself to address the safety problem of WAM. Our CTPN representation of WAM is such that the safety problem in authorization models is equivalent to the reachability problem in Petri nets. Reachability is a fundamental property for studying the dynamic properties of any system. It can be formally defined as follows.

**Definition 15** [9] A marking \(M\) is said to be reachable from a marking \(M_0\) if there exists a sequence of firings that transforms \(M_0\) to \(M\).

Since in our model, authorizations are represented as a marked place \(pr\) with token \((s,o)\), the safety question whether \(s\) possesses \(pr\) on \(o\) can simply
be answered by conducting a reachability analysis on the corresponding CTPN which answers whether a place \( pr \) will ever be marked by a token \((s,o)\). Thus,

\[
\text{Safety of WAM} \equiv \text{Reachability of its CTPN representation}
\]

since workflow authorization state in WAM is equivalent to marking of its corresponding CTPN.

Therefore, existing reachability analysis techniques, methods and results can be directly adopted to WAM. One may use one of the following three approaches to conduct safety analysis:

1. **Simulation**: Simulation is a technique to analyze a system by conducting controlled experiments. However, sometimes it is expensive and also it is not possible to use simulation to prove that the system has the desired properties because it is not a formal analysis technique.

2. **Reachability tree**: The basic idea behind the reachability tree is to construct a graph which contains a node for each reachable state and an arc for each possible change of the state. This is similar to unfolding the maximal state in [11] for analyzing TAM (The Typed Access Matrix Model). Since this represents all possible states, we can answer all safety questions. However, this tree can grow infinitely large even for a small PN (if they are unbounded). It has been shown in [8] that the reachability problem although decidable, has at least exponential space and time complexity.

3. **Matrix-equations**: In this approach, the dynamic behavior of a PN is captured in algebraic equations that can be represented as a matrix. Given an initial state, this technique allows us to determine whether a specific state is reachable. However, this requires the PN to be acyclic.

Fortunately, for our CTPN representation of workflow, we can adopt the less expensive matrix equation approach. Before we present our approach, we first recognize that our CTPN representation of a consistent workflow specification is acyclic in its structure, referred to as an acyclic Petri net, which does not contain cycles.

**Definition 16** [1] A workflow specification \( W \) is said to be inconsistent if either of the following is true: (1) the set of dependencies impose a cyclic execution order among some of the tasks in \( W \) (2) the set of dependencies specify that a set of complementary states (e.g., \( cm \) and \( ab \)) have to be reached for at least one task in \( W \).

In the above definition, the first condition specifies an inconsistent precedence relationship among the tasks. For example, a set of two dependencies where one dependency states task \( tw_1 \) can begin only after task \( tw_2 \) completes the execution, and the other dependency states \( tw_1 \) has to be executed before \( tw_2 \), would form an inconsistent precedence ordering between \( tw_1 \) and \( tw_2 \). The second requirement specifies inconsistent logical relationships among the task dependencies. For example, if we consider a set of dependencies where one states if \( tw_1 \) aborts, \( tw_2 \) has to abort and the other dependency specifies \( tw_2 \) has to commit if \( tw_1 \) aborts. There is an obvious inconsistency for \( tw_2 \) if \( tw_1 \)
aborts. Refer to [1] for a detailed discussion on consistency checking. It has been shown in [1] that if the workflow specification is consistent then there does not contain any cycle in the corresponding CTPN representation. This leads to the following proposition.

**Proposition 1** If the workflow specification \( W \) is consistent, then the corresponding CTPN is acyclic.

Our approach to conducting safety analysis relies on the following theorem.

**Theorem 1** [9] In an acyclic Petri net, a marking \( M_i \) is reachable from an initial marking \( M_0 \), iff there exists a nonnegative integer solution \( U \) satisfying

\[
M_i = M_0 + AU,
\]

where \( A \) is the corresponding incidence matrix that can be derived from the Petri net as follows: \( A \) is an \((m \times n)\) matrix such that \( m \) and \( n \) are the number of places and transitions, respectively, and each \( a_{ij} = a_{ij}^+ - a_{ij}^- \) where \( a_{ij}^+ \) (\( a_{ij}^- \)) is the number of arcs from transition \( j \) to its output (input) place \( i \). (See [9] for a proof.)

**Algorithm 1** [Safety Analysis of WAM]

Given a workflow specification \( W \), an initial state of authorization and the final state of authorization,

1. Construct CTPN of \( W \) according to the mapping in section 3.4.1.
2. Obtain the initial marking \( M_0 \) of CTPN by marking CTPN according to the given initial authorization state.
3. Obtain the final marking \( M_i \) of CTPN by marking CTPN according to the given final authorization state.
4. Solve \( M_i = M_0 + AU \). If \( U \) is a non-negative integer, output Yes and stop; otherwise output No.

The above algorithm presents the approach to test for the safety of WAM. The complexity of this algorithm is nothing but that of solving \( m \) equations with \( n \) variables where \( m \) and \( n \) represent the number of places and transitions, respectively.

### 3.4.4 Implementation

We have implemented a *Workflow Specification and Analysis Software* (WSAS) in Visual Basic. WSAS consists of two parts: The first part is to test whether the specification of the workflow is correct or not. The second part tests the safety of workflows, i.e., given an initial state of the workflow, it tests whether a specified state is reachable or not. WSAS has been built on top a prototype, called *IDEF/System Dynamics Evaluation Software*, developed by Boucher and Jafari [4], which assumes the underlying Petri net model is an ordinary Petri net. Currently we are working on enhancing WSAS to handle CTPN. Our approach is as follows. If the number of subjects is finite, the color sets associated with each place in a CTPN can be converted into a finite number of parallel arcs represented by an ordinary PN. By doing so, our WSAS can be utilized for conducting the analysis.
3.5 CONCLUSIONS

In this paper, we have shown how authorization constraints representing separation of duties can be incorporated into workflow authorization models. We have proposed a methodology to conduct safety analysis of workflow authorization models. We are currently extending the tool which we have developed to test for safety of workflows to address the safety of WAM. Since our software tool can only analyze ordinary Petri nets, we are currently investigating translating the CTPN to an ordinary PN so that our tool can be directly used to analyze WAM.

References


RULES AND PATTERNS FOR SECURITY IN WORKFLOW SYSTEMS

Silvana Castano and Maria Grazia Fugini

Abstract: Assignment of tasks to agents in a Workflow (WF) system should occur according to security policies regarding user authorizations to access data and documents through the WF tasks. This paper presents an approach to discretionary secure assignment of tasks to agents taking into account authorization constraints, in the framework of the WIDE (Workflow Interactive Development Environment) WF management system. The approach is based on the concepts of role, agent, and task, and on authorization patterns and rules. Security rules (or triggers) specify which actions (e.g., security warnings, logs, audit actions) should be taken when a security violation (event) occurs, following the ECA paradigm of active databases. A basic set of rules is provided in the abstracted form of authorization patterns which are generic rule skeletons to be properly instantiated to enforce authorization constraints in a given WF application.

4.1 INTRODUCTION

WFs are complex activities (or business processes) that involve the coordinated execution of several tasks to reach a common objective [8]. The design of WF applications requires the capability to cope also with security requirements taking into account the organization of users work and the structure of business processes [3]. Issues related to security in WFs, and distributed systems in general, have been receiving much interest in recent literature. A model to flexibly specify role-based authorization constraints in WF systems is described in [2]. Security in WFs is tackled in [9] where a discussion is provided regarding security controls in collaborative WFs (e.g., discretionary and
mandatory access controls, and object-oriented security). In [5], automatic construction of authorizations in a federated database is described, supporting flexible cooperation and data sharing. Task-based authorization in distributed systems is discussed in [11] as a flexible and adaptable access control paradigm.

Our approach to WF security consists in specifying rules for the control of agent assignment to WF tasks in order to enforce authorization constraints on WF execution. Rules (or triggers) are composed of events, conditions, and actions (ECA paradigm [13]). The event part of the rule specifies violations to the considered authorization constraint, the condition part determines if the occurred event actually corresponds to a violation situation to be managed, while the action part specifies the reaction of the system to the occurred violation within the normal flow. The approach has been developed in the framework of WIDE (Workflow on Intelligent Distributed database Environment), an EEC Esprit Project aimed at realizing a WF management system on top of an active database, using rules as the exception modeling paradigm. In fact, WF design in WIDE consists in modeling the “normal behavior” of a WF as well as the exceptions arising as “predictable deviations of the normal behavior” of the WF itself.

Enforcing authorization constraints by means of rules in WFs can be complex, because the normal behavior and the anomalous situations have to be identified, together with their corresponding corrective actions. Following recent proposals in the software engineering area [7], in WIDE pre-defined authorization patterns are introduced to reduce the design effort related to WF security exception modeling and handling. Authorization patterns are rule skeletons modeling typical authorization constraint exceptions for WFs in given domains. Rule skeletons constituting a given pattern can be reused and adapted to a new WF application by instantiating them into triggers to be executed on the WIDE active database.

The main contribution of our work is related to the use of a trigger-based mechanism for authorization constraint enforcement and of a pattern-based formalism and associated catalog for trigger design. In the literature on WF security, authorization constraints for WFs have been recently studied (see, for instance, [2]) from a specification point of view, by providing a logic-based language for constraint specification that facilitates also system analysis. In this paper, the focus is more on the implementation of WF authorization constraints based on the active rule paradigm. Moreover, we provide authorization patterns in a catalog as a means to reuse the knowledge on authorization constraints when designing a new WF, to avoid the definition of triggers from scratch each time an authorization constraint must be enforced in a WF.

The paper is organized as follows. First, the WIDE model and the use of rules to WF specification are presented. Then, issues related to authorization constraints and triggers in WFs are discussed. A basic set of authorization patterns is described to enforce the most frequent authorization constraints regarding secure task execution. Issues related to constraint enforcement based
on patterns and triggers are tackled. Finally, concluding remarks and future developments are given.

4.2 USING RULES FOR EXCEPTION HANDLING IN WIDE

In this section, we briefly review the basic concepts of the WIDE WF model; then, we concentrate on the structure of rules for treating exceptions, and on rules for specifying authorization constraints to cope with typical WF security requirements.

4.2.1 Overview of the WIDE model

In WIDE, a process is a WF schema defined as a collection of tasks which are the elementary work units. Tasks are organized into a flow structure defining the execution dependencies among tasks. The flow structure is specified by means of a restricted number of constructs allowing sequences, alternatives, and parallelism. Each WF schema has one start symbol and several stop symbols;
the start symbol has one successor and each stop symbol has one predecessor. A WF schema may include the definition of structured data, represented by WF variables. WF variables, besides enabling information exchange among tasks of the same case, are accessed by the WF Management System (WFMS) for checking possible constraints and for determining the tasks to be scheduled, when conditional executions are specified. For a detailed description of WIDE constructs the reader can refer to [4]. A WF case is an execution of a WF schema, i.e., an instance of the corresponding WF schema. Multiple cases of the same process may be active at the same time. A case is executed by scheduling tasks (as defined by the flow structure) and by assigning them for execution to a human or an automated agent. As a case is started, the first task (the successor of the start symbol) is activated. As a task connected to a stop symbol is completed, the case is also completed.

An example of WF specification using the WIDE model is shown in Fig. 4.1. This refers to a simple Document Preparation WF, composed of five tasks: Preparation, Evaluation, Rejection, Approval and signing, and Issuing. As the WF starts, the Preparation task is executed under the constraint evaluatingAgent (depicted by an arrow that represents a WIDE trigger), and its completion will cause the starting of the Evaluation task. After this task ends, either the rejection procedure (after which the flow terminates) or the approval procedure can be executed but not both (conditional fork with mutual exclusion), depending on the outcome of the Evaluation task. An approved and signed document will then be issued to the final destination (Issuing task) under the issuerAgent constraint, and the flow terminates.

Rules and constraints, which are the main issue of this paper, will be explained in detailed in the following sections.

### 4.2.2 Rules in WIDE

The WIDE approach to WF design consists in modeling separately the normal behavior of a WF and the predictable deviations, or exceptions, of the normal behavior of the WF. Rules (or triggers) in WIDE are employed to specify and manage exceptions. Rules conforming to the ECA paradigm: the event part defines when the rule is triggered; the condition part verifies if the triggered rule needs to react to the triggering event, while the action part specifies the operations required to manage the event.

Rules in WIDE are specified in the object-oriented language Chimera-Exc [6]. Chimera-Exc requires that the object-oriented schema upon which rules execute is defined. Chimera-Exc rules exploits three types of classes: WIDE classes, WF-specific classes, and event handling classes.

**WIDE classes** include description of the organization (roles, agents, and so on), and description of tasks and cases. These classes are WF-independent, and are predefined in the system; objects are created when new roles, agents, tasks or cases are created. For instance, Chimera-Exc rules may refer to attributes of running cases by accessing the attributes of the WIDE class case (e.g., `case(C)`, `agent(A)`, `C.responsible=A`, `A.name='John'`)
selects the case(s) whose responsible is John).
WF-specific classes store WF variables. Each case will be represented as an object within this class, created when the case is started.

Event handling classes store information carried by occurred events. For instance, the externalEvent class is referred to access the parameters of an occurred external event. A more detailed description of the WIDE specification language Chimera-Exc can be found in [6].

Event part. Each rule in Chimera-Exc can monitor multiple events, with a disjunctive semantics: the rule is triggered if any of its triggering events occurs. Events in Chimera-Exc, corresponding to the types of events previously listed, are specified as follows: i) data events, raising in correspondence of data manipulation primitives create, update, delete (e.g., constraint violation, task/case cancellation, unavailability of an agent); ii) external events, raised by external applications through the raise primitive (e.g., document arrival, telephone call, incoming e-mail); iii) WF events, enabling the monitoring of task/case starts and completions, expressed through the predefined events caseStart, caseEnd, taskStart(taskname), taskEnd(taskname); iv) temporal events, expressed as deadlines, time elapsed since a certain instant, or cyclic periods of time using the Chimera-Exc syntax.

Condition part. A condition is a predicate on the state of the WIDE database at the time of the condition evaluation which indicates whether the event must be managed. Rule condition includes class formulas (for declaring variables ranging over the current extent of a class, e.g., \( \text{tr}(C) \); \( C \) in this case ranges upon object identifiers of the \( \text{tr} \) class), type formulas (for introducing variables of a given type, e.g., integer(I)), and comparison formulas, which use binary comparison between expressions (e.g., \( \text{T.executor} = 'John' \)). Terms in the expressions are attribute terms (e.g., \( \text{C.destination} \)) or constants. The predicate occurred, followed by an event specification, binds a variable defined on a given class to object identifiers of that class which were affected by the event. For instance, in \( \text{agent}(A) \), occurred(create(agent),A), \( A \) is bound to an object of the agent class that has been created. If the result of a query is empty (i.e., no bindings are produced), then the condition is not satisfied and the action part is not executed. Otherwise, bindings resulting from the formula evaluation are passed to the action part in order to perform the reaction over the appropriate objects.

Action part. The action (or reaction) part can contain notifications to one or more agents or corrective actions on the current execution, expressed through the following Chimera-Exc primitives: i) Chimera data-manipulation primitives: these allow the creation of an object via the create primitive, the modification of the value of an object’s attribute via the modify primitive, or the deletion of an object via the delete primitive. For instance, delete(agent,A) removes all objects of class agent to which variable \( A \) is bound after the con-
tion evaluation. ii) Operation calls to the WF engine: these include primitives for notification of alarms to agents, starting a case or a task, and for assigning, re-assigning, rejecting, canceling, or rollbacking tasks or cases (e.g., notify(C.responsible, 'agent is unavailable'), reassignTask(T)).

In the next section, we will describe the use of ECA rules to detect and manage exceptions violating authorization constraints on the execution of tasks by agents.

4.3 AUTHORIZATION CONSTRAINTS AND RULES

Assignment of tasks to agents is performed on the basis of the organizational model of WIDE which is shown in Fig. 4.2 using the Entity-Relationship notation. Only authorized agents can execute tasks. Moreover, the concept of role is introduced, according to the concepts defined in [12], to represent the capability of an agent to execute a task. According to this model, authorizations for agents to play roles and for roles to execute tasks are defined in the system, represented by the play authorization and execute authorization relationships, respectively. These authorizations are defined to reflect specific organization policies and rules, and task assignment is performed in respect of the defined authorizations. This way, the “need-to-know” and the “task confinement” principles can be enforced in the system [3]. According to the need-to-know principle, agents are constrained to execute only the task(s) of their competence, and each agent can access only the information necessary for the completion of the task(s) he/she is authorized for.

According to the task confinement principle, agents are constrained to access
information objects only during the execution of a task. In fact, agents can access data objects only after their assignment to a given task, and any attempt to access data outside an authorized task is rejected. In addition to these two basic principles, authorization constraints can be imposed in the system, to enforce other security policies to flexibly regulate task assignment and execution, to cope with WF security requirements, in analogy with other approaches in the literature [1, 2, 9]. In particular, the following categories of constraints are considered:

- **Constraints on agents**, concerning the assignment of agents to roles for task execution. In particular, an authorization model for WF must support different types of constraints on agents. Examples of constraints on agents are: i) “Two different agents must execute two tasks $T_1$ and $T_2$” to enforce a separation of duties constraint and ii) “The same agent must execute two tasks $T_1$ and $T_2$” to keep the involved information confidential, realizing a “binding of duties” constraint.

- **Constraints on roles**, concerning the execution of tasks by roles. In particular, an authorization model for WF must support different constraints for task assignment and execution. An example of role constraint is the following: “At the least $K$ roles must be associated with the WF in order to start its execution”, to enforce a “cooperation” constraint on a WF.

Authorization constraints on agents and roles are not statically defined in WIDE, since a specification language for this purpose is not available in the environment. Rather, in analogy with the general specification paradigm adopted in this system, we enforce authorization constraints by means of active rules. Active rules are defined to detect the exceptions representing possible violations to authorization constraints, and to properly react to detected exceptions.

For example, with reference to the WF of Fig. 4.1, a “binding of duties” constraint on the task **Issuing** imposing that the agent executing this task must be the same agent who executed the task **Approval and sign**, is enforced by means of the Chimera-Exc trigger **issuerAgent** whose specification code is shown in Fig. 4.3. The event part specifies that the trigger is raised as the **Issuing** task starts. The condition determines which is the instance involved...
by declaring two variables $T_1$ and $T_2$, both ranging over the tasks classes. The occurrence of the predicate restricts $T_2$ to range over the task for which the event taskStart has been raised. Further, it requires that the case of this task be the same as the one of $T_1$ (i.e., Approval and signing) and checks if their executor agents are different. In this case, the action associated with the trigger consists in reassigning to $T_2$ the agent of $T_1$.

Rules in WIDE can be associated with a given WF at different levels (e.g., task level, schema level), or independently of any WF, therefore affecting multiple schemas. For security, we focus on the first kind of rules, which are the most common ones in modeling authorization constraints. Task level rules capture exceptions related to a single task. For instance, an exception defining a warning to the security officer as a reaction to a task activation by an unauthorized agent should be declared associated with the task itself, as in the “binding of duties” trigger example.

Rules should instead be declared at the schema level if they enforce a security constraint affecting the entire WF. For instance, the constraint that $K$ roles must be associated with the WF should be enforced by means of a rule declared at the schema level, since a whole case is affected.

### 4.4 AUTHORIZATION PATTERNS

Defining security rules for all possible authorization constraints to be enforced in a WF can become cumbersome, specially when complex flows are specified, with several involved tasks and agents. On the other hand, typical authorization constraints that need to be enforced in a WF can be prefigured, as in the examples illustrated in the previous section. Therefore, the idea consists in identifying the skeleton of a rule enforcing a given authorization constraint and in properly packaging such skeleton into an authorization pattern. Authorization patterns predefine typical (sets of) rules capturing the knowledge about the exceptions (i.e., violations) to given authorization constraints and the actions that can be performed to react to them. Authorization patterns can then be used as the starting point for designing authorization rules in each situation where an authorization constraint applies.

Authorization patterns are defined according to a reference model composed of the following elements:

- The **pattern specification**, which is a description of the authorization constraint enforced by the pattern, and is composed of several parts. Some parts (i.e., name, intent, and classification) allow the designer to identify and understand the goal of the pattern. The template part allows the description of the pattern itself. Finally, the keywords, related to, and guidelines parts are defined to allow the designer to locate the pattern in a repository, to understand the links with other WF patterns available in WIDE, and to provide suggestions about possible usages and personalization of the pattern for trigger definition. The template part contains the core specification of the pattern, in terms of events, conditions, and ac-
Pattern Specification

Name: bindingOfDuties

Intent: This pattern checks that the agent executing task T2 is the same as the agent executing a previous task T1 in a given WF, to enforce information confidentiality in the two tasks through the binding of duties constraint.

Classification: Authorization patterns | Agent authorization patterns

Template:

```
<taskname1>  
|            |        
|            |        
|<taskname2> |
```

define trigger bindingOfDuties

events taskStart(''<taskname2>'')

condition task(T2),
occurred(taskStart(''<taskName2>''),T2),
case(C), T2.caseId=C, task(T1), T1.caseId=C,
T1.name=''<taskName1>'',
[T2.activationNumber=T1.activationNumber,]
T1.executor!= T2.executor

actions 1. notify(C.responsible, "Authorization violation in task"+oIdToString(T2)
2. reassign(T1.executor,T2)
3. <action>
end

Keywords: security, separation of duties, violation, audit

Related to: integrity, roleExamination

Guidelines: The condition part dealing with activation numbers is needed only if the trigger is attached to a piece of WF with a loop inside.

Figure 4.4 The bindingOfDuties authorization pattern
tions. The template contains parametric fields to be filled in with specific values provided by the designer. Mandatory and optional parts can also be specified in a template. Events and conditions represent the main elements of the pattern, since they describe how to capture exceptions in a generic way. The action element provides in general a list of suggestions. In fact, reactions to exceptions are in general application-dependent, and the most suitable action must be selected depending on the specific situation.

- **Sample usages**, which are pattern instantiations on specific application examples. These examples show how an authorization pattern can be personalized in different contexts and applications by illustrating how variables/parameters appearing in the “template” field of the pattern specification can be supplied by the designer to produce a concrete authorization trigger.

- **Template interface**, which is a user-oriented interface simplifying the pattern instantiation process by providing default values for variables/parameters. Its purpose is to hide syntactic details of the Chimera-Exc language while compiling a pattern within a given application.

To capture the constraints on agents and roles previously discussed, the following basic set of authorization patterns is provided in the WIDE catalog:
- bindingOfDuties pattern
- separationOfDuties pattern
- numberOfRoles pattern

whose specifications are shown in Fig. 4.4, Fig. 4.5, and Fig. 4.6, respectively.

The separationOfDuties and the bindingOfDuties patterns specify generic rules to enforce agent constraints, to guarantee that the executing agents of the two tasks are different or are the same, respectively. The numberOfRoles pattern models the exceptions that can arise when violating the constraint on a minimum number $K$ of roles required to start the execution of a WF, which is an example of role constraint. It checks whether the number of roles appearing in the Roles attribute of the object task is not lower than a pre-defined minimum value stored in the minNumberOfRoles variable. When the pattern is used to generate a trigger, the value for this variable is set for each specific case to be checked.

In a pattern template, predefined parts, parameterized parts, and optional parts are defined. With reference to Fig. 4.4, we observe that the event and conditions clauses of the pattern template are parametric, that is, they are expressed in Chimera-Exc using generic parameters to become independent of any WF specific task. Generic parameters are specified within the “< >” symbols. Moreover, an optional part (shown between the symbols [ ] in the figure) is specified in the condition clause of the pattern, related to the activation number, which should be used only if a rule has to be defined for WFs with a loop, as suggested by the Guidelines part of the specification.
Pattern Specification

Name: separationOfDuties

Intent: This pattern checks that agent executing task T2 is different from the agent executing a previous task T1 in a given WF to enforce the separation of duties authorization constraint.

Classification: Authorization patterns | Agent authorization patterns

Template:

```
define trigger separationOfDuties

events taskStart('<taskname2>')

condition task(T2),
    occurred(taskStart('<taskName2>'),T2),
    case(C), T2.caseId=C, task(T1), T1.caseId=C
    T1.name='<taskName1>',
    T2.activationNumber=T1.activationNumber,
    T2.executor=T1.executor

actions 1. notify(C.responsible, "Authorization violation in task"+oIdToString(T2))
    2. delegateTask(T2)
    3. <action>

end
```

Keywords: security, separation of duties, violation, audit

Related to: integrity, roleExamination

Guidelines: The condition part dealing with activation numbers is needed only if the trigger is attached to a piece of WF with a loop inside.

Figure 4.5 The separationOfDuties authorization pattern
**Pattern Specification**

**Name:** numberOfRoles

**Intent:** This pattern counts the number of roles that are associated with a WF to allow WF starting only if a minimum number of roles is involved, to enforce a "cooperation" constraint.

**Classification:** Authorization patterns | Roles authorization patterns

**Template:**

```
define trigger numberOfRoles
    events caseStart
    condition case(C), occurred(caseStart,C),
    task(T), T.caseld=C, roles(R), T.roles=R,
    card(R) < <minNumberOfRoles>
    actions 1. cancelCase(C)
    2. notify(C.responsible, "Number of roles under the required minimum for case", caseId)
end
```

**Keywords:** security, number of roles, violation, audit

**Related to:** integrity, roleExamination

**Guidelines:** Besides case cancellation, also a notification message can be chosen for audit purposes.

---

**Figure 4.6** The numberOfRoles authorization pattern
define trigger evaluationAgent
  events taskStart('evaluation')
  condition task(T_2), occurred(taskStart('evaluation'), T_2), case(C), T_2.caseId=C, task(T_1), T_1.caseId=C, T_2.name='preparation', T_1.executor = T_2.executor
  actions delegateTask(T_2)
end

Figure 4.7 Example of pattern instantiation - the evaluationAgent trigger

Since a pattern provides a generalized description of the rule(s) necessary to enforce a given security constraint, it can be used for defining new rules of this kind in different WFs. The process by which a pattern is (re)used for generating new rules targeted to a specific WF is called pattern instantiation.

4.4.1 Pattern instantiation

Pattern instantiation is a mechanism for creating triggers to enforce a given authorization constraint on specific tasks and cases starting from an available pattern. Instantiation consists in binding all the parameterized parts of a pattern according to the desired usage of the trigger. The instantiation is based on a set of rules, that act on the event and condition parts of the pattern template, and on a set of constraints that must be verified to guarantee the correctness of the instantiation. A formal description of the instantiation rules and constraints is presented in [6]. Basically, such rules and constraints guarantee that all generic parameters that appear in the events and condition parts of a trigger in the pattern template are properly bound to corresponding Chimera-Exc expressions. An example of instantiation of the pattern bindingOfDuties of Fig. 4.4 is the trigger issuerAgent shown in Fig. 4.3. In this example, parameters <taskname1> and <taskname2> have been instantiated into Issuing and Approval and signing, respectively, which are the tasks to which the binding of duties constraint must be applied. Moreover, only the reassignment action has been selected for these two tasks of the Document Preparation WF.

As another example of instantiation, the trigger of Fig. 4.7 is obtained from the separationOfDuties pattern to implement the evaluationAgent trigger associated with the task Evaluation of the WF of Fig. 4.1, requiring that the agent who evaluates a document be different from the one who prepared it.

A tool called WERDE (Workflow Exceptions Reuse and Design Environment) has been developed to support the management of the catalog of patterns in WIDE. The tool provides functionalities to access and manage the pattern catalog, that is, to retrieve patterns considered useful for a given application, to store new patterns (possibly with associated sample usages), and to remove existing patterns. The extensibility of the catalog is an important aspect, to allow the insertion of new patterns related to new authorization constraints of
interest. In fact, there can be many different types of authorization constraints for which a suitable pattern has to be defined and made available in the catalog.

4.5 ENFORCING AUTHORIZATION CONSTRAINTS THROUGH TRIGGERS

In this section, we briefly discuss issues related to WF execution in presence of triggers enforcing authorization constraints. According to the constraint classification proposed in [2], three different types of constraints can be identified in a WF: i) static constraints, which can be evaluated before WF execution; ii) dynamic constraints, which can be evaluated only during WF execution, and iii) hybrid constraints, which can be partially evaluated without executing the WF. Constraints of type i) and iii) have the advantage of avoiding the execution of the WF if a violation occurs; they rely on the possibility of statically declaring constraints with some language. Since we adopt a pattern and trigger-based approach to constraint enforcement, we can not evaluate a constraint before the execution of a WF. However, we can associate the patterns related to static constraints with the caseStart event, which is the first event occurring in the system upon activation of a WF instance. Triggers are then evaluated as the caseStart event raises, avoiding the continuation of the whole case if a violation occurs. For example, the constraint that $K$ roles must be involved to start a WF is static, and, in fact, our numberOfRoles pattern is declared at the WF level, associated with the caseStart event, and avoids to start the execution of the case if this constraint is not met at the case starting.

Constraints on the agents executing two different tasks in sequence are dynamic, and the corresponding patterns are associated with the involved tasks and are evaluated as their execution starts.

As for hybrid constraints, they can be managed using two different patterns: a pattern associated with the caseStart event handling the exceptions related to the constraints that should be evaluated as soon as the case starts, and another pattern expressing the exceptions related to situations that can be checked only during case execution. An example of hybrid constraint is the one requiring that if at least $K$ roles are necessary to start the WF, and that tasks $T_1$ and $T_2$ must be executed by two different agents. This constraint can be enforced using the numberOfRoles pattern first, to check the first part of the constraint. Then, if this constraint is satisfied (i.e., no exceptions occurred), the separationOfDuties pattern can be used to define a trigger for controlling the second part of the constraint during flow execution.

4.6 CONCLUDING REMARKS

In this paper, we have presented an approach to design authorization constraints in WF systems based on the use of rules to be executed by the WF active database. Moreover, we have shown how these rules can be constructed for a WF by instantiating predefined rule skeletons called authorization patterns. Authorization patterns describe the knowledge about violations detection and
handling in a general way, to bounded to appropriate values in order to be included in a WF as triggers. A tool called WERDE has been implemented to support pattern definition and usage in WIDE; the tool operates on a pattern catalog, where authorization patterns are properly stored and classified. The tool exploits the pattern interface to guide the designer in completing a correct and executable WF, including the verification of authorization constraints which become WF triggers.

Currently, we are working on the extension of authorization patterns in the catalog, with, for example, patterns enforcing the separation and binding of duties constraints on roles, to extend the situations that can be controlled and make the system more flexible. We are also studying problems related to access control to information objects in external information systems interfaced by the WF. We are implementing a set of functionalities for schema analysis in presence of authorization triggers and other security functionalities for WFs, such as private and public key encryption algorithms for WF data confidentiality and for user certification. For this purpose, the tool will be linked to the functionalities of another tool which has been implemented for security management in the context of the CNR DEMOSTENE Project devoted to security of distributed systems in the Public Administration domain.

Acknowledgments

This work has been partially supported by the Italian Consortium for Informatics (CINI) and by the Italian National Research Council in the framework of the “DEMOSTENE Project”. We thank Dr. Eng. Fabio Casati who collaborated in revising security rules and Dr. Angelica Tescaro who collaborated to part of this work during her degree thesis.

References


III Privacy
Abstract: Privacy is concerned with the protection of personal information. Traditional security models (such as the Bell-LaPadula model) assume that users can be trusted and instead concentrate on the processes within the boundaries of the computer system. The InfoPriv model goes further by assuming that users (especially people) are not trustworthy. The information flow between the users should, therefore, be taken into account as well. The basic elements of InfoPriv are entities and the information flow between them. Information flow can either be positive (permitted) or negative (not permitted). It is shown how InfoPriv can be formalised by using graph theory. This formalisation includes the notion of information sanitisers (or trusted entities). InfoPriv is concluded with a discussion of its static and dynamic aspects. A Prolog prototype based on InfoPriv has been implemented and tested successfully on a variety of privacy policies.

5.1 INTRODUCTION

We live in an information age in which more and more personal information about the individual is stored in various database systems. These databases are maintained by health, financial and government-related institutions. It is of paramount importance to protect personal information from misuse. Refer to [5] for a discussion of privacy concerns and options for protecting information privacy on the National Information Infrastructure of the US.

Reports have been released about unauthorised disclosures of information in large information systems such as the NCIC (National Crime Information
Centre) [1] and those of the IRS (Internal Revenue Service) of the USA [4]. These reports further support the privacy concern.

The IRS has a number of security mechanisms in place to address the problem. These include the Electronic Audit Research Log (EARL) for monitoring and detecting unauthorised browsing of tax-related information. However, the General Accounting Office [4] concluded that EARL is limited in detecting the unauthorised viewing of tax-related information by IRS employees (called ‘browsing’).

It follows from the above that traditional security mechanisms are generally insufficient to ensure the privacy of information in large systems. We develop an information-flow model (named InfoPriv) here that may be used to model privacy policies. The basic building blocks of InfoPriv are entities and the information flow between them. Only entities are used in InfoPriv as opposed to the users and entities (objects) of traditional security models.

Entities are viewed as information containers and indirect information flow can occur between entities. We define negative information flow as a way of preventing indirect information flow. Entities and the potential information flow between them translate directly to directed graphs called ‘information can-flow graphs’. The entities form the vertices and the potential information flow forms the arcs.

In the next section we discuss the basic privacy principles. We define the Principle of Completeness and use it to unify the ideas of users and entities. The rest of the paper is roughly divided into three parts: Static Aspects of InfoPriv, a Formalisation and an introduction to the Dynamic Aspects of InfoPriv.

The Static Aspects of InfoPriv are concerned with entities and the potential information flow between the entities. Potential information flow is further divided into positive and negative information flow. InfoPriv will be formally presented in terms of graph theory next, followed by the Dynamic Aspects of InfoPriv. This paper will finally be concluded.

5.2 THE INFOPRIV MODEL

The purpose of this section is to describe a model called InfoPriv that is suitable for modelling privacy. InfoPriv and its underlying principles will be developed and justified throughout the rest of this paper by means of examples. We will start this section by defining the Principle of Completeness and describe how it may be used to relate security and privacy issues and how to model the privacy of a system.

5.2.1 Principles of privacy

The basic principle of privacy is that information should only be stored in a system for well-defined purposes [2, 3]. For instance, any information collected by the Internal Revenue Service (IRS) must only be related to and used for tax purposes. Any other use of IRS information is considered to be a violation of privacy.
We extend this principle by introducing the Principle of Completeness. The Principle of Completeness (PoC) states that the privacy of information can be better protected by having an improved understanding of the context of the information. The context of information is defined as the way in which the information will be used. An example of a context is the set of rules in an organisation that govern which employees should have access to payroll information. The PoC will now be illustrated by means of an example.

Consider two people: John Smith and Sarah Parker. Further assume that Sarah Parker works for the IRS and is the sister-in-law of John Smith. There is obviously a conflict-of-interest if Sarah has to process John’s tax-related information (employees of the IRS are not permitted to ‘browse’ their relatives’ tax information). However, specifying this requirement in a computer system that is based on a traditional security model is not so easy since traditional security has been designed around specific requirements. For instance, a very large number of security classes have to be used when applying the Bell-LaPadula model to this situation.

Using the PoC may solve the problem of Sarah having access to John’s tax information. A security system that adheres the PoC should permit constraints of the form ‘no person should have access to a relative’s tax information’ to be stored in addition to the normal security requirements (such as ‘Sarah has access to John’s tax information’).

An ideal implementation of the PoC is a security system that permits general privacy policies to be incrementally refined until it can be proven (from the privacy policy) that no unintended information flow can occur. Unintended information flow is information flow that is not prohibited by the privacy policy. However, the SSO (System Security Officer) may be under the (wrong) impression that the privacy policy does indeed prohibit the above-mentioned information flow. This is particularly applicable to complex policies.

Note that the specific structure and form of a privacy policy is outside the scope of this paper. We assume in this paper that a privacy policy is a set of general statements of the form “Employees of the IRS should not have access to their relative’s tax information” or “Jane Ullman is permitted to determine John Smith’s salary”. This paper is intended to describe how a privacy policy can be analysed once it is modelled in terms of InfoPriv.

The rest of this section is devoted to the development of the InfoPriv model by making use of the PoC.

5.3 STATIC ASPECTS OF INFOPRIV

The static aspects of a system are those parts or aspects that do not change over time. For example, a motorcar consists of a body, an engine and four wheels. The relationships between these parts stay the same even if it is impossible to determine where the car will travel over time. In the case of InfoPriv the static aspects consist of entities and the potential information flow between them. Note that we assume in this paper that the entities and the potential information flow between them (hence the privacy policy) stay fixed during the
lifetime of a system. This assumption is made due to space restrictions and the evolution of the privacy policy is the subject of [9].

Traditional security models make a distinction between (human) users and entities [12, 13, 14]. The main distinction between a user and an entity is that the user is viewed as external to the (computer) system while the entity is internal to the system. We mean by ‘internal to the system’ that the system contains the entity and has total control over it. Examples of entities ‘internal to the system’ are database tables, files and processes.

‘External to the system’ may be interpreted as meaning that the system only has sufficient information about a user as to permit interaction between it and the entities in the system. For instance, typical information that will be recorded for the user ‘Sarah Parker’ includes her password, privileges and other security related information. The system cannot force Sarah to keep certain information secret.

The first major difference between InfoPriv and traditional security models is that no distinction is made between users and entities in a computerised system. According to InfoPriv a computer system only consists of entities and the information flow between those entities. Refer to [8] for information about the lattice model that is also based on information flow. An entity can be viewed as an actor that interacts with various other actors throughout its entire lifetime. Examples of entities are “John Smith”, “Sarah Parker” and “Employee Table”. We will use quotation marks to indicate entities.

The two most important properties of an entity are that it should be uniquely identifiable and that it has memory (it is, therefore, an information container). “John Smith” knows various things such as where he lives, what his salary is and who his spouse is. He may even know things about other entities such as his wife’s name and salary and the names of his children.

An integral part of InfoPriv is the interaction between entities. This interaction can be modelled by using information flow. Consider the following entities: “John Smith” and “Sarah Parker”. Assume that “John Smith” may talk to “Sarah Parker”. We say that information may flow between John and Sarah. A convenient way to depict this (potential) flow of information is to make use of a directed graph where the vertices represent the entities and the arcs represent the potential information flow between the entities. This is shown in Figure 5.1.

---

Figure 5.1 Potential information flow between entities.
Note that we refer to a graph that depicts potential information flow (such as Figure 5.1) as an information can-flow graph or can-flow graph for short. There are two important advantages in using a can-flow graph to represent a privacy policy. First, graphs are a natural way of representing information, especially the potential information flow in a system as well as constraints on that flow. The second advantage is that a large base of graph algorithms can be used to analyse a can-flow graph for possible illegal information flow [7].

The potential flow of information in Figure 5.1 can be realised by either “John Smith” writing information to “Sarah Parker” or “Sarah Parker” reading information from “John Smith”. This symmetry between reading and writing forms the basis of the second major difference between InfoPriv and traditional security models. Reading and writing are traditionally viewed as separate operations, especially for discretionary security models [13]. This is no longer the case in InfoPriv and reading and writing are replaced by information flow altogether. The potential information flow of Figure 5.1 can, therefore, be seen as either “John Smith” writing to “Sarah Parker” or “Sarah Parker” reading from “John Smith”.

We justify the symmetry between reading and writing by considering again the distinction between users and entities of traditional security models. Users are viewed as outside a computer system and it is, therefore, not logical for entities inside the system to ‘write’ information to the users. It makes much more sense from that perspective to say that a user reads information from an entity instead of saying that the entity has written information to the user. A write operation is, therefore, an operation that is performed upon an entity inside a computer system. InfoPriv assumes no distinction between users and entities (both users and entities are viewed as entities modelled inside the world) so the difference between reading and writing is less important.

The last point of discussion for this section is the question of what constitutes an entity and what not. Consider “John Smith” again. It is quite intuitive to visualise “John Smith” as an entity since John can be uniquely identified. What about John’s salary? Does it qualify as an entity or should it be an attribute of John? The answer to this question is that it depends on the situation (also depending on the definition of an attribute that is covered in the formalisation of InfoPriv).

It is perfectly valid to view John’s salary as a number that should be stored as an attribute of John since it cannot be uniquely identified. However, it is also valid to store John’s salary in a separate entity provided that this entity can be uniquely identifiable. John’s salary would be stored in a separate entity when it is necessary to control the flow of information to and from John’s salary separately from John. Figure 5.2 depicts a separate entity (“John Smith’s Salary”) that stores John’s salary.

Figure 5.2 further shows that information can flow from the entity “John Smith’s Salary” to “John Smith” and via “John Smith” to “Sarah Parker”. This is the notion of indirect information flow that will be discussed in the following section.
5.3.1 Indirect and negative (potential) information flow

Indirect information flow occurs when an entity is capable of disclosing information that it received from other entities (refer to the example of John Smith's salary). This is in contrast to direct information flow where the possible information flow in a system is explicitly defined. An example of direct information flow would be "information can flow from John Smith's Salary to John Smith and from John Smith's Salary to Sarah Parker". Assume for the moment that only direct information flow can occur. The privacy policy of Figure 5.2 would then be represented as depicted in Figure 5.3.

Note that information may be prevented to flow from one entity (say "John Smith's Salary") to another entity (say "Sarah Parker") by removing the corresponding arc from the can-flow graph (see Figure 5.3).

Indirect information, however, is more complex and prevents one from determining the information flow between entities by a simple examination of the can-flow graph. A mechanism is needed by which information can be explicitly prevented from flowing between two specific entities. We will use the notion of negative information flow for this purpose.

Suppose that we want to prevent indirect information flow between "John Smith's Salary" and "Sarah Parker" in Figure 5.2. A convenient way of depicting this is by making use of special arcs (called negative arcs) that represent
illegal information flow. Figure 5.4 depicts positive and negative arcs. Thick lines and negative labels indicate negative arcs.

The positive arcs between “John Smith’s Salary” and “John Smith” and between “John Smith” and “Sarah Parker” indicate positive (potential) information flow. Positive (potential) information flow is permitted by the privacy policy and can be direct or indirect.

The negative arc between “John Smith’s Salary” and “Sarah Parker” indicates that information is not permitted to flow either directly or indirectly from “John Smith’s Salary” to “Sarah Parker”. Mechanisms will be discussed by which negative arcs can be implemented (refer to the Dynamic Aspects of InfoPriv).

We conclude this section by introducing the idea of trust and information sanitisers. Current security models are sometimes too strict concerning the flow of information between entities [11]. Assume that “John Smith” and “Sarah Parker” work in the same department in a company. They need to exchange information in order to do their tasks. However, according to information-flow models such as the Bell-LaPadula model [6] information is prohibited from flowing from “John Smith” to “Sarah Parker” in order to prevent John from disclosing his salary to Sarah. This is unrealistic since most organisations let colleagues communicate with each other with the clear understanding that payroll information should not be discussed. People are, therefore, trusted not to disclose certain information.

One way of permitting John to talk to Sarah without disclosing his salary is to make John an information sanitiser. This is done by dividing him into logical sub-entities with each sub-entity corresponding to information that John discloses to specific entities. Figure 5.5 shows “John Smith” as an information sanitiser [11]. Note that this division of “John Smith” into sub-entities is according to the PoC. The more accurately we can model the way in which John Smith compartmentalises and discloses information, the higher is the level of privacy that the security system may provide.
Figure 5.5 shows that "John Smith" consists of two logical entities: "John Smith1" and "John Smith2". "John Smith1" is the part of "John Smith" that contains confidential information while "John Smith2" contains public information. "John Smith" is permitted to select information from "John Smith1" and 'place' it in "John Smith2" thereby publishing it (no information flow occurs from our point of view since we cannot tell what John is thinking).

**Definition 1:** An information sanitiser is an entity that is permitted to selectively disclose confidential information to other entities. Such an entity is modelled as consisting of logical sub-entities with each sub-entity containing information that is intended for disclosure to specific entities. Note that an information sanitiser models trust.

So far we have given an informal overview of InfoPriv and will proceed to formalise it in the following sections.

### 5.4 FORMALISATION

We will now develop InfoPriv formally by using graph theory. A can-flow graph can be defined as the tuple $G = (E, \rightarrow)$ where $E$ is the set of entities and $\rightarrow$ is the set of arcs (or potential information flow between the entities).

Figure 5.4 depicts a can-flow graph with $E = \{"John Smith's Salary", "John Smith", "Sarah Parker"\}$ and $\rightarrow = \{<"John Smith's Salary", "John Smith", +1>, <"John Smith", "Sarah Parker", +1>, <"John Smith's Salary", "Sarah Parker", -1>\}$. Note that '+1' indicates a positive arc while '-1' indicates a negative arc. We will use angled brackets '<', '>' throughout the rest of this paper to indicate tuples.

Each entity is associated with a set of values (i.e. John Smith is 30 years old and lives in 10 Bourbon Street). We define the set $V$ of all the entity-values, $A$ the set of all attribute names and the function $\phi$ to map entities and their attributes to their corresponding values.

**Definition 2:** The set $V$ consists of all the values. For example, $V = \{10000, "10 Bourbon Street", ...\}$.

**Definition 3:** The set $A$ consists of all the attribute names. For example, $A = \{"value", "salary", "designation", "age", "address", ...\}$.
Definition 4: The function $\phi: E \times A \rightarrow V$ maps the entities and their attributes to corresponding values. If $\phi$ is represented by the set of triples $\{\langle \text{"John Smith"}, \text{"Age"}, 30 \rangle, \langle \text{"John Smith"}, \text{"Address"}, \text{"10 Bourbon Street"} \rangle \}$ then we can conclude that John Smith’s Age is 30 (as stored in the attribute “Age” of entity “John Smith”) and his address is “10 Bourbon Street”.

We stated in the previous section that indirect information flow can occur between entities. Indirect information flow will now be formally defined in terms of the full-reachability function $\psi$.

Definition 5: The full-reachability function $\psi: E \rightarrow \wp (E)$ maps each entity to the set of entities that can be reached from it in the can-flow graph. Formally, for an entity $e_i$

$$\psi'(e_i) = \{e_j \mid (e_j \in E) \land (\langle e_i, e_j, +1 \rangle \in \rightarrow)\}$$

$$\psi(e_i) = \bigcup_{e_j \in \psi'(e_i)} \psi(e_j)$$

For instance, the full-reachability set of “John Smith’s Salary” is \{“John Smith”, “Sarah Parker”\} since its value can flow to “John Smith” and to “Sarah Parker” (via “John Smith”). Note that $\psi$ ignores negative arcs.

Information is permitted to flow from an entity to any entity in its full-reachability set provided that a negative arc does not prevent this flow. For instance, if $\psi$ (“John Smith’s Salary”) = \{“John Smith”, “Sarah Parker”\} then information can flow from “John Smith’s Salary” to “John Smith” but not to “Sarah Parker” since the arc $\langle \text{"John Smith’s Salary"}, \text{"Sarah Parker"}, -1 \rangle \in \rightarrow$.

To conclude, the static aspects of InfoPriv may be used for a static analysis of a privacy policy (hence the can-flow graph). Potential unauthorised information flow may be identified and the can-flow graph can be modified to prevent such information flow from occurring during the lifetime of the system. Refer to [10] for a description of an InfoPriv Workbench that may be used for this purpose.

The next section will contain a description of the dynamic aspects of InfoPriv.

5.5 Dynamic Aspects

The previous section formalised the Static Aspects of InfoPriv. A can-flow graph is really a representation of how information can flow between entities (the static aspects). We correspondingly refer to the can-flow graph as a can-flow relation [14]. Consider the can-flow graph of Figure 5.2 again. The only information contained in Figure 5.2 is that information may flow from “John Smith’s Salary” to “John Smith” and information may further flow from “John Smith” to “Sarah Parker”.

We cannot deduce from a can-flow graph what information flow will actually occur and in what order. For instance, information may first flow from “John Smith” to “Sarah Parker” followed by information flow from “John Smith’s Salary” to “John Smith”. This will not lead to the disclosure of John’s salary to Sarah Parker (assuming that “John Smith” does not know his salary at first).
However, by analysing the static aspects we can only make the pessimistic assumption that Sarah Parker will eventually be able to determine John’s salary. This unauthorised information flow can be prevented by either preventing the direct information flow from “John Smith’s Salary” to “John Smith” or by preventing the direct information flow from “John Smith” to “Sarah Parker”. John obviously has to know his salary, so the direct information flow from “John Smith” to “Sarah Parker” has to be prevented (John and Sarah may be placed in different departments). Note that numerous graph traversal algorithms do exist by which paths through a directed graph can be found [7]. We will not discuss these and possible conflict resolution algorithms further. Refer to [10] for a discussion of conflicts and conflict resolution of a can-flow graph.

The other alternative is to determine the actual information flow during run-time. This constitutes the dynamic aspects of InfoPriv. A few observations about the ‘world’ are in order before we describe the dynamic aspects of InfoPriv. We assume that the world consists of entities where each entity has its own viewpoint of the world. These viewpoints change as a result of interactions between entities. For instance, the personnel manager at John Smith’s company may have decided to change John’s salary without notifying John. The viewpoint of “John Smith’s Salary” is different from John’s since he only remembers an earlier value of his salary. However, the personnel manager can inform John of his new salary or John can see it on his salary statement at the end of the month, thereby changing his viewpoint of the world.

Dynamic aspects may be conveniently modelled by extending the definition of the $\phi$ function. Each entity has an associated $\phi$ function instead of one global $\phi$ function. For instance, the $\phi$ function for “John Smith” will be denoted by $\phi_{\text{John Smith}}$ where $\phi_{\text{John Smith}}$ represents John Smith’s view of the world. The change of $\phi_{\text{John Smith}}$ over time models the change of John Smith’s viewpoint over time (hence John’s dynamic aspects).

**Definition 6:** The function $\phi_{\text{Entity}}$ represents Entity’s viewpoint of the world. The change of $\phi_{\text{Entity}}$ represents the dynamic aspects of “Entity”.

We will now define how information flow between two entities will influence their viewpoints. Assume that $\phi_{\text{John Smith's Salary}} = \{<\text{"John Smith's Salary"}, \text{"value"}, 20000>\}$, $\phi_{\text{John Smith}} = \{<\text{"John Smith's Salary"}, \text{"value"}, 10000>\}$ and information flows from “John Smith’s Salary” to “John Smith”. “John Smith” will now know the new value of his salary. After the information flow $\phi_{\text{John Smith}}$ will be modified to $\phi_{\text{John Smith}} = \{<\text{"John Smith’s Salary"}, \text{"value"}, 20000>\}$.

Note that entity attributes can be changed by using information flow. Assume the privacy policy of Figure 5.6. The value of “John Smith’s Salary” can be changed by the entity “Personnel Manager” since information can flow from it to “John Smith’s Salary”. Assume that $\phi_{\text{Personnel Manager}} = \{<\text{"John Smith’s Salary"}, \text{"value"}, 10000>\}$. “Personnel Manager” can now change his ‘memory’ to $\phi_{\text{Personnel Manager}} = \{<\text{"John Smith’s Salary"}, \text{"value"}, 20000>\}$. Information flow from “Personnel Manager” to “John Smith’s Salary” will update John Smith’s salary to 20000.
Negative information flow can be implemented with the $\phi$ functions of the various entities. For instance, assume that we want to prevent the value of "John Smith’s Salary" to reach "Sarah Parker". We can permit information to flow from "John Smith" to "Sarah Parker" as long as information has not flowed from "John Smith’s Salary" to "John Smith" ($\phi_{John\ Smith}$ should, therefore, not contain a triple of the form <"John Smith’s Salary", $a$, $v">$ with $a \in A$, $v \in V$.

Note that it may still not be realistic to prohibit John Smith from talking to Sarah Parker after he has determined his salary. Information sanitisers can be used to model John Smith more realistically in terms of sub-entities (refer to the indirect information flow of InfoPriv).

The dynamic aspects (as discussed in this section) are summarised in the algorithm of Figure 5.7. Lines 1 to 5 test whether any of the entities of which information has reached entity $t$ may not reach entity $u$. If this is not the case the $\phi$ function of entity $u$ is updated according to $\phi_t$ (refer to the example of "John Smith’s Salary" and the "Personnel Manager").

Note that negative arcs are not transitive. Assume that a negative arc indicates that information may not flow from entity $A$ to entity $B$ and a second arc indicates that information may not flow from entity $B$ to entity $C$. This does not mean that information may not flow from entity $A$ to entity $B$ (there may be a positive arc from entity $A$ to entity $C$). Only a direct negative arc from entity $A$ to entity $C$ will prevent information to flow from entity $A$ to entity $C$.

A prototype of the dynamic aspects has been implemented in Prolog. This prototype has been quite successful in representing and analysing various privacy policies such as the IRS scenario mentioned previously. It is general enough to permit the specification of any privacy requirement that can be described in terms of Prolog predicates. We are in the process of testing the prototype for more general privacy policies such as workflow. The details of this prototype are outside the scope of this paper.
DoFlow \((\rightarrow_{pos}, \rightarrow_{neg}, E, t, u, \text{success})\)

Input: \(\rightarrow_{pos}\) is the set of positive arcs, \(\rightarrow_{neg}\) the set of negative arcs, \(E\) the set of entities and \(t, u \in E\) (information flow from \(t\) to \(u\) is attempted)

Output: success is a flag that indicates whether the flow was successful or not

For all \(\langle e_i, a_i, v_i \rangle \in \phi_t\)
where \(e_i \in E, a_i \in A\) and \(v_i \in V\) do
if \(\langle e_i, u \rangle \in \rightarrow_{neg}\) then
  success \(-\) False
  exit
update \(\phi_u\) according to \(\phi_t\)
success \(-\) True

Figure 5.7 Algorithm to implement the dynamic aspects of InfoPriv.

5.6 CONCLUSIONS

It is crucial to ensure the privacy of personal information in order to prevent the misuse of such information. Suitable privacy models, therefore, have to be developed by which the privacy policy of a system can be described and enforced. This paper has presented a model (called InfoPriv) for privacy that is based on the Principle of Completeness (PoC). This principle dictates that the level of privacy that can be guaranteed by a system depends on how thoroughly the context of the information can be modelled.

One of the results of the PoC is the unification of users and entities. Instead of making a distinction between the users of the system and the entities we chose to model only entities. Traditional data entities map directly to the entities as defined by InfoPriv while users map to entities with weaker guarantees (we cannot determine the precise information flow between people for instance). The entities (viewed as information containers in InfoPriv) together with the flow of information between them form the building blocks of InfoPriv.

We further noted that indirect information flow can occur between entities. It is necessary to control the indirect information flow between entities and this is done by means of negative information flow. Negative information flow was defined as a way to prevent the actual flow of information between two entities.

An InfoPriv policy maps well to a graph with the entities forming the vertices while the potential information flow between the entities corresponds to directed arcs. We introduced the idea of an information sanitiser. An information sanitiser is an entity that is trusted to selectively disclose confidential information to other entities. InfoPriv was further formalised in terms of graph theory.
We proposed two ways of implementing negative information flow. The first involves a static analysis of the can-flow graph for conflicting information flows (positive information flow that is prohibited by negative information flow). The second method involves a dynamic analysis of the actual information flow during run-time.

We lastly discussed the dynamic aspects of InfoPriv. The definition of an entity was extended by adding a local view of the world to each entity instead of having a global or absolute world. The dynamic aspects, therefore, correspond to a change over the local views of all the entities. We showed how an entity can change the attribute of another entity by first altering its own 'memory' and initiating an information flow to the destination entity after that.

It was, therefore, attempted to illustrate in this paper how a privacy model can be derived by considering relevant privacy issues. The privacy issues, a full mathematical model of privacy and the implementation of the model warrant further research. Note that a Prolog prototype based on InfoPriv has been implemented and tested successfully on a variety of privacy policies.

Acknowledgments

We would like to thank Professor Sushil Jajodia for his generous contributions to this paper.

References


IV  Policy Modeling
6 SECURITY POLICIES IN REPLICATED AND AUTONOMOUS DATABASES

Ehud Gudes and Martin S. Olivier

Abstract: Autonomous object databases are becoming important in the Internet world of today and involve integration of several local databases. Such databases support local access for transactions and queries and local control over authorization of classes and objects. At the same time, these database objects are often replicated in various sites and are available for access by global queries and transactions. Such global access, which may involve a global query optimizer, is required to handle conflicts between the local authorizations of replicated objects, but give consistent results regardless of site dependent optimizations.

The paper uses previous models for object-based authorization, and extends them with policies to handle conflicts between local and global authorizations. It also discusses object migration and security administration. The problem of providing autonomy in a consistent way is discussed extensively.

6.1 INTRODUCTION

Autonomy is an important concept in today's fragmented but connected world. Organizations that support databases on the Web, often require autonomy in local access of their local data and in controlling access to it; at the same time they want to provide access to their data objects (or to copies of their objects) to a community of external users who may access them through a global system or a distributed query interface. In this environment conflicts may occur when different sites or security administrators place different access
restrictions on such replicated data objects. The policies and algorithms to handle such conflicts in a consistent way, is the topic of this paper.

In principle, most of the ideas presented in this paper, hold in both relational and object-oriented database environments. However, since object databases seem to be more dynamic in nature and more local in character, we feel that the object database model is a more appropriate context to investigate this question. In this paper we will rely on some of our previous work on authorization in object-oriented (OO) databases such as [1]. In that paper the question of query modification in OO databases in the presence of authorization rules was investigated, and, in particular, the relationship between inheritance and authorization was discussed. An access evaluation algorithm was presented, and an administration model and policies were discussed. In particular, the inheritance policies of Negative vs. Positive, and Implicit vs. Explicit were handled. These ideas were generalized for Methods in [2]. The main idea we need from [1] is the policy to evaluate the access rights on an object, giving several authorization rules on objects (classes) above or below its inheritance hierarchy. In most of the paper we discuss hierarchies along one dimension — the inheritance sub-class/super-class hierarchy. Other hierarchies are discussed briefly at the end of the paper.

When an object database is distributed, it is often the case that its schema is also distributed and therefore its authorization information is distributed as well. Furthermore, it is common that different sites may have their own security rules and their local security administrator (SA) to define them. We know of three models that address the issue of security policies in autonomous databases. Argos [4] assumes a global policy which must be consistent with the local DBMS policies and is enforced as such. (The Argos policy is as follows: if global denies then deny, else if global permits then select one local DBMS; if permitted by the LDBMS fine, else try to grant it; if grant successful fine, else deny.) DOK [11] accepts the local policies and tries to integrate them into a global one. DOK, however, does not consider all the different cases as they are described in this paper. SPO [8] assumes that the owning site has the final say in how its data is accessed — the local policy is therefore paramount, with a relatively small federal policy adhered to by all sites. The current work differs from the others because it assumes a shared global (or federal) schema (even though the schema may only exist in partial forms at the various sites and be integrated by a query optimizer as and when required) and, because of local autonomy, conflicting access rules may be specified (or implied) for the same class at different sites. Note specifically that we assume that there may be multiple administrators where each is issuing her own authorization rules on the local copy of the database. As an example, consider a number of libraries with on-line reference material in their databases. These libraries have established a federated database to increase the available information to their members. Each library maintains an autonomous site. When a document is to be retrieved from this federated database, the optimizer may, in general, use criteria such as proximity, link availability and link speed to select the optimal sources from
which to retrieve requested information. Access restrictions may, however, influence the selection of sources. The licence agreement that a particular library has with the supplier of a particular document, may restrict use of that document to members of the specific library. If a global user can get access to the document at another library the access should be permitted to the document at that library — access is only restricted at this library. Similarly, a library may find that the requirements of a particular user is excessively high and prohibit that user from accessing the information at its site; again accesses at other sites are, in principle, still permitted.

It is also possible to envisage another scenario, where library members pay a subscription fee to access specific information. In this case, when a member has not subscribed to some document, access to that document should be prevented at all sites. The paper will address these alternative cases systematically.

Also note that it is, in principle, possible to replicate information from one library at other libraries. Similarly, information may be moved from one site to another site. This may be done for efficiency or various other reasons.

Suppose now that a global query enters the system and the optimizer must decide where to get the data from. We want the retrieved results to be consistent regardless of the optimizer decisions! This means that we have to integrate the various authorization states of the various local databases before the optimization. This integration is not trivial and raises several policy issues such as: 1) Conflicts between implicit vs. explicit, and negative vs. positive rules in the different sites; and 2) how the decisions will be influenced by whether we have equal authority administrators vs. a ‘master administrator’ and ‘local administrators’ One possible solution is to provide several types of authorization rules, such as authorization rules only allowing or denying access to local copies of an object compared to authorization rules allowing or denying access to all copies of an object.

The existence of various authorization rules require consistent policies to handle conflicts between them. These policies are the main subject of this paper, and they are discussed in general in the next section, and in detail in a later section. Once the policy decisions are made, the algorithms to integrate two (or more) sets of authorization rules are straightforward, and they are discussed briefly in the section on algorithms. The problem of copying and migrating objects in this environment is subsequently discussed. Finally we briefly discuss a few other issues, including other hierarchies (granularities), security administration and information flow.

6.2 POLICY ISSUES

6.2.1 The model

First we present the model and its assumptions. The model we use is depicted in figure 6.1. We have an object-oriented database distributed over several sites. Each site has its own schema of classes, attributes and authorization rules. Each site has its own security administrator (SA). Sites maintain local
autonomy in that they expect all local queries and transactions to obey the local authorization rules. Schema parts and objects may be replicated in several sites. Global queries or transactions first enter a global optimizer which accumulates all authorization information and physical information and decides on the actual sites from where data will be fetched. Evaluation of authorization in a single site is done by the algorithm presented in [1] and its result is the authorization tree (AT) — the set of attributes and classes to which access is allowed.

Next we state our assumptions on the database and the authorization rules. We assume an inheritance-based object-oriented database, with the policies specified in [1]. In that paper we mainly discussed the class-generalization hierarchy, and assumed mostly that all rules are defined on classes and sub-classes. Similar rules may be used for the granularity hierarchy (i.e. class/attribute or class/object). To simplify matters they will be discussed separately later. The same will hold for rules which define predicates, i.e granules defined by predicates (see [6]). Also, in most of the paper we assume that authorization rules are defined on classes and attributes and not on individual objects (they are discussed later), but we often will use the concepts of classes and objects interchangeably.

Assigning access rights to a class has multiple possible interpretations [7]. In this paper we assume that a class is merely labelled to control access to instances of the class. In particular do we assume that access rights may be added or revoked at any level of the class hierarchy and are inherited from a level
where they are defined down to — but not including — the level where they are modified. (Note that this interpretation presents another valid alternative to those discussed in [7] because axiom 4 of [7] does not apply here.)

We assume that in each site (or set of sites) there is a hierarchy of object-classes with its own security administrator who can specify authorization rules. The schemas in two different sites may not be the same, although we assume for simplicity that one schema extends the other (up or down), but if two classes appear in both schemas, then the path between them also appears in both schemas.

As stated above, we assume complete autonomy on behalf of the administrators of the autonomous objects (actually, autonomous object hierarchy), The security administrators (SAs) act independently and specify various authorization rules on the objects. The problem is the combination of these rules, since we assume sharing of some (large) parts of the schema.

6.2.2 The policies

Now we state several of the principles we want this autonomous system to obey:

P1) When a local security administrator makes a decision about her local data, such as denying local access to person P, she should not be concerned that person P can get that access from some other site. On the other-hand, the run-time should be consistent: if the optimizer decides to access the data from her site, it should not get that access! Therefore, we should always guide the optimizer where to get the data from (i.e pass the positive access site information).

P2) When a security administrator has the right to define authorization rules which may apply to more than one site, we expect the system to follow through and apply them correctly.

P3) One never gets more access by accessing only a local site than by issuing a global query! This means that by issuing a global query we can only get more data permitted — not less than accessing the local site only (e.g by a local query or application). This will considerably influence the policy decisions below. We call this last policy the principle of maximum access.

The application of the policies above depends on the type of organization one has. We distinguish between two different types of organizations:

---

1 As a corollary of P1, if a global access rule restricts local access, it may be inconvenient for the local system to consult other sites for local access. Therefore, for performance and reliability reasons such a restrictive rule should be propagated to all local sites. See also the section on Administration issues.

2 One positive result of this principle is that when a site fails, one never gets more access than without that failure.
1. **EQUAL (EQ)** — all sites are equal, each SA has the same power.

2. **Master-Slave (MS)** — The master’s SA has more power than the slaves’ SAs. We do not precisely define the exact ‘division’ of power; some possibilities are discussed below.

All access rules may be

1. Intended globally; or
2. A distinction may be made between global and local rules:
   
   (a) **Local** — rules (positive or negative) which apply to the local site only, called below PL or NL (positive or negative local).
   
   (b) **Global** — Rules which apply in all relevant sites, called below PG or NG (positive or negative global).

Usually, all four combinations of organizations and rules types are possible, although some restrictions may apply in some cases. Next we detail how the access evaluation is done in each of the above cases, while enforcing the general policies outlined above.

### 6.3 ACCESS RULES — DETAILED POLICIES

Suppose that some subject \( s \) wants to access some object of class \( X \). Let \( A(X) \) be the set of access values that have been specified for \( s \) in the concerned mode \( m \) (such as reading, writing, etc). Denote each such access rule by \( a^d_i \), where \( i \) indicates the site where the rule has been specified and \( d \) indicates the distance in the lattice (or class hierarchy) from \( X \) to the (higher) class where the rule has been specified. If the rule has been explicitly specified for \( X \), then \( d = 0 \), etc.

Each \( a \) is \( P \) or \( N \), indicating a positive rule (allowing access) or a negative rule (denying access), respectively. If local policies are supported, \( a \) may be \( PG \) or \( PL \) — for a global or local positive access specification to \( X \) — or \( NG \) or \( NL \) — for a negative global or local access specification to \( X \), respectively.

It is assumed that every access rule is propagated from the class where it is specified, to all classes lower in the lattice (with \( d \) incremented for every level that the rule is propagated down). At any given class a variety of rules therefore exist that need to be combined to determine the effective rule to be used. The following two principles are usually used when combining such rules (same as in [1]):

1. The shorter the distance that a rule has been propagated, the more specific it is considered to be; rules propagated over a shorter distance therefore take precedence over rules that have been propagated over a longer distance; and

2. If rules propagated over an equal distance conflict, access is denied.

Let \( [a^d_i, a^d_j] \) be the access value (\( a^d_i \) or \( a^d_j \)) propagated the shorter distance:

\[
[a^d_i, a^d_j] = \begin{cases} 
  a^d_i & \text{if } d < e \\
  a^d_j & \text{if } d > e 
\end{cases}
\]
if $d > e$ then $a^e_j$
if $d = e$ then
  if $i \leq j$ then $a^d_i$
  else $a^e_j$

When $d = e$ either of the two values may in fact be used — they have been propagated an equal distance. For formal manipulation, we find it useful to select a unique value in this case — hence the last three lines of the definition. Note that, where $d = e$ one authorization may in principle be negative while another may be positive; however this situation will not occur where we use this operation below.

To express these two principles formally, the function $\text{min}$ is defined to determine the effective access rule when considering two explicit or propagated access rules:

$$
\text{min}(a^d_i, a^e_j) = \begin{cases} 
  [a^d_i, a^e_j] & \text{if } d \neq e \\
  [a^d_i, a^e_j] & \text{if } d = e \\
  a^d_i & \text{if } a^e_j = N \\
  a^e_j & \text{if } a^d_i = N \\
  a^d_i & \text{else} \\
  a^e_j & \text{else} 
\end{cases}
$$

The conditional expression if $a^d_i = N$ (without superscripts or subscripts following $N$) means if $a^d_i$ is equal to any $N$ (negative) value. Whenever superscripts or subscripts are omitted in such a conditional expression, it should be interpreted in this way.

If local policies are supported, the shortest distance principle does not necessarily apply: If a site denies subject $s$ local access to some subtree for which global access has been granted higher in the lattice, the local denial should not override the global positive authorization — as long as there is some site that contains the subtree where a negative rule has not been specified. The same occurs when two local rules are combined, if none is more specific than the other, then the positive rule wins, unless both rules are in the same site. To determine the effective access rule, where the two constituent rules are both local rules:

$$
\text{local}(a^d_i, a^e_j) = \begin{cases} 
  (i = j) \lor (a^d_i = NL \land a^e_j = NL) & \lor (a^d_i = PL \land a^e_j = PL) \text{ then } [a^d_i, a^e_j] \\
  a^d_i & \text{else} \\
  a^e_j & \text{else} 
\end{cases}
$$

The combination of a local and a global policy is problematic in one sense: The global policy will override the local policy, but if all local policies deny access, global access has effectively been denied. Consider the following, where we assume that the first argument is global and the second one local:

$$
\text{globloc}(a^d_i, a^e_j) = \begin{cases} 
  a^d_i & \text{if } a^d_i = NG 
\end{cases}
$$
else if \( i = j \land a_j^x = NL \) then \( PG_{\infty}^d \)  
else \( a_i^d \)

The logic behind this definition is that a global rule always takes precedence over local rules. The only 'exception' occurs if the site that granted a positive global authorization denies local access to the data: the positive global authorization still exists, but no site where it may be accessed by user \( s \) may remain. For that reason, in such a case the combination results in a positive rule where the site is specified as \( \infty \), meaning that no positive site is known (at this stage).

Let \( +_n(a_i^d, a_j^x) \) combine two access rules \( a_i^d \) and \( a_j^x \). The definition of \( +_n \) is considered below for the various cases. Consider access rule combination for read access in the following four cases.

1. **EQUAL, no local policies** If the two rules are not of the same level, the more specific rule takes precedence, otherwise the negative rule takes precedence. Therefore, let \( +_1(a_i^d, a_j^x) = \min(a_i^d, a_j^x) \). The logic behind this choice to combine rules has been discussed when \( \min \) was defined. That is, when all rules are global, the simple 'min' policy which is commonly used in a centralized database is used.

2. **EQUAL, with local policies** If both rules are global or at the same site then this case should be treated similar to the previous case. Otherwise, if there is a global negative rule, it takes precedence. If no such negative rule exists and a positive rule does exist, it takes precedence. These are the consequences of precedence of global rules over local rules and the principle of maximum access.

\[
+_2(a_i^d, a_j^x) = \begin{cases} 
\min(a_i^d, a_j^x) & \text{if } i = j \lor (a_i^d = \text{global} \land a_j^x = \text{global}) \\
\text{local}(a_i^d, a_j^x) & \text{else if } a_i^d = \text{local} \land a_j^x = \text{local} \\
\text{globloc}(a_i^d, a_j^x) & \text{else if } a_i^d = \text{global} \\
\text{globloc}(a_j^x, a_i^d) & \text{else}
\end{cases}
\]

3. **Master-slave, without local policies** This case should essentially be handled similar to the case for equals without local policies.

\[
+_3(a_i^d, a_j^x) = +_1(a_i^d, a_j^x)
\]

The above case means that there is a distinction between a master and a slave, but all rules are in a sense global. A common 'division' of power will be that only Master SAs can define authorization rules. This should not be seen as self contradicting, since even if we restrict the security administrators to be on the Master site only, there is still the case of two Master sites which is handled by the above rule.
4. Master-slave, with local policies  Again the Master-slave has an impact on the administration of security. For example, we may want to restrict that global rules can only be issued by master administrators, while slave SAs can only issue local rules. (See also the discussion of administration in below). However, we may want to restrict that only one site can issue global rules. This case is therefore handled similar to the case for equals with local policies, but Master-slave rules are treated similar to Global/local rules:

\[ +_4(a_i^d, a_j^e) = \begin{cases} \text{error} & \text{if } i \neq j \land a_i^d = \text{global} \land a_j^e = \text{global} \\ \text{else } a_i^d + 2 a_j^e & \end{cases} \]

Note that if both a master and a slave issue local rules, the Master has no advantage over the slave when combining these rules.

**Theorem 1** \( +_n \) is commutative and associative for each \( n \).

**Proof** A detailed proof is outside the scope of this paper. Intuitively, one can argue that in each of the above operators a single selection is done from a pair \((X,Y)\), where the selection is dependent only on the values of \(X\) and \(Y\) and not on their order. Thus commutativity is trivial. Associativity is assured by the transitivity of the 'selection' process. If \(X\) was selected over \(Y\), and \(Y\) was selected over \(Z\), then \(X\) will be selected over \(Z\), whether it is first combined with \(Y\), or whether \(Y\) is first combined with \(Z\).

Since \( +_n \) is commutative and associative, it is possible to define operators to combine an arbitrary number of rules. Let \( \sum_n \) do this combination for each of the four cases. Where \( n \) is not significant, it will be omitted.

Because of the operator commutativity and associativity, the order in which the various access rules will be combined within the access evaluation algorithm, is immaterial. Likewise, if rules are at different levels of the hierarchy, the order of their combination is also immaterial. This is specified formally in the next theorem.

Denote the effective access rule that applies at a class \(X\) for subject \(s\) by \(e(X)\), then

\[ e(X) = \sum_{a_i^d \in A(X)} a_i^d \]

Denote the fact that \(Y\) is immediately above \(X\) in the lattice by \(Y \succ X\). Let \(D(X)\) be the set of all the access rules that have been directly specified for \(X\). The significance of the next theorem is that the effective access rights to some class \(X\) may be computed from its immediate ancestors and its direct access specifications.

**Theorem 2**

\[ e(X) = \sum_{Y \succ X} e(Y) + \sum_{a_i^d \in D(X)} a_i^d \]
Proof  Let \( \succ^* \) be the transitive closure of \( \succ \). (The transitive closure includes the case where the operands are equal; i.e. \( \forall X, X \succ^* X \).) Then, from the definition of \( A(X) \):

\[
A(X) = \sum_{a_i^d \in D(Y), Y \succ^* X} a_i^d
\]

If \( X \) is the root node in the lattice, the theorem follows trivially.

To use induction, consider some node \( X \) which is not the root in the lattice. Assume that the theorem holds for all nodes above \( X \) in the lattice. Then

\[
\sum_{Y \succ X} e(Y) + \sum_{a_i^d \in D(X)} a_i^d = \sum_{a_i^d \in D(Z), Z \succ Y, Y \succ X} a_i^d + \sum_{a_i^d \in D(X)} a_i^d = \sum_{a_i^d \in D(Y), Y \succ^* X} a_i^d
\]

which proves the theorem.

The correctness of the algorithms given below, depends on these theorems.

\[\square\]

6.4 ALGORITHMS

The algorithm to merge two authorization trees according to the above policies is quite straightforward; therefore, only a general outline is given here.

First, we assume that in both trees the rules were propagated to all nodes of the trees. Theorem 2 allows us to do this without considering all rules for every node, but only the nodes immediately superior to the node under consideration. It is therefore possible to propagate rules from the top of the lattice to the bottom efficiently.

Once this propagation is done separately for each AT, then the merge algorithm scans the two trees in parallel. There may be two cases:

1. The trees have exactly the same structure — for each node combine the rules according to the policies (see theorem 1).
2. One tree has parts which are not in the other. According to our assumptions these can be of three types: Disconnected parts, higher parts, lower parts.

   (a) For disconnected parts just union the authorization rules.
   (b) For higher parts union the rules and propagate their results down.
   (c) For lower parts, propagate from the shorter tree to the lower parts.

An important result that the algorithm must record and return to the optimizer, is the identification of the site of the ‘remaining’ rule. This means for example, that if one site denies local access to an object and another site allows local access, then according to our policy, access will be allowed. Now one can argue that the optimizer should be trusted to select the site using physical optimization decisions unrelated to security. On the other hand one may be ‘paranoid’ and argue that, for reasons of autonomy, the denying site does not like to yield that denied access, and would like to leave the ‘responsibility’ for a positive access to the other site. In this case the security related site information is important, and should be transferred by the algorithm to the optimizer. Also remember that the local access may have been denied for load or other reasons as described in the introduction.
6.5 COPYING AND MOVING OBJECTS

The existence of local autonomy systems as discussed above raises interesting issues with regards to copying or moving objects. Again, we want to maintain the autonomy of local users to copy local objects, while keeping the security of the system consistent. Note, that when we talk about copying objects, we usually mean copy classes and attributes; the special case of object instances is discussed below. Note also that copying an object (or attribute) may require copying of its class (and even superclasses). We are not specifying here the semantics of copy or move operations, but worry mostly about which access rights should be copied along with these classes.

6.5.1 Copying objects

Copying objects within the same site is handled by local policies and is of no interest here. When copying objects from one site to a second site, it is obviously assumed that the copier has read access to the object and create access in the other site. The main issue is whether or not the authorization rules in the first site are copied with the object. Below we discuss this issue for the four cases we described above.

1. **EQUAL, no local policies**  In the EQUAL one option environment case, the copier may be one of two classes of users:
   
   1. The SA herself, in which case she can grant any access rights she likes and the copied rights are of no concern.
   2. The user himself. Here, we believe the access rules governing this object should be copied (plus write and delete rights for the user).

   Remember that the principle of maximum access restricts queries to the local site from accessing more information than a global query would. This means that global restrictions (Negative rules) should always be copied along when the object is copied.

   Another option is not to copy any rights except for the copier's own rights. This may not achieve the performance results desired for other users, but will be simpler to implement.

   Note that copying should not always be permitted. We do not address policy issues in this regard in the current paper.

2. **EQUAL, with local policies**  Global rules should be copied as in the previous case. Whether local rules should be copied may depend on the circumstances. If a local negative rule is in force at the source site for performance reasons, load at the destination site may determine its desirability there. If, on the other hand, a local rule expresses the local site's opinion on whether a given subject should be allowed to access the information, the rule is better copied — otherwise copying information may affect the effective access rules for the object. While a change in effective access rules may be acceptable in the former case, it will almost always be unacceptable in the latter case.
Since all SAs have the same power, a local SA may change or delete a global rule without any problems.

3.4. Master-Slave cases These cases are similar to the respective EQUAL cases. The only difference is that a Slave SA cannot change or delete global rules.

6.5.2 Moving objects

Because of the inheritance structure, moving an object may cause many problems, since it also removes the access rules associated with the object which will impact lower objects access. There are several options:

1. Before moving the object, propagate the access rules below. However, since the distance that a rule has been propagated influences the effective access rule, this downward propagation may have to consider the existence of other rules. Alternatively, explicit rules may be given a 'distance' attribute so that it is possible to have an explicit rule as if it has been propagated some distance already. We do not consider this aspect further in the current paper.

2. Allow only SAs to move an object (actually only the SA from the source site, provided she has a create right on the target site). Then the SA may manually specify access rules for any objects below the moved object.

3. Allow copying of "root" objects only, and in that case move the entire schema and access rights associated with that root class.

The rest may be treated as in copy.

However, again for the Master-Slave situation a possible problem exists. Moving an object from a Master causes a problem if it is required that a Master should have copies of all objects that it has been designated master for.

6.6 OTHER ISSUES

6.6.1 Other Granularities

So far we have looked at the class hierarchy granularity. The policies for other granularities seem to be similar. The main exception may be the Class/Instance relationship (see also [5]). Generally, it should behave like a typical is-a relationship; for example, if in a single site there is a rule which denies access to a class, and another rule which grants access to an instance of a class, the 'shorter' second rule obviously wins. Even if the rules exist on separate sites the principle of maximum access determines the effective access rules. So most of the policies above apply. One case that may be different is that of a Master-Slave for instance-based protection. A class-based protection is often defined for an entire schema; this may not be effective when instances may also be protected individually: Since an instance is a physical object, e.g. a user's bank account,
there may be a site where that physical object usually resides, i.e. a 'home' site. Now these 'home' sites may be different for different instances! Therefore, different instances will have different 'masters'. It seems that it makes sense to define only a single home site per instance, and in that case the policies for home/no-home can be similar to the Master-Slave policies. The problem, however, is that generally such policies cannot be applied at compile time since, at compile time, we do not know what specific object will be required (if a query selects some instance of some identified class).

A similar problem exists for predicate-based access rules, and it was also pointed out in [6] that such checks are needed. How do we expect to integrate such checks? Currently, run-time checks are usually used by Mandatory systems or by Information-flow systems [9]. In SQL-based systems, instance-based rules are usually translated into views, and there will be as many views as instance-based rules. This, seems to be too much overhead in object-based systems, where instance-based rules may be more common.

What we suggest to do in this case is as follows: First, a note on the existence of such a conflict (between class and object) should be known to the optimizer and the optimizer should gather all these types of rules in a packet called the class-instance packet. The optimizer then can decide on the site from which to retrieve the instances without regard to this packet, but this packet must be sent by the run-time query evaluation to the chosen site, and that site must check the class-instance packet and apply the required policies, before transferring the results back.

6.6.2 Administration issues

The administration of security in object-oriented databases was discussed in [1]. Several issues were discussed there, including the impact of schema changes on authorization rules, the delegation and revocation of security contexts, and the relationship between inheritance and rule maintenance (addition and deletion of rules). Basically, all the policies from [1] can be used also in the autonomous objects case. One may, though, restrict the operation of some SAs in some cases. For example, as mentioned in earlier, in a Master-Slave environment slave SAs should not be allowed to define (or remove) global rules.

A new problem arises when a security administrator (SA) defines a new authorization rule or deletes an authorization rule which affects replicated objects (classes) in other sites. Our basic assumption that at different sites separate SAs may add and delete authorization rules still holds, since these rules will be merged by the access-evaluation algorithm as was discussed earlier. The only problem is with global negative rules. Whenever a new negative authorization rule is defined, except for the Local case, it should be propagated to all sites with copies of this object. This is essential, since otherwise we could violate the policy that a local access can never have more rights than a global one!

So the procedure for addition of rules is the following: If it is not a global negative rule (or a negative Master rule) then add the rule to the local schema,
otherwise propagate the rule to all relevant schemas\(^3\). Similar problems exist when an authorization rule is removed. A local SA should not be allowed to remove a global negative rule. Such a rule should be removed from all relevant schemas.

A related issue is which SA can do what? It may be the case that we have a hierarchy of SAs, where some can only define local rules on local schemas, others can define or remove global rules. A similar hierarchy can exist between master and local SAs. We think that such hierarchy can be handled by a simple role-based model (see \cite{17}).

6.6.3 Information Flow

Information flow in object-oriented databases was discussed in \cite{9}, using a run-time approach, and in \cite{3} using a compile-time approach. Both approaches can be extended to include the policies above. Basically, one has to construct the RACL and WACL lists using the policies above and update them when rules are added or deleted. Similarly, one has to construct the UAT and CUAT data structures of \cite{3}.

One problem that may arise is the following: Since local access is always more restrictive than global access, it may be that by local analysis a transaction may not cause information flow, while by global analysis it will. This creates a problem in local transactions since it restricts their freedom considerably, and, in particular, restricts the flexibility of local users developing local transactions. In such an environment it may make sense to propagate all rules to all sites. Only then is local flow analysis guaranteed to be correct. Further research is needed to see if there is a way around this undesirable propagation.

6.7 SUMMARY

In this paper a model for autonomous objects security was presented. The main problem in such a model is that each site may have its own security administrator who maintains its own authorization rules. A local site requires that local transactions behave consistently with local rules. The question that arises is what happens to global transactions and queries which may access copies of the local objects in various sites, and how conflicting authorization rules are handled. The paper provided a set of policies to handle the various cases, all are based on a simple principle, the principle of maximum access: by globally accessing the database you always get at least all the information you can get from a local access. Other issues such as object migration, rule administration and information flow were also discussed.

In conclusion, autonomy is an important concept in today’s distributed but connected database world. The issue of supporting this autonomy in a consistent and least restrictive way has been discussed in length.

\(^3\)Since propagation may take time, a two-phase commit protocol may be used to ensure consistency of the security specifications.
References


Abstract: This paper focuses on “programmable security” for object-oriented systems and languages. A primitive distributed object model is used to capture the essence of object behavior and access control schemes. This model can be used to construct virtually any distributed object language or system while supporting a spectrum of decentralized authorization models.

7.1 INTRODUCTION

High assurance security is crucial to deploying distributed object systems in mission-critical applications. Unfortunately, most software architectures and tools require designers to implement security from scratch. Such ad hoc solutions often fail in open distributed environments (Jonscher and Dittrich, 1995).

One solution is to integrate primitive security mechanisms and constructs for “programming” and “verifying” security within distributed object languages and architectures. This approach is similar to the incorporation of primitive data types, type constructors and type-checking in traditional programming languages. Like strong typing, providing and checking security at the language level will significantly improve code reliability.

This paper illustrates programmable security for object systems using security mechanisms embedded in a primitive distributed object model. The essence of object behavior and a flexible ticket-based access control scheme are incorporated in the primitive model. All object and security functionalities are captured and articulated using meta objects. This facilitates the construction of virtually any distributed object language or system – even C++, Java and CORBA – while supporting a spectrum of decentralized authorization models.
7.2 META OBJECT MODEL

The Meta Object Model (MOM), as presented by Hale et al. (1998), is a core distributed object model designed for building object systems and languages.

7.2.1 MOM Objects

MOM objects (Figure 1) comprise: (i) a message handler, (ii) three information repositories: object registry, metadata repository and object access control list (OACL), and (iii) object contents (methods and subobjects).

Message handlers delegate and process messages, at times interacting with method interfaces (for method invocation requests) and method arbiters (for method replies). Message handlers constrain the set of messages an object will accept from its immediate environment (domain). The addressing scheme for messages is based on MOM identifiers (local ids: \( \text{lids} \) and global ids: \( \text{gids} \)). The MOM authorization model employs tickets for access control. Message handlers contain message filters that provide access control by further constraining the set of accepted messages based on the embedded tickets and the local authorization state. The message filter in the message handler of an object authorizes messages by referring to the OACL for the local authorization state (set of prevailing locks) of the object.

Each MOM object also maintains an object registry with bookkeeping information (\( \text{lid}, \text{component type}, \text{etc.} \)) about each object component. Object registries are mainly used to avoid conflicts when creating and deleting objects.
Metadata repositories provide meta objects with templates for creating objects, manifesting the emergent object-oriented behavior of classes and metaclasses.

7.2.2 Message-Passing and Methods

MOM messages are processes that persist until they are consumed. They carry method invocation or authorization requests, acknowledgements and replies.

Message handlers accept or reject messages and marshal object requests. They also control the distribution of requests and replies. An incoming message can be received as a local request or it can be delegated to another object in an adjacent domain.

MOM's method architecture has three components: (i) method interfaces, (ii) method arbiters and (iii) method bodies. Each method uses a distinct method interface to accept method invocation requests and manifest synchronization constraints. A method interface spawns a method arbiter and method body upon acceptance of a method invocation. Method arbiters negotiate communication between method bodies and their environments.

Methods can be mutable or immutable. Mutable methods model instance variables by creating processes with state that can be accessed multiple times. Immutable methods are stateless methods that terminate and return values. They often serve as interfaces to instance variables (accessor methods).

Each immutable method invocation produces a distinct method arbiter. On the other hand, mutable method interfaces prohibit the creation of new method arbiters until the previous arbiters have terminated. Requests to an active mutable method are forwarded to the active arbiter.

Immutable method interfaces create new method arbiters and bodies for each request. An arbiter passes arguments to its method body and waits for requests from the method body and/or replies from methods invoked by the body; it formulates a reply message at the termination of the invocation.

Component creation and deletion are handled by special methods that refer to object registries to prevent naming conflicts and maintain consistency. Metadata repositories are queried for the structures of new objects.

7.2.3 Security in MOM

Message filters (Jajodia and Kogan, 1990) in MOM message handlers accept messages by comparing embedded tickets with the local authorization states of objects. The local authorization state of an object defines a set of ticket-based permissions that are recorded in its OACL. These two mechanisms permit the implementation of a variety of authorization models for secure object systems.

OACLs are local information repositories with tuples defining object authorizations. The tuple <Comp,Priv,Tok> specifies that component Comp associates privilege Priv with token Tok. E.g., <Interface1,lock,a> states that Interface1 has a lock associated with token a, i.e., Interface1 is accessible by all messages holding a tickets. Component tickets are also held in OACLs. The tuple <Arbiter2,key,b> states that b is a ticket held by Arbiter2.
7.2.3.1 Ticket-Based Security. Tickets are unforgeable tokens visible only to trusted processes (message filters and OACLs). Untrusted processes (messages and methods) may carry and pass tickets. Methods can request that new tickets be created, but they cannot forge old ones.

Ticket distribution and revocation are thorny issues (Gilgor et al., 1987). A naive distribution policy allows owners to distribute tickets. Another might permit ticket holders to distribute tickets (under certain circumstances). Ticket revocation is more complicated, particularly for capabilities where revocation must be partial, selective and transitive.

MOM provides mechanisms for adding and removing tickets from OACLs, but these cannot ensure that the critical properties of revocation and distribution are respected. Constraints may be built on top of MOM to enforce distribution/revocation policies (or implement access control models). This is accomplished using structured tickets and reconfiguring message filters.

7.2.3.2 Authorization Model. MOM uses a ticket-based scheme for its authorization model. The authorization model resolves (s,o,a) tuples as TRUE or FALSE for subjects s, objects o and access types a. An authorization state is defined by \( \text{State : Object} \rightarrow \text{Privilege} \rightarrow \text{Token} \rightarrow \text{Bool} \) where Token is an atomic ticket representing the subject. MOM tickets (or keys) are unforgeable tokens that are embedded in messages. The dual of a key is a lock, which is associated with a token and a privilege type. Locks define the authorization state of an object by specifying ticket-based permissions on object components.

Figure 2 defines MOM's model of grant and revoke privileges. The model permits access types such as \text{GRANT} \cdot \text{LOCK} and \text{GRANT} \cdot \text{REVOKE} \cdot \text{KEY}. Every type other than \text{KEY} behaves as a lock. E.g., \text{GRANT} \cdot \text{REVOKE} \cdot \text{KEY} is a lock that applies to \text{REVOKE} \cdot \text{KEY} privileges. A subject with a key matching the \text{GRANT} \cdot \text{REVOKE} \cdot \text{KEY} lock held by an object can add (grant) tokens to \text{REVOKE} \cdot \text{KEY} list.

The ALL privilege confers all privileges to a subject. A subject holding a key matching a lock ALL held by a component has complete access to that
Rule 1: \( \forall p : \text{priv}, o : \text{obj}, t : \text{token}, s : \text{state}; s o \text{ALL } t \Rightarrow s o p t \)

Rule 2: \( \forall s_1, s_2, \exists c : \text{comm}; \text{EVAL} c s_1 s_2 \Rightarrow \text{TRANS} s_1 s_2 \)

Rule 3: \( \forall p, s_1, s_2, o_1, o_2, t; s_1 o_1 \text{KEY} t \land s_1 o_2 \text{GRANT} p t \Rightarrow \)
\( (s_2 o_2 p t \land (\forall o', p', t'. o' \neq o_2 \lor p' \neq p \land t' \neq t) \Rightarrow s_1 o' p' t' = s_2 o' p' t') \Rightarrow \text{EVAL} (\text{ADD} p t o_2 o_1) s_1 s_2 \)

Rule 4: \( \forall p, s_1, s_2, o_1, o_2, t; s_1 o_1 \text{KEY} t \land s_1 o_2 \text{REVOKE} p t \Rightarrow \)
\( -(s_2 o_2 p t \land (\forall o', p', t'. o' \neq o_2 \lor p' \neq p \land t' \neq t) \Rightarrow s_1 o' p' t' = s_2 o' p' t')) \Rightarrow \text{EVAL} (\text{REMOVE} p t o_2 o_1) s_1 s_2 \)

Rule 5: \( \text{EVAL} (c_1) s_1 s_2 \land \text{EVAL} (c_2) s_2 s_3 \Rightarrow \text{EVAL} (c_1; c_2) s_1 s_3 \)

Figure 3. Authorization semantics.

The command set in Figure 2 permits dynamic and explicit authorization state modification. Commands can be embedded in messages as authorization requests. Subjects can add or remove ticket-privilege associations in objects for the tokens they hold as keys. Command sequences are permitted in messages.

Figure 3 provides the authorization model semantics. Rule 1 defines the ALL access type. Rule 2 formalizes the predicates EVAL and TRANS in Figure 2. Note that EVAL returns TRUE when a command will take one state to another while TRANS returns TRUE if a transition between states is possible. Rule 3 defines the ADD command. A subject must have grant privilege over an access type in an object to add a token of that type to the object. It also stipulates that subjects can only add tokens held by them as keys. Rule 4 defines the REMOVE command. It specifies when it is legal for a subject to remove authorization tuples. Rule 5 formalizes the transitive nature of commands.

7.2.3.3 Ticket-Based Access Control. Tickets, message filters and OACLs provide access control in MOM systems. Meta objects can own OACLs with local authorization state information. Each OACL contains authorization tuples of the form <component, access_type, token>. (Tuples with method names in component fields implement method based access control (MBAC) (Gal-Oz et al., 1993).) Messages carry the access type, e.g., GRANT.LOCK, and tokens (tickets) owned by the subject. Message filters authorize messages against OACLs. Messages are authorized if they contain keys (tickets) that match locks held by the intended recipients. Message filtering can occur at all objects along the message route. However, performance can be improved by placing message filters only in strategic objects.

Meta objects refer to metadata repositories to define the initial authorization states for objects they create. Classes are meta objects with methods for constructing instances. Authorizations can be inherited by instances/subclasses by token propagation and message delegation. Access to instance variables is controlled by instances and defined by token propagation at instance creation.
Access to methods can be controlled by classes if invocations are to be delegated from instances to classes. This implements implicit authorization flow. Objects manifest explicit authorization flow by invoking methods containing authorization commands. Authorization commands issued in messages can modify the OACLs of destination objects.

7.3 PROGRAMMING ACCESS CONTROL

This section shows how MOM's ticket-based scheme is used to implement various access control models.

7.3.1 Discretionary access control

Discretionary access control (DAC) is based on subject identity. It gives the owner of a resource the authority to grant or deny access to the resource. Standard permissions are read, write and execute, but grant and revoke permissions can be included. MOM models these as permissions to execute methods and send messages. Authorization states can be modified using authorization commands embedded in messages.

Figure 4 shows syntactic constructs for issuing authorization requests within methods. Implementing DAC involves mapping tickets to identities. Therefore, an authorization request specifies a reference to the object at which the request is directed. Simple authorization commands in the request tell the object to add
7.3.2 Mandatory Access Control

Mandatory access control (MAC) requires subjects and objects to be tagged with security clearances and classifications defined by a partially ordered set of label pairs: a security level and a category, e.g., (top secret, crypto).

MAC is modeled in MOM by mapping tickets to clearances. A unique ticket exists for each label pair in the partial order, e.g., the (secret, air force) clearance might be associated with ticket secairtkt.

Figure 5 shows an instance variable and a method (in different classes in the same package) that have been classified as top secret and secret, respectively. The effect of method doit() calling report(rank) is to send a method invocation to the read accessor for rank. The invocation is denied by the filter controlling access to rank because it lacks the appropriate ticket.

7.3.3 Role Based Access Control
Role based access control (RBAC) assumes that subjects adopt one or more roles, each defining a set of permissions. MOM implements RBAC by mapping tickets to roles.

Figure 6 shows a class `human` that can assume `teacher` or `mechanic` roles. Templates for the two roles specify the appropriate permissions. When an instance of `human` is created, a role is selected and permissions from the role template are given to the instance. The effect of the code in Figure 6 has instance `joe` taking on a `teacher` role; it is given the appropriate ticket `teachktkt` from the `teacher` role template.

### 7.3.4 Task Based Access Control

Task based access control (TBAC) apportions trust on a transaction by transaction basis. Implementing TBAC requires functions that operate before and after the transaction to verify preconditions and postconditions. Figure 7 shows a `cashier` charging a `consumer` for a purchase. Note that `consumer` permission is a precondition for the debit procedure. After the debit is completed, `consumer` revokes this permission so that he/she cannot be charged again.

Programming language support for TBAC involves the ability to name tasks and define Boolean functions `before` and `after` that verify preconditions and postconditions, respectively. A task does not commence unless `before` returns `TRUE`. It does not complete and must rollback if `after` returns `FALSE`.

Figure 7 shows how temporary trust is established between `consumer` and `cashier`. First (in the `before` function), `cashier` must get `consumer` approval. Then, `consumer` adds a matching lock and key to account and `cashier`, re-
It_task := task : before { ... } : after { ... }

Task id
epsilon

Item := vis method performs id

Effect

Code

task charge :
: before {
  /* get permission */
} : after {
  /* notify consumer */
} performs debit(); ...

Figure 7. Task based access control.

spectively. After the before function terminates and returns TRUE, the debit method is invoked and a message is sent with the temporary key to debit the account. The after function then notifies consumer that the debit is complete, and consumer removes the lock from account.

7.4 PROGRAMMING SECURE INTEROPERABILITY

MOM facilitates the secure interoperation of heterogeneous distributed objects at the source and object levels. Source level interoperability is a natural outcome of modeling distributed object systems with MOM as the common substrate. The MOM execution model supports object level interoperability by permitting the encapsulation of native object implementations with MOM wrappings. These wrappings augment native object execution by marshaling communication between objects and engaging MOM's programmable security constructs. The following subsections describe MOM's runtime system and Mumbo, a MOM-based language for orchestrating secure interoperability.

7.4.1 MOM Runtime System

The MOM runtime system is a distributed virtual machine for MOM objects that manifests concurrent message-passing and method invocation. Any object system with a MOM mapping can operate in the MOM runtime system.

Source level interoperability requires mappings between the source languages and MOM. E.g., if C++ and Java are given MOM mappings, any C++ and Java programs could be compiled into MOM and could reside in the same object
space. While this permits interoperation, it does not address secure interoperation. But MOM’s programmable security constructs can still be used to create extensions of C++, Java, or new languages that promote secure interoperation.

Source level interoperability is not always practical. Source code may not be available or it may not be efficiently mapped into a MOM system. Native agents in MOM can integrate legacy code seamlessly at runtime using a modified MOM message handler that converts messages into calls to native objects. This message handler generates calls to native functions in dynamic link libraries (DLLs). The approach has two benefits. Wrapping native resources inside MOM objects allows them to inherit the authorization services of the wrapper objects. Classes that form the basis of language extensions and virtual operating systems are readily developed using wrappers. Furthermore, I/O routines are easily implemented as native methods in MOM-based languages.

7.4.2 Mumbo

Mumbo is an object coordination language built from MOM. It permits synchronization of object resources and promotes interoperability using wrapper/translator technologies. Mumbo resembles Java in that instances, classes and interfaces are the main components in program development. However, Mumbo treats classes and interfaces as meta objects for flexibility and fine-grained concurrency. Mumbo also employs MOM’s meta object access control scheme to integrate DAC and traditional object-oriented protection.

Mumbo’s runtime system is a MOM runtime system. At runtime, a Mumbo domain object is placed in a MOM root domain to encapsulate the compiled system. Mumbo permits the introduction of new systems (users) at runtime. A user can join the runtime environment (when a new Mumbo domain object is added to the existing root domain) or start a new environment (when a distinct root system is spawned for the Mumbo system).

7.4.2.1 Primitive Elements. Mumbo currently has three primitive types: Names, Booleans and Lists. Other primitive types are easily added. Abstract data types can be created by class definitions. A generic Object type is introduced to denote a base type for objects.

Names are atomic data elements in Mumbo represented by character strings. They can be compared (by equality) but not modified, e.g., appended or truncated; they are primarily used as list elements.

Booleans are represented by the named constants TRUE and FALSE. The standard Boolean functions and, or, not and the equality conditional are available. Operators instanceof, implementationof and elementof are Boolean expressions in Mumbo: instanceof returns TRUE if an object (evaluated from an expression) is an instance of a class object; implementationof checks if an object resolved from an object expression is an implementation of an interface; elementof tests the membership of an evaluated expression in a list.
Lists in Mumbo are sequences of expressions (including other lists); they evaluate to sequences of primitive data elements. The standard list functions head, tail and cons are available.

7.4.2.2 Mumbo Objects. Classes, interfaces and instances are constructed from MOM objects. Each object expression evaluates to an object reference, a gid that uniquely identifies its Mumbo runtime system. Access to a slot or method mandates the use of an object expression to specify a referrent. Any access specified without an object expression is assumed to be local.

Mumbo methods employ the native modifier to denote a method interface to be used as a proxy for a native function. The name of the DLL follows the native modifier and the method must have the same name as the native function in the DLL. E.g., native public void mydll.myfunction(); declares a native method implemented by myfunction() inside mydll.dll.

Since Mumbo classes (and metaclasses) are first class objects, the opportunity exists for dynamically modifying class and metaclass behavior at runtime. Inheritance is modeled by delegating messages to object superclasses.

Interfaces contain method and slot signature information inside the metadatas repository. This information is used to determine whether or not an object implements the specified interface. Interfaces can be instantiated to create interface objects that define the roles of class instances. A novel feature of Mumbo methods and slots is their ability to claim responsibility for implementing a piece of an interface. Methods with different names can satisfy a portion of an interface as long as they have matching signatures. This feature facilitates abstraction within the interface/component framework.

7.4.2.3 Discretionary Access Control in Mumbo. The opportunity for interaction between distinct Mumbo units (users) poses hazards to the resource security and integrity. Mumbo employs MOM's ticket-based access control scheme to provide authorization services to Mumbo elements. (Tokens are associated with Mumbo elements to implement DAC.) Mumbo enables developers to use traditional notions of public, protected and private elements to specify initial authorization states for Mumbo programs. Protection can then be fine-tuned using authorization commands issued from within method bodies.

For example, the command Authorization Request A.mycar {Grant this Lock; Revoke this.private Grant.Lock}, asks A.mycar to perform two services: (i) grant it a Lock for the requesting object, and (ii) remove from A.mycar() the ability to grant a Lock by this.private.

Initially, a developer can specify an element as public, protected or private. Public elements are open to everything in the root environment. Private methods are available only to instances of a class. Private slots are only visible inside an object. The designation of an object, method or slot in Mumbo as protected means that it is accessible only within its Mumbo domain. Permissions on all elements can change, e.g., public elements can become private and vice versa. Furthermore, the authorization states of elements at any given time could be
neither public, protected nor private, depending on whether authorization commands are issued. Note that public, protected and private protection modes are intended to permit developers to easily specify the initial authorization states of Mumbo elements.

7.5 RELATED WORK

The foundational object systems, ACTORS (Agha, 1986) and Loops (Stefik and Bobrow, 1985), have motivated this work. Both systems capture object behavior using meta objects. Recent work by Abadi and Cardelli (1996) seeks a common formalism for object behavior. MOM’s ability to support a variety of decentralized authorization models for distributed objects also arises from reconciling object behavior and access control in meta objects.

Authorization for distributed objects has been addressed by Nicomette and Deswarte (1997) and van Doorn et al. (1996). The former approach is similar in that it relies on collaborative security kernels for decentralized access control. However, it promotes vouchers – indirect access rights transmitted by objects with capabilities. Vouchers are intriguing because they support the principle of least privilege in capability-based systems. Modifying the structure of tickets to include vouchers is a natural extension to MOM. The latter approach extends Modula-3 network objects to secure network objects (SNOs) with security features and promotes subtyping as a means of specifying security properties for objects. SNOs and MOM bind object-oriented programming languages into service for integrating security into objects and methods.

Java addresses program security in open distributed environments with a novel security architecture (Dean et al., 1996). The architecture centers on a security manager that authorizes specific method invocations. Unfortunately, corrupting the security manager effectively circumvents access control. Our decentralized approach integrates security functionality within each object, resulting in more robust security solutions.

Database security research has influenced access control of objects (see, e.g., Dittrich et al., 1989). Several authorization models have been proposed (e.g., Bertino et al., 1994; Rosenthal et al., 1994). Method based access control (MBAC) for object systems was introduced by Gal-Oz et al. (1993). Access types are reduced to a single execute type by considering methods as the primary basis for access control. MBAC is a natural choice for MOM because all access occurs through method invocation.

ORION/ITASCA adopts DAC for objects (Rabitti et al., 1991). It embraces notions of explicit/implicit, positive/negative and weak/strong authorizations. The authorization model is based on four fundamental access types and incorporates roles. An extension by Bertino et al. (1994) supports additional access types, type dependency modeling and distributed authorization control. MOM type definitions can be extended to model positive/negative and weak/strong authorizations by reconfiguring message filters. Semantic-based forms of implicit authorization naturally emerge from object systems built with MOM.
Providing flexible mechanisms and models that support multipolicy access control is becoming increasingly important. In theory, each object in a multipolicy environment could be protected according to a different policy. Bertino et al. (1996) employ flexible access control mechanisms and mediators (Wiederhold, 1992) to “tune” access control mechanisms to specific policies. Multipolicy systems seek the highest common ground in access control – as does MOM – to support the interoperability of disparate authorization policies.

The Argos system unifies heterogeneous access control models in an open distributed environment (Jonscher and Dittrich, 1995). It offers a configurability for modeling various identity-based authorization policies. While identity-based authorization models are pervasive, MOM’s approach using tickets is more general. A ticket can represent an identity, a role, a transaction, or a clearance level and can therefore be used to implement a variety of authorization policies.

The Distributed Computing Environment (DCE) employs a decentralized authorization service for access control in open distributed environments (Rosenberry et al., 1993). DCE manages authorizations with access control lists (ACLs). Principals (subjects) are registered and assigned group and organization membership. A member’s name, group and organization information define its privilege attributes. Member privilege attributes are embedded in a ticket provided by the authentication server at login. An ACL manager resides in each file server to authorize access requests. DCE supports various authorization models by allowing customization of ACL managers. Our approach also provides a common set of mechanisms for interoperability of heterogeneous components, but it also permits the uniform treatment of secure distributed objects in new language-based environments.

7.6 CONCLUSIONS

Online enterprises require high assurance security for mission-critical services. Developers are hampered by the lack of tools and methodologies for constructing verifiably secure distributed object systems. The Meta Object Model (MOM) integrates object functionality and primitive access control mechanisms to facilitate the development of secure distributed object languages and systems. MOM has been used to design Mumbo, a coordination language providing discretionary access control for distributed objects.

Future work will focus on augmenting MOM with authentication and audit mechanisms. Plans also include mapping C++ and Java to MOM, and using Mumbo to pursue secure interoperability of heterogeneous distributed objects.

References


V Mediation
Abstract: In this paper we discuss the security requirements for mediation, and present our approach towards satisfying them, with an emphasis on confidentiality and authenticity. Furthermore we outline the design of the basic security mechanisms for mediators. Our basic approach suitably combines the concepts of credentials, for authentic authorization with some kind of anonymity, and of asymmetric encryption, for confidentiality, and it can be extended to include additional mechanisms like digital signatures and fingerprints. Additionally it adopts the model of role based security policies because of its application orientation and of its potentials to integrate and unify various policies.

8.1 INTRODUCTION

Recent trends in information technologies led to vastly improved communication facilities like the Internet, an explosion of on-line multimedia information providers, and challenging new demands of users. Whenever a user looks for a piece of information, he may aim at identifying promising sources, which can be quite heterogeneous and autonomous, and then retrieving and integrating the required data. And whatever data a source has to offer, it may aim at supporting a wide range of potential clients, which in general are unknown in advance. According to these trends, various forms of interoperable information systems have been developed. While federated database systems [26, 18]...
have already come into existence, increasingly ambitious further demands have evolved and resulted in the paradigm of mediated information systems [31, 33]. Some current projects on mediation are TSIMMIS [29], HERMES [27], Information Manifold [17], SIMS [1], AURORA [34], DISCO [28], Squirrel [13], DIOM [19], Garlic [7], OBSERVER [20], InfoSleuth [2], and MMM [3, 4].

In mediated information systems a client, seeking for information, and various and autonomous sources, holding potentially useful data, are brought together by a third kind of independent components, called mediators. Mediation is required to deal with the heterogeneity and the autonomy of the sources, not only from the functional point of view but also with respect to all aspects of security. This includes confidentiality and authenticity, as well as integrity, anonymity, non-repudiation and availability.

Previous work on security of interoperable information systems has mainly been done for federated databases [15, 10, 8], where the emphasis laid on resolving heterogeneity. According to the structure of federated databases, the security mechanisms were identity rather than credential based. There also appeared some contributions to security in mediated systems [6, 32, 14].

The concept of credentials has been advocated by Chaum [9] for supporting privacy in networked systems. Since then it has been adopted for various purposes in interoperable systems, for electronic payment and marketplaces as well as for middleware systems like CORBA [21]. Further work includes [25, 5, 11].

The model of role based security policies [24, 16, 12] has been successfully used before, and in particular studied for integrating various policies.

8.2 REQUIREMENTS OF FEDERATED AND MEDIATED SYSTEMS

While both federated database management systems and mediated information systems are used to integrate various autonomous information sources, several differences between both approaches may be identified. Most interesting in the context of this paper are differences related to and affecting security issues. We first observe that many of the differences result from differing motivations of participants of federated and mediated environments, respectively.

In a federated system the federation establishment is stimulated by the members of the information source organization in order to support a closed group of client users. In many cases the client group is part of the organization which supports its interactions by means of the federation. Furthermore there exist dependencies between clients and information sources due to their identical organizational origin. The information sources act in a common interest anchored in the organization they belong to. This network of dependencies probably has had significant impact on specific security architectures as often found in federated systems. As mentioned above, the information sources directly belong to a holding organization and therefore are trusted. Most federated systems do not authenticate information sources in a client-verifiable way. The methods used to integrate information sources at the federation layer often involve administrative interaction, which takes place before client queries can be accepted.
This suggests, that the set of information source layer members is rather static than dynamic. Under the assumption of a closed world the static nature of this approach leads to temporary loss of service when one or more information sources, which hold information relevant to the client query, are unavailable.

Opposed to information sources the federation clients are not trusted and require proper authentication and authorization. Because clients normally are members of the federation's organization, there is a closed group of registered users. New users are assigned predefined roles but some systems also support anonymous client accesses.

The motivation to integrate information sources using mediators is quite different. Clients demand systems enabling them to effectively work with heterogeneous information sources. This demand stimulates information sources to supply their information on an ad-hoc basis, in particular for purchase. Information sources are likely to meet the client's requirements and to cooperate with mediators. Like in a marketplace of supply and demand there exist different motivations for cooperation in each layer. Generally clients, information sources and mediators are independent of each other. Information sources exist in competitive and non-competitive relationships with other information sources. Obviously there is no base for mutual trust between the three layers of a mediated system. While information sources probably will have cooperation contracts with mediators, it can be assumed that spontaneous clients are unknown beforehand. Clients thus cannot be registered in a static way before queries can be accepted. Even the group of information sources probably won't be as stable as found in federated systems due to the lack of organizational associations in mediated systems. Though one or more information sources may be temporarily unavailable a client query can be satisfied. There apparently is no useful assumption of a closed world in mediated systems due to their dynamics. Other new requirements relate to non-repudiation issues (e.g. origin, affirmative authorization) for traded items.

We believe that mediated systems are more suitable to model dynamics and low trust of interacting parties. A mediator's top-down design paradigm allows for a stable presentation schema of integrated information at varying degrees of source fluctuation, whereas a bottom-up approach requires a redesign of the global integration schema each time a local schema changes or is added. Mediators strive for tolerance with respect to information provider failures and offer service to ad-hoc clients which have not registered with the service beforehand. The latter forbids employment of merely identity based identification approaches as traditionally used in federated systems. This paper shows a possible approach to achieve secure mediation while considering our trust model and high dynamics in a certain mediation scenario as presented in the following sections.

8.3 SCENARIOS

Basically there are two extreme scenarios for mediator security handling imaginable: simple forwarding or complete mediation of security information. Both
scenarios feature benefits and drawbacks, of which some will be outlined here. Based on this analysis we will postulate a hybrid scenario that will be taken as a motivation for our approach.

In the *simple forwarding* scenario security requirements and their fulfillment travel back and forth between clients and information sources, while being forwarded completely unmodified and uncomplemented by the mediator. That is, all three layer participants can authenticate each other. When identities are authenticated, it is difficult to allow anonymous clients. On the other hand information sources know their clients and may use fine grained authentication, authorization and accountability. The sources inform clients about their security requirements via the mediator. In this scenario the mediator does not complement or modify these specifications such that the clients are completely aware of each used source's security requirements. A client may profit from a wealth of detail, but it is the client software's business to present a general view of a query's security requirements. Since in this scenario the mediator does not provide an integration layer for source policies, it is the client software's duty to do that. Consequently, the necessity to trust the mediator is limited to forwarding security information properly and privacy preserving.

A mediator which provides *complete mediation* of security information retrieves security requirements from information sources and integrates them. It presents the clients a coherent view of security requirements for a given query. This layer of abstraction conforms with the presentation of external objects. On the other hand an isolation of clients and sources is artificially created. While this allows for anonymous clients, it has severe drawbacks on the granularity of authentication, authorization and accountability at the sources. Sources cannot destine query results for specific clients and the latter cannot directly determine the origin of results. Obviously in this scenario clients as well as information sources need to trust the mediator.

Our approach is based on a compromise of both of the above scenarios to get the best of both worlds. It is one of a mediator's design goals to integrate information and we seek to achieve integration for security information, too. On the other hand we think that it is necessary to provide information sources and clients with sufficient information to establish a secure relationship via the mediator. Further it is a goal to protect the client's privacy and to minimize the necessary trust towards the mediator. In our hybrid scenario the mediator integrates source requirements and lets the clients choose, how much information to divulge about themselves. Only minimal necessary information about the clients then is sent to the sources. Subsequently, they can authenticate, authorize and audit the clients and appropriately protect results. The mediator cannot use results in a fraudulent way but is still able to integrate results of different sources.
8.4 DESIGN OF SECURE MEDIATION

8.4.1 Fundamental requirements of secure querying

The following security requirements for querying are considered: 1. Any source wishes or is even legally obliged to autonomously follow a security policy with respect to confidentiality which ensures that requested information is delivered to appropriate clients only. In order to achieve this goal, clients have to provide evidence that they are eligible for requested information, and sources have to maintain mechanisms to inspect such evidence and to decide whether and which information is returned. Furthermore, a source has to ensure that information is actually delivered to only that client which provided the inspected evidence. 2. The policy with respect to confidentiality as stated above should be at least compatible with additional viewpoints concerning authenticity, anonymity, integrity and availability. 3. And any client wishes that shown evidences cannot be misused.

Surely, these fundamental requirements should be met for the simple case that a client directly addresses a source, as well as when both the client's request and the source's delivery are mediated.

8.4.2 Basic informational environment

We assume that there are trusted third parties (TTPs), trusted by all participants of a transaction, that offer at least the following services:

- A TTP signs a certificate of the rough form
  \((\text{identity(address)}, \text{public (encryption) key, public (verification) key})\),
  thereby assuring that the participant specified by the first component is the owner of the keys.

- A TTP signs a credential of the rough form
  \((\text{attribute, public (encryption) key, public (verification) key})\),
  thereby assuring the attribute specified by the first component is enjoyed by the owner of the matching secret keys for decryption and signing.

Our basic protocols employ attributes contained in credentials, when shown to a source, as evidence that the owner of the matching secret keys might be eligible for some requested information. That is, a source decides on the basis of the presented attributes, whether and which information is returned. It is important to observe, that the source does not care how it has got knowledge of the credentials, whether directly from the owner of the matching secret decryption key or otherwise.

For the basic protocols we always only need the public key for encryption in credentials, as sketched in the following. Suppose that a participant wants to ensure that some returned data contain meaningful information only for the supposed owner of the matching secret decryption key. Then the participant
takes care that the delivered data is the ciphertext of the plaintext which con­
tains the information under consideration, where the encryption is done with
the public key. The other keys are merely provided as a precaution for more
advanced protocols.

8.4.3 Secure direct querying

Given this basic informational environment, we can specify the basic protocols.
We distinguish a preparatory phase and a query phase.

In the preparatory phase, clients and sources do not yet interact. A client,
wishing to request information later on, assembles credentials with his at­
tributes supposed to provide evidence of his eligibility. And a source, entitled
to answer queries later on, defines a security policy with respect to confiden­
tiality which relates sets of attributes to the amounts of information allowed
for delivery. More precisely a security policy is abstracted to be specified in the
following form:

- As input, the policy accepts some set of credentials belonging to a unique
  owner. For instance, this is the case if all occurring public keys are the
  same. It is important to observe that it is not necessary for the source to
  know the identity of that owner.

- Only based on the set of attributes shown by the credentials, the policy
  states which kind of information is allowed to be delivered to the owner.

The protocol for the query phase is outlined as follows.

Protocol for secure query answering.

1. The client sends a request (identity(address), query, set of credentials) to
   the source.
2. The source verifies each credential, checks whether the set of credentials
   is acceptable, i.e. belong to a unique owner, and determines the associated set
   of attributes.
3. The source evaluates the query under the restriction that only such in­
   formation is generated that, on the basis of the associated set of attributes, is
   allowed to be delivered.
4. The result of the restricted query evaluation is considered as plaintext and
   encrypted with (some of) the public key(s) occuring in the shown credentials.
5. The resulting ciphertext is sent back to the client.

This protocol satisfies the fundamental security requirements, as stated in
Section 8.4.1:

a) Clients and sources exclusively have to trust the TTPs that signed cre­
dentials. b) Sources are supposed to have an interest in checking eligibility.
Thus, for instance, data subjects, the data of which is stored in a source, have
to trust the source with respect to checking eligibility appropriately. c) Eligibility
is supposed to be definable in terms of attributes. d) Since attributes
are shown in the form of credentials that do not contain a field for the identity of the owner, a client can stay anonymous as far as the source cannot infer the identity from its knowledge about the attributes and the connection data. e) Even in an untrusted network, only the unique owner of the verified credentials can recover the plaintext and thus gain the requested information. However, in some situations, we should have to take care of possible plaintext attacks. These situations are given if an attacker is himself eligible for another user’s eligible request. A possible countermeasure would be to employ nondeterministic encryption, i.e. adding some random data to the plaintext before encrypting it. f) If any participant misused somebody else’s credentials, the correct owner could be erroneously or maliciously blamed for the request. However, in a dispute about the sender of the request nobody can exhibit any essential evidence pro or contra the blame. If there is an interest in documenting the sender of a request, the protocol must be extended by appropriately signing the request. g) Further concerns about authenticity or requirements on integrity could be dealt with by additional actions that are based on appropriate signing. These actions would be founded on the certificates offered by the basic informational environment. h) In particular, if a source is concerned about a client redistributing received data without the source’s approval, the source can fingerprint the delivered copies of the data. i) There are no specific provisions or additional obstacles to availability.

8.4.4 Secure mediation

We now extend the approach presented in Section 8.4.3 for the case of mediation. To begin with, we ignore security requirements for the moment and just state a rough abstract protocol for mediated query answering, see also Figure 8.1

Protocol for mediated query answering.

Request phase.

a) A client C sends a global query q to a mediator M.

b) The mediator M decomposes the query q into a set of subqueries qs, where the subquery qs is supposed to be appropriate for some source S.

c) The mediator sends the subquery qs to the source S, for each relevant source S.

Delivery phase.

d) Each relevant source S evaluates its subquery qs and produces a local answer consisting of data ds.

e) Each relevant source S sends its local answer ds back to the mediator M.

f) The mediator M integrates the received local answers ds into a global answer d.

Now taking care of the fundamental security requirements we can easily combine the basic protocols for secure query answering with the protocol for
mediation. In the straightforward case the protocols for the preparatory phase remain unchanged. And the mediation protocol is modified by integrating the basic protocol for the query phase as follows. In step a) the client includes a set of credentials into the request for query $q$. In step c) the mediator just forwards the received set of credentials to each of the relevant sources. In step d) each relevant source performs the security actions of step 2) of the basic protocol, and query evaluation is restricted as stated in step 3) of the basic protocol. Finally, in step e), each relevant source first encrypts its local answer according to step 4) of the basic protocol before sending it back to the mediator.

It can be checked that the fundamental security requirements are invariantly satisfied as for the simple case. There are only some minor restrictions. We note two aspects. Now another participant, the mediator, acquires knowledge about the client's credentials and thus could give raise to false blames about the sender of a request. But this possibility does not introduce a substantially new problem. And a mediator may compromise the integrity of data, if no additional actions are taken. There is also an improvement with respect to a
client’s wish to stay anonymous with respect to a source, since in general there is no need for a direct connection between the client and a source.

However, there are important observations dealt with in section 8.4.5. Firstly, the functional requirements on the mediator may be seriously affected if in the integration step f) the expected operations for integrating local answers can not be performed on the ciphertexts. And secondly, step b) can be greatly improved by facilities of the mediator to assist a client in the management of credentials.

8.4.5 Advanced secure mediation

8.4.5.1 Layered mediation. So far we treated the case that there is only one layer of mediation. However, we have argued that mediation does not essentially affect the fundamental security properties of direct querying, and thus we could use our approach also for mediation across several layers.

8.4.5.2 Referencing and using public encryption keys. In Section 8.4.2 we simply assumed credentials to contain the public encryption key of the attribute’s owner. And in Section 8.4.3 we showed a straightforward way how a source can employ such an encryption key. These features allow some useful variations. Firstly, there is no essential need for including the public encryption key in the credential. In place of the key itself it is sufficient to equip the credential with information on how to retrieve the key of the attribute’s unique owner. And secondly, confidentiality of delivered information can also be ensured as follows: The source encrypts the answer with any session key using any encryption method, and it encrypts only the session key with the public key. Then both the ciphertext and the encrypted session key are returned to the client.

8.4.5.3 Mediated management of credentials. In Section 8.4.4 we presented a modified mediation protocol, in which the mediator during step c) just forwards the received set of credentials to each of the relevant sources. There is room for a lot of important improvements which, basically, assist a client in managing his credentials. The most important issues to be addressed are: A client may wish to present a minimal set of credentials to each of the relevant sources. He may also require that, if there is any choice to answer his global query, the mediator should decompose the query in such a way that subqueries are sent to sources with minimal credential requirements. More generally, a client would like to be assisted in revealing as few of its attributes as possible. On the other hand, a client may specify a wanted level of quality with respect to the global answer to his query. This goal requires that the mediator takes best advantage of all available credentials. More generally, a client would like to be assisted to achieve a maximal level quality of the answer. Obviously, in general there will be a tradeoff between minimizing the use of credentials and maximizing the quality of information. Accordingly, a client would like to be assisted in balancing the conflicting goals. Even more generally, a client
would like to negotiate with the mediator which set of credentials he is willing to submit. Additionally, due to heterogeneity, the formats of the credentials currently at the client's disposal may not be accepted by some of the possible sources. In this case, the client would like to be assisted in getting reformatted credentials from some of the TTPs. For these and similar tasks, the mediator has to be able to resolve all kinds of heterogeneity among the security policies of the sources. Thus the characteristic services of mediators with respect to pure query answering should be extended to dealing with security policies as well. Moreover, the mediator, having its own mediator schema and its own local data and possibly also materialized data from previous queries, could have its own mediator security policy. Surely, such a mediator security policy must be suitable to integrate appropriate views on the various security policies of the sources. For this purpose, the mediator security policy should be considered as part of an extended mediator schema, and accordingly it should be declared during the preparatory phase. Furthermore, whenever a source is contracted to participate in the mediated information system, an appropriate security wrapper has to be constructed from the given mediator policy and the source policy.

Apparently all these and other related issues could be treated in many different ways. We argue that exploiting features of object orientation and of role based evaluation control are most promising. Object orientation is used for a unified view of all parts of the information system, and for providing appropriate granularities of controlled units. Role based control is selected for being application oriented and for its potential to integrate and unify various policies. Finally evaluation control is meant to combine aspects of access control, to be exercised mainly when invoking an operation, and of information flow control, to be exercised mainly when returning the result of a (nearly) completed operation. Proposing a specific object oriented role based evaluation control model is beyond the scope of the present paper.

8.4.5.4 Integration of local answers - functionality versus confidentiality. As already observed in Section 8.4.4, during the delivery phase of the protocol for mediated and secure query answering we are faced with the problem that the following requirements may be conflicting: Firstly, the mediator has to integrate and possibly materialize the local answers, sent back to the mediator by the sources to be finally delivered to the client. And secondly, the mediator should not be able to break the security policies of the sources. In particular, ideally the mediator should not gain meaningful information from the partial answers.

We discuss several solutions to this problem. They vary in two parameters: the achieved functionality for integration and the required trust in the mediator necessary to keep partial answers confidential.

Pessimistic solutions. These solutions follow the specification as given in Section 8.4.4. Here the mediator operates on the ciphertexts only, and thus
no trust in the mediator is necessary. Without any provisions, the achieved functionality for integration will be rather low. Essentially, the mediator can only annotate and forward the local answers. The functionality for integration can be improved if the mediator causes all sources to uniformly use a privacy homomorphism \cite{23, 30} for encrypting their local answers. Such a privacy homomorphism allows a subset of typical database manipulations on ciphertexts to be carried out as if they were executed on plaintexts. In order to employ a privacy homomorphism the mediator instructs all relevant sources to use an appropriate encryption method and the same session key. As discussed before, there are no essential limitations in doing so. Of course, in this situation the encryption method should be asymmetric because otherwise we could not guarantee confidentiality among the sources.

**Optimistic solutions.** These solutions allow the mediator to observe the local answers as plaintexts. Then the mediator can operate on local answers without any restrictions, but sources and clients have to put their trust on the mediator, at least to some extent. Surely, once the mediator has observed plaintext answers, the sources cannot technically enforce correct usage of the information gained. The best they can achieve is to bind the mediator to fixed obligations. Later on they can try to somehow supervise the behaviour of the mediator, and to blame the mediator for detected misuse. The following modification of the delivery phase is suitable for this purpose.

In step e) of the protocol, before sending local answers back to the mediator, the source performs the following actions: It *fingerprints* the copy of the data to be delivered such that later on that copy can be identified as devoted to the specific mediator \cite{22}. It attaches binding *approvals* to the data. An approval of form \((\text{distribute, oid, S, M, C})\) roughly states that "source \(S\) allows mediator \(M\) to distribute the content of (the object identified by) \(oid\) to client \(C\)" , and it is digitally signed by the source. And it *encrypts* the data using a public encryption key of the *mediator*.

In forthcoming disputes, the source can use the fingerprints to prove that the mediator has been delivered the data, and the mediator can use the approvals to prove that it has been allowed to further distribute the data.

However, not all possible problems are solved. Whenever later on the source claims that some further participant illegally holds a copy of the delivered data, then that copy may originate either from the mediator or the client who has issued the global query. The last case is also a problem without mediation. The new problem of mediation is to discriminate between misbehaviour of the client and misbehaviour of the mediator.

At the expense of additional security overhead all problems could be resolved with the same techniques sketched above, namely fingerprinting and approvals, now specific for the client (instead of specific for the mediator).

**8.4.5.5 Materialization of local answers.** Once the mediator has got local answers from the sources, it could materialize that data in order to reuse
it for further queries. Obviously, on the one side materialization raises new variants of the old problems concerning functionality, confidentiality, trust and claim of origin. But on the other side it could increase the overall efficiency of the mediator. A full treatment of all details of the interdependence of efficiency and security in the context of materialization is beyond the scope of this paper.

8.5 CONCLUSION

This paper has discussed the requirements for secure mediation, and it has presented the overall design and various advanced features for meeting them. A more detailed analysis and further topics for multimedia applications as well as promising areas of additional research and system development are sketched in the preproceedings and will be treated in more depth elsewhere.

Acknowledgments

We wish to thank Gerrit Bleumer and the anonymous reviewers for their valuable suggestions about draft versions of this paper.

References


VI Information Warfare Defense
Abstract: In spite of all existing security mechanisms, it is quite difficult to protect databases from electronic attacks. This research provides techniques to make an assessment of the damaged data and then to recover the affected data to consistent states after an attack is detected. Damage assessment is done using data dependency approach in order to obtain precise information on the damaged part of the database. Two algorithms are presented in this paper. The first algorithm performs the damage assessment and recovery simultaneously; whereas the second algorithm separates these two processes for improved efficiency. Both algorithms allow blind- writes on data items allowing damaged items to be recovered automatically.

9.1 INTRODUCTION

With the increasing popularity of Internet, worldwide information sharing becomes a common practice. At the same time, this connectivity with the rest of the world opens channels for intruders to access and possibly damage sensitive information. Although there are several techniques available, as described in [1] and [4], to prevent unauthorized access to sensitive data, these preventive measures are not always successful. It seems extremely hard to build systems that share information over electronic networks and still remain invulnerable to attackers. Hackers are always in search of new ways to prevail over the system security. Password sniffing and session hijackings are among various other means of intruding into a system, and the system will not be able to detect an attacker from a legitimate user in these cases. Besides, there remains possibility of significant damage by insider-turn-foes.
The productivity of any organization heavily depends on the information it shares with and protects from the rest of the world. An attack on an organization's information resources can have significantly devastating impact on the ability of the organization. Such an attack through electronic media is called Information Warfare. Defensive information warfare consists of three major phases: prevention, detection, and recovery from attacks. Various preventive measures to protect the databases from intruders in a defensive information warfare environment have been discussed in [1] and [4]. There are several ways to detect an intrusion into a system. Of these, a statistical approach has been discussed in [7] and a knowledge-based approach has been offered in [9]. Storage jamming [10] and [11], a method for misleading attackers to access fake data, can also be used to detect the intrusion. A more detailed discussion of intrusion detection techniques can be found in [8]. Pattern matching against known process of attack, examination of statistical profiles, inspection of known values of data, for example, are few others of these methods. Sometimes an attack may go unnoticed for a while, and as a result, the damaged data may spread and corrupt other undamaged data through other users. For example, a legitimate user may use the value of a corrupt data and update several other uncorrupted data based on the value read. This can have a cascading effect over time. Therefore, it is of utmost importance that the database be reconstructed by repairing the damage as soon as an attack is detected.

Traditional recovery methods [2], [3], [5], [6] fail to provide the integrity and efficacy needed to react to the situation under consideration. These issues are discussed in the next section. The objective of this research is to make an exact assessment of the damaged data when an attack is detected and then recover the affected data to a consistent state. As hindering the activities of other users of the system may be the intention of the attacker, it is desirable that the recovery process must bring the system back in real-time, while maintaining system integrity to the maximum extent possible.

In section 2, we examine various recovery methods and their shortcomings in defensive information warfare environment. Section 3 introduces our recovery model. A graph-based approach to the damage assessment is introduced in section 4. The algorithms are presented in section 5. Section 6 offers the conclusion of this research.

9.2 MOTIVATION

Conventional recovery algorithms presented in [2], [3], [5], [6] use a log to register each write operation of a transaction. During a system failure, the effects of all write operations of non-committed transactions that are already written into the stable database are undone. Furthermore, the effects of all write operations of committed transactions are redone if they are not in the stable database. This guarantees the integrity and consistency of the database. The log is also temporarily purged when it is determined that the stable database reflects the updates of all committed transactions (thus requiring no redo) and no effects of any of the non-committed transactions (thus requiring no undo).
The recovery method does not require any read operations for any of the transactions. The transactions are also never stored entirely since only the before images and after images are required during the undo and redo processes.

The above approach does not work when there is a malicious transaction that has already updated few data items and has committed. The system treats the attacker as any other valid transaction and makes the update permanent. This is guaranteed by the ACID properties (Atomicity, Consistency preservation, Isolation, and Durability) of transactions. Whenever the attacker is detected, all the updates of the attacker must be undone including the updates of the transactions that directly or transitively read from the attacker. Then, these valid transactions must be re-executed to return the database to a consistent state. However, this is not possible for the following two reasons. First, as the log does not store the read operations, the read-from relationships can not be determined. Secondly, since the transactions are never stored entirely, redoing the valid transactions is impossible. Therefore, it is necessary to update the log to store all operations of each transaction in the log. Nevertheless, it is not efficient to re-execute all transactions from the point of attack. For example, if an attack is detected after a month of its occurrence, it requires significant amount of time to undo and then redo all the transactions that have directly or indirectly read-from the attacking transaction. Besides, the system remains unavailable to users during the recovery process and thus yields to denial of service. In most military and some commercial applications, denial of service is highly undesirable. Therefore, development of an efficient algorithm to recover the system from electronic attacks is quite essential.

9.3 RECOVERY MODEL

As stated earlier, we assume that the database has been attacked and the attacking transaction has been detected. The method of detection is beyond the scope of this paper. This research is based on the following additional assumptions: (1) the scheduler produces a strict serializable history, (2) the log stores all operations of each transaction, and the order of operations in the log is the same as that in the history, and (3) the log is not modifiable by users (so that an attacker will not be able to damage the log).

**Definition 17** A write operation $w_i[x]$ of a transaction $T_i$ is dependent on a read operation $r_i[y]$ of $T_i$ if $r_i[y]$ must be scheduled before $w_i[x]$.

**Definition 18** A data value $v1$ is dependent on another data value $v2$ if the write operation that wrote $v1$ was dependent on a read operation on $v2$.

Since the log does not reflect all partial orders among operations in the history, the exact dependencies are hard to compute from the log. However, as per our requirement, the modified log does maintain the order of all conflicting operations. So, it is safe to consider that a write operation of a transaction depends on all read operations of the same transaction that precede the write operation. Assume that $S$ is the set of read operations of a transaction, say
$T_i$, on which a write operation, say $w_i[x]$ of $T_i$ depends on. But scanning the log, we find that the set of read operations, say $S_1$, that appears before $w_i[x]$, is a superset of $S$, i.e., $S \subseteq S_1$. From $S_1$, we would be able to recalculate the value of $x$ although some of the operations in $S_1$ will not be needed in the calculation.

Consider the history $H = r_1[a] r_1[b] w_1[c] r_2[a] w_2[b] w_2[d] c_2 r_3[d] r_3[a] c_1 r_3[c] w_3[d] c_3 r_4[b] c_4 w_5[a] w_5[b] c_5 r_6[b] w_6[b] r_6[c] w_6[c] r_6[d] w_6[d] r_6[a] w_6[a] c_6$. As per the relaxed definition, $w_1[c]$ in history $H$ is assumed to be dependent on $r_1[a]$ and $r_1[b]$. For that reason, the value of $c$ written by $T_1$ is predicted to be dependent on the values of $a$ and $b$, although it may really depend either on $a$ or on $b$.

**Notation** Let $p = T_n(q_1, q_2, \ldots, q_k)$ denote that the value of $p$ written by $T_n$ depends on values of $q_1, q_2, \ldots, q_k$ read by $T_n$.

Scanning the history $H$, we get the following dependency of values: $c = T_1(a, b), b = T_2(a), d = T_2(a), d = T_3(a, c, d), a = T_5(), b = T_5(), b = T_6(b), c = T_6(b, c), d = T_6(b, c, d)$, and $a = T_6(a, b, c, d)$.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$c$</td>
<td>$d$</td>
</tr>
<tr>
<td>$T_1(a, b)$</td>
<td>$T_2(a)$</td>
<td>$T_3(a, c, d)$</td>
<td>$T_5()$</td>
</tr>
<tr>
<td>$T_6()$</td>
<td>$T_6()$</td>
<td>$T_6(b)$</td>
<td>$T_6(b, c)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$T_6(b, c, d)$</td>
</tr>
<tr>
<td>$T_6(a, b, c, d)$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9.1 Data Dependency and Damage Assessment in $H$.

Table 1 shows the above data dependencies in $H$ as they occurred over time. Let us assume that $T_2$ has been determined as the attacker. Therefore, $b$ and $d$ are concluded to be damaged. Later, when $T_3$ writes $d$ after reading it, $d$ continues to be damaged. As the value of $b$ written by $T_5$ is independent of any contaminated data, $b$ has been refreshed. $T_6$ has read and updated $b$ and $c$ prior to reading the damaged data $d$. Therefore, $b$ and $c$ remain fresh. Likewise, $d$
still remains damaged and, furthermore, \( a \) is damaged as \( w_6[a] \) is dependent on \( r_6[d] \).

**Definition 19** A write operation is called a valid write if the value is written by a benign transaction and is independent of any contaminated data.

A valid write on a damaged data refreshes the data. Table 1 also shows how damage has spread and/or refreshed over time. The operations have been underlined to illustrate the damage. Note that the write operation of \( T_6 \) on \( a \) may be independent of the value of \( d \), and in that case, \( a \) will not be contaminated. This dependency can only be determined from the semantics of the updating query that is not taken into consideration in this work. For the simplicity of the recovery process it is safe to assume that \( a \) depends on \( a, b, c, \) and \( d \), i.e., 

\[
    a = T_6(a, b, c, d).
\]

### 9.4 DAMAGE ASSESSMENT

A graph based approach has been used in this section to observe how damage has spread through the database and which damaged items are refreshed via valid-writes. The graph we use for this purpose is a directed graph. A node in the graph represents a new value of a data item at a given time, and contains information such as the data item name, time of update, and a boolean value indicating whether the data is contaminated. For simplicity, we symbolize each node by either a circle or by a square with the data item name inside it. A circle denotes a clean data item while a corrupted data item is denoted by a square. Each edge represents an update by a transaction that either corrupted the data, or transmitted the damage, or refreshed the data. The nodes for any particular data item are drawn vertically below one another to specify the order among them with respect to the time of update.

![Damage Assessment Diagram](image.png)

**Figure 9.1** Damage Assessment in \( H \) using a Directed Graph.

The graph starts with all circular nodes, one for each data item in the database. Whenever an attack is detected, a square-type node is created for each data item that is updated by the attacker. An edge is added to each new node from the initial node that represents the same data item. The edge
carries the identification of the updating transaction. For each update in the log that depends on the damaged data, a square-type node is created and an edge is added to the new node from the node(s) on which the update depends. Moreover, whenever a transaction performs a valid-write operation on a damaged data item, a circular node is created for the item and an edge from the previous square-type node of the same item is added showing that the damage has been repaired. Thus, we have three types of edges in the graph: the edges that denote the initial corruption of data by attacking transactions, the edges that show transmission/continuation of damage, and the edges that indicate the refinement of damaged data through valid-writes. Note that no other valid-writes appear in the graph except for those representing the edges of the third type. Figure 1 shows the graph built from history \( H \). The graph depicts that items \( b \) and \( d \) were originally damaged by transaction \( T_2 \) (attacker). Later \( b \) has been restored through a valid-write by \( T_5 \). However \( T_3 \) and \( T_6 \) carried on the damage on \( d \). Moreover, \( T_6 \) also contaminated \( a \) by updating it after reading \( d \).

Of all the data dependencies derived from \( H \), only the following contribute to the spread and recovery of the damage: \( b = T_2(a) \), \( d = T_2(a) \), \( d = T_3(a,c,d) \), \( b = T_5() \), \( d = T_6(b,c,d) \), and \( a = T_6(a,b,c,d) \). Also observe that exactly these dependencies are represented by the edges in the graph.

Figure 9.2 A More Complex Graph on Damage Assessment.

Figure 2 displays a more complex graph that is obtained as an extension of the graph in figure 1. These extensions are based upon additional database transactions. Note that at the end, only item \( b \) is detected as damaged. During
recovery, only those transaction operations that contributed for this damage should be recomputed.

9.5 THE ALGORITHMS

We present two algorithms here. The first algorithm performs damage assessment and recovery simultaneously; whereas the second algorithm separates damage assessment and recovery processes for improved efficiency. As stated earlier, the modified log stores all read operations of every transaction. This is necessary to determine dependencies of operations. Although the value read can be determined from the after image of the previous write operation on the same data item, for optimization reasons, the value read may as well be stored along with the read operation. In the following algorithm, fresh_list and read_list_Ti are lists of records with two fields: data item field and value field. The structure damage_item_list includes the list of damaged data as concluded by the algorithm. The value of each damaged item in the damaged_item_list that would have been in the database if the attacking transaction had not been executed, is calculated and stored in the fresh_list along with its associated data item field. Read_list_Ti contains data items and values read by Ti.

Notation Let \([Ti, x, v1, v2]\) denotes the write operation of Ti in the log where v1 and v2 are respectively the before and after images of x. The read operations of Ti are denoted by \([Ti, x, v]\) which indicates that the value of x read by Ti is v.

This notation of representing a write operation of a transaction has been used in [3]. We present our first algorithm on damage assessment and recovery below. This algorithm assesses the damage and calculates the fresh value of the damaged data simultaneously. Note that, as mentioned earlier, the algorithm is based on the assumption that one or more transactions have been identified as attackers. Perhaps the intrusion detection mechanism, or an human analyst checking the values of known data items in the database, or some other means helped in the identification process.

9.5.1 Algorithm 1

1. Set damage_item_list = \(
\emptyset \); /* Empty set */
2. Set fresh_list = \(
\emptyset \);
3. Scan the log until the end
   3.1 For every write operation \([Ti, x, v1, v2]\) of an attacker, if \(x \notin damage_item_list\), add x to damage_item_list; and add the record \((x, v1)\) to fresh_list; /* v1 is the before image of x */
   3.2 For any other transaction \(T_j\) appearing after a write operation of the first attacker, set read_list_Tj = \(
\emptyset \);
   3.3 For every read operation \([Tj, x, v]\) of \(T_j\), add record \((x, v)\) to read_list_Tj;
      3.3.1 If \(x \in damage_item_list\), replace value v of record \((x, v)\) in read_list_Tj by value of x in fresh_list;
3.4 For every write operation \([T_j, x, v_1, v_2]\) of \(T_j\),
3.4.1 If the set of data items in read_list\(_{T_j}\) \(\cap\) damage_item_list \(\neq\) \(\emptyset\),
recalculate new value \(v_2\) of \(x\), by using values in read_list\(_{T_j}\);
3.4.1.1 If \(x \in\) damage_item_list, replace value of \(x\) in fresh_list by
new value \(v_2\);
3.4.1.2 Else
   add record \((x, v_2)\) to fresh_list; add \(x\) to
damage_item_list;
3.4.2 Else
   3.4.2.1 If \(x \in\) damage_item_list, remove \(x\) from damage_item_list;
      remove the record of \(x\) from the fresh_list;
4. For each item in damage_item_list, replace its value in the database by the
   value in the fresh_list.

Once the list of contaminated data is determined, all data items in the
list are blocked from being read by other transactions. This will stop further
spread of the damage in the database. However, while recovering the damaged
data, overwrites on them by any active transaction must be allowed. Such an
overwrite will be a valid-write because no damaged data is allowed to be read.
This option will refresh the damaged values. Once a damaged data is refreshed
through the recovery process or through a valid-write, the data can be made
available for read/write purposes. Next we explain some of the steps in the
above algorithm.

Since the attacking transaction may not be the first one of this type, there
is a possibility that the data item updated by this particular transaction may
have been damaged through a previous attacker. Therefore, in step 3.1, the
damage_item_list is checked to see if \(x\) is already there. In that case, neither
the damage_item_list nor the fresh_list need to be updated. Since the fresh_list
should have a correct value of \(x\), that value is the before image for this update
and must be left there. However, if \(x\) was not damaged previously or was
damaged but has been refreshed, then the before image of \(x\) is the correct
value of \(x\). The value \(v_1\) in item \([T_j, x, v_1, v_2]\) of the log is the before image
and, therefore is inserted into the fresh_list. When \(T_j\) reads a damaged data \(x,
\) in step 3.3.1, the correct value of \(x\) that is stored in the fresh_list (see theorem
2 below) is appended to the read_list\(_{T_j}\) for correct calculations of any future
updates made by \(T_j\). In step 3.4.1, \(v_2\) is the correct value that should have been
in the database if the attacking transaction was not executed. To recalculate
the new value of \(x\), we need to know the logical operation, say \(o\), that was
initially used to calculate the after image of \(x\). The information on this logical
operation, \(o\), may not be derived from \(v_1\) and \(v_2\) only. There are several ways
to solve this problem, one of which would be to embed the operation, \(o\), in the
log, for example \([T_j, x, o, v_1, v_2]\). Step 3.4.2.1 will be executed only when the

\(^1\)Refer to explanation.
transaction has not read any damaged data but blind-wrote a damaged data. This operation refreshes the damaged value of the data. So the data item is removed from the damage.item.list and from the fresh.list.

9.5.1.1 Proof of Correctness of the Algorithm. The following lemmas and theorems prove that the above algorithm recovers the database to a consistent state that should have been there if the attacking transactions were not executed.

**Lemma 1** Every data item that has been updated by an attacker is added to the damage.item.list.

**Proof** It is obvious from step 3.1 of the algorithm.

**Lemma 2** Every data item that (non-transitively) depends on a damaged data written by an attacker is added to the damage.item.list.

**Proof** Assume that a data item $y$ that directly (non-transitively) depends on $x$ where $x$ is updated by an attacker $T_i$. Now consider the transaction $T_j$ that read $x$ and updated $y$. Therefore, the order of $w_i[x], r_j[x], w_j[y]$ in the history is $w_i[x] < r_j[x] < w_j[y]$, and the operations also appear in the same order in the log. When $[T_i, x, v1, v2]$ is found in the log, by lemma 1, $x$ is added to damage.item.list. Step 3.3 of the algorithm, adds $x$ along with the value read to the read.list.$T_j$ as $[T_j, x, v]$ is detected in the log. Hence, when $[T_j, y, v1, v2]$ is discovered, by then $x$ already belongs to both damage.item.list and read.list.$T_j$. Therefore, step 3.4.1. will find that $x \in$ the intersection of these two lists and then step 3.4.1.2 will add $y$ to the damage.item.list.

**Theorem 1** A data item $x \in$ damage.item.list iff $x$ is damaged.

**Proof** If $x$ is damaged by an attacker, by lemma 1 $x$ is appended to the list. Similarly, by lemma 2, if $x$ is damaged after being updated by a non-attacker who read a damaged data written by an attacker, $x$ will be added to the list. There remains a case to discuss when the write operation on $x$ is transitively dependent on a damaged data written by an attacker, but $x \notin$ damage.item.list. In this case, find first such data $x_1$, written by $T_i$, that is not in the damage.item.list, but the damaged data on which $x_1$ depends, say $x_2$, is in the damage.item.list. Note that $x_1$ may not be different from $x$ itself. Following a proof similar to as in lemma 2, $[T_i, x_2, v]$ will appear in the log before $[T_j, x_1, v1, v2]$. Therefore, $x_2$ will be in both read.list.$T_i$ and damage.item.list when $[T_i, x_1, v1, v2]$ is found in the log and thus, $x_1$ must have been added to the list at this point as performed by step 3.4.1.2. This proves our above assumption of the existence of such an $x_1$ to be false. Therefore, every damaged data $x$ will be added to the damage.item.list. Moreover, $x$ will remain in the list unless $x$ is removed in step 3.4.2.1. This step will execute only when the transaction that wrote $x$, has not read any damaged data. In this case, $x$ is refreshed and no longer remains damaged.
It remains to prove that no non-damaged item will be in the damage_item_list. Again, assume that $y$ is such an item and was last written by $T_k$. Then $T_k$ is neither an attacker, nor it had read any damaged data before writing $y$. Therefore, even if $y$ was in damage_item_list, it must have been removed from the list in step 3.4.2.1. This contradiction proves the second part of the theorem.

**Lemma 3** If a data item $x \in$ damage_item_list then there is exactly one $(x, v)$ pair in the fresh_list.

**Proof** For every addition of $x$ to the damage_item_list (in steps 3.1 and 3.4.1.2), there is exactly one $(x, v)$ pair added to the fresh_list. The $(x, v)$ pair is not added to the fresh_list anywhere else in the algorithm. Similarly, $x$ removed from the damage_item_list only in step 3.4.2.1, and right there the $(x, v)$ pair is also removed from the fresh_list. In no other steps $(x, v)$ pair is removed from the fresh_list. This proves the lemma.

**Definition 20** A value $v$ is called the correct value of $x$ if $v$ would have been the value of $x$ in the database in the absence of any attacking transaction.

**Theorem 2** The value $v$ in the $(x, v)$ pair in the fresh_list is the correct value of $x$.

**Proof** Considering theorem 1 and lemma 3 together, it is clear that for every damaged item $x$ in the database has exactly one $(x, v)$ pair in the fresh_list. It remains to prove that $v$ is the correct value of $x$. This is proved by induction as follows.

Assume that all the data items in the damage_item_list are the result of updates of the first transaction that did the damage. Obviously, the transaction is the attacker. As per lemma 3, each of these damaged data also appear in the fresh_list. As per step 3.1 of the algorithm, for every $(x, v)$ pair, $v$ is the before image of $x$, and this before image is the after image of a valid write by the last transaction that wrote $x$ before the attacker. This value should have been in the database if the attacking transaction was not executed, i.e., the fresh_list starts with all correct values in all $(data \ item, value)$ pairs.

Next, we intend to show that for every addition of $(x, v)$ to fresh_list, $v$ is the correct value of $x$. There are two cases where such a pair can be added to the fresh_list: when a non attacking transaction spreads the damage, and when another attacking transaction executes. Case 1 occurs if a transaction, $T_i$, reads a damaged data $x$ before writing any data $y$ ($x$ and $y$ may be the same). In step 3.3.1 of the algorithm, whenever $T_i$ reads a damaged data $x$, the value read is ignored and the value $v$ of $x$ in the fresh_list is inserted into the read_list.$T_i$. This value is the correct value of $x$ since the fresh_list, as proved in the previous paragraph, already contains correct value of $x$. When $T_i$ writes $y$, the new value of $y$ is calculated in step 3.4.1 from the read_list.$T_i$ which contains only correct values. Thus, the new value of $y$ is a result of a valid write and is correct. This value is replaced in step 3.4.1.1 (if $y$ is already in
fresh_list), or inserted into the list in step 3.4.1.2 (if \( y \) is not in the fresh_list). In case 2, another attacking transaction \( T_i \) updates \( x \) and therefore \( (x, v) \) is inserted into fresh_list. There are two possibilities here too. First, \( x \) may have been damaged by another transaction and hence \( (x, v) \) pair is already in the fresh list. In this case, the fresh_list is not updated and so continues to have the correct value of \( x \). Secondly, \( x \) may not have been damaged before. Therefore, as explained in the previous paragraph, the before image of \( x \) after \( T_i \) updates \( x \) is the correct value of \( x \) and is inserted into the fresh_list. This completes the proof.

**Theorem 3** The database state produced by this algorithm is the same as the state that would have been produced if there was no attack on the database.

**Proof** It is clear from theorem 1, lemma 3, and theorem 2 that every damaged data in the database has its correct value in the fresh_list. Since, step 4 of the algorithm replaces the value of every damaged data in the database by the value of the same in the fresh_list, the database will not have any effect of attacking transactions at all. Moreover, the data items that were either not affected by the attackers or were refreshed later on through blind-writes will have their correct values in the database and will not be modified during the recovery process. Therefore, none of the effects of any good transactions will be lost.

Although the previous algorithm will precisely detect all damaged data items and repair them, it will block all active transactions in the system until the recovery process is complete. During this process, a significant delay is expected due to the computation required to determine the valid values of all damaged data and also for the disk accesses needed to the log that keeps a copy of all committed transactions in the system. The next algorithm solves this problem by first determining the set of damaged data and then making non-damaged data available to active transactions in the system. This will make the unaffected part of the database operative while the recovery continues. As pointed out earlier, during the recovery process the damaged data are available for blind-writes. This further increases the availability of data in the system while the recovery is in progress.

It is possible to accomplish the above-mentioned goal by removing the fresh_list and the related calculation from the previous algorithm during the damage assessment phase. Then, during recovery, the new value of each damaged data can be calculated. In order to do this, however, the partial order of all transactions that have accessed damaged data is needed. The process involves a second reading of the log incurring more disk accesses. To resolve this problem, we use a different data structure, damage_audit_table, as described next.

The auxiliary structure damage_audit_table stores information about transactions that have either corrupted the data, or transmitted/continued the damage, or refreshed the data. Transaction records are appended to the table in the same order as they appear in the log and each record has four fields: transaction_id, data_written, valid_read, and invalid_read. The data_written field
stores only data items with values that are either written by an attacker, or data that are dependent on the damaged data, or the fresh value of a previously damaged data after a valid-write. Notice that all data items in this column also appear as nodes in the damage assessment graph and vice versa. While the valid_read field stores all non-damaged data with their values as read, the invalid_read field keeps the damaged data along with their values that are read by the transaction. The algorithm consists of two phases: a damage assessment phase and a recovery phase. We present the damage assessment phase next. Again, note that the following algorithm requires the identification of the attacking transaction(s) as assumed for the previous algorithm.

9.5.2 Algorithm 2.1 (Damage Assessment)

1. Initialize damage_audit_table = {}; and damage_item_list = {};
2. Scan the log until the end
   2.1 When an attacker, \( Ti \), is found, add a new record with the transaction_id of \( Ti \) into damage_audit_table;
   2.1.1 For every write operation \([Ti, x, v1, v2]\) in the log, add \((x, v1)\) order pair to data_written column of \( Ti \)'s record; and add \( x \) to the damage_item_list if it is not there;
   2.2 For any other updating transaction \( Tj \) appearing after a write operation of the first attacker, add a record for \( Tj \) into damage_audit_table;
   2.2.1 For every read operation \([Tj, x, v]\)
       If \( x \in \) damage_item_list,
           add \( x \) to invalid_read column of \( Tj \);
       Else
           add \((x, v)\) pair to valid_read column of \( Tj \);
   2.2.2 For every write operation \([Tj, x, v1, v2]\)
       If invalid_read column of \( Tj \) is \( \neq 0 \),/*\( Tj \) has spread the damage*/
           add \((x, v2)\) pair to data_written column of \( Tj \); and add \( x \) to the damage_item_list if it is not there;
       Else
           If \( x \in \) damage_item_list,
               /*\( Tj \) has a valid-write on damaged data */
               write \((x, v2)\) pair to data_written column of \( Tj \); and remove \( x \)
               from damage_item_list;
   2.2.3 If \([\text{Commit}, Tj]\) found,
       If both invalid_read and data_written columns of \( Tj \) are empty, remove \( Tj \)'s record from damage_audit_table
       Else If \([\text{Abort}, Tj]\) found,
           remove \( Tj \)'s record from damage_audit_table.

Once the damage_list is determined, all non-damaged data are made accessible to users while the recovery process continues. Moreover, users are allowed to make blind-updates on damaged data. Next, we present the recovery phase.
of the algorithm. This algorithm uses the damage_audit_table and the damage_item_list as input in determining the correct values of the damaged data.

9.5.3 Algorithm 2.2 (Recovery)

1. Scan records in damage_audit_table until the end;
   1.1. For every attacking transaction other than the first one,
      1.1.1 For each \((x,v)\) pair in data_written column, substitute \(v\) by
      the before image of \(x\);
      /* For the first attacker, the before image is already there */
      /* If multiple attackers have written \(x\) consecutively, the before
       image of \(x\) for each of these transactions is that of the first
       transaction */
   1.2. For every non-attacking transaction with non-empty invalid_read
      column
      /* These transactions have spread damage */
      /* Any non-attacking transaction with an empty invalid_read column
      in damage_audit table, has refreshed some data items and their
      records need not be modified */
      1.2.1 For every \(x\) in invalid_read column, scan the data_written column
      upward starting from the previous record in damage_audit table
      to find the first \((x,v)\) pair; /* the last update on \(x\) */
      add \((x,v)\) pair into the valid_read column; and remove \(x\) from
      invalid_read column;
      1.2.2 For every \(x\) in data_written column, calculate the value \(v\) of \(x\)
      using values in the valid_read column; and substitute \((x,v)\) in
      data_written column; /* \(v\) is the correct value of \(x\) */

2. For every \(x\) in damage_item_list, check the new log that has just been
   created while the recovery process was in progress;
   2.1 If \(x\) is not modified in the log, /* otherwise, the update on \(x\) is a
     valid-write */
     scan data_written column of damage_audit_table from bottom-up to
     find first \((x,v)\) pair, and substitute the value of \(x\) in database with \(v\).

9.6 CONCLUSIONS

The existing recovery algorithms are not designed to operate in an information warfare environment. This research offers recovery algorithms that restore the database to a consistent state by recomputing the affected operations of all benign transactions that follow the attacker. For this purpose, the transaction log is modified to store the read operations of all transactions in addition to their write operations. The first algorithm performs the damage assessment and recovery simultaneously while these two methods are separated in the second algorithm. The first algorithm requires that the entire system is brought to a halt until the recovery is complete. The second algorithm, which comprises of two phases, releases the unaffected part of the database soon after the damage
assessment phase is completed. This makes the system available to users while
the recovery process continues.

Acknowledgments

1. This work was supported, in part, by the US AFOSR Summer Faculty
Research Program.

2. The authors would like to thank Prof. Sushil Jajodia for his valuable
suggestions on this research.

References

[1] P. Ammann, S. Jajodia, C. McCollum, and B. Blaustein. Surviving In-
formation Warfare Attacks on Databases. In Proc. IEEE Symposium on

Recovery in Database Systems. Reading, Massachusetts: Addison-Wesley,
1987.


agement Systems Against Information Warfare Attacks. Technical report,
The MITRE Corporation, 1996.


S. Demurjian, and J. Dobson, editors, Database Security IX: Status and

In D. Spooner, S. Demurjian, and J. Dobson, editors, Proc. of the IEEE
Computer Security Foundations Workshop, pages 176–185, Kenmare, Ire-
land, 1996.
VII Multilevel Security
Abstract: This paper describes version management in the Secure TransActional Resources-Database System (*-DBS) currently being developed at Penn State. This system employs concurrency control based on a secure multiversion timestamp ordering protocol. Efficient version management is critical to the performance of such a system. This paper describes a method of version management that requires no trust, adapts effectively to skewed access patterns, provides access to any version with at most one disk access and supports tuple level concurrency control. Based on our implementation, we report on the performance of this method.

10.1 INTRODUCTION

Multiversion databases are used as part of the design for Secure TransActional Resources-Database System (*-DBS) project currently being developed at Penn State. This paper addresses issues related to secure and efficient version management. The methods developed are implemented and the results and insights obtained are presented.

Secure version management methods often assign high-level transactions smaller timestamps than those of low-level transactions forcing them to access older versions. This bias can lead to performance penalties on high-level transaction. We attempt to improve performance of these transactions and at the same time improve performance for transactions that update or modify databases at their own security level. In this paper, we propose a dynamic on-page caching scheme and an in-memory version directory that reduces and
improves the efficiency of I/O. The remainder of this section provides background in multilevel security and multiversioned databases.

10.1.1 Multilevel Security

A brief review of multilevel security (MLS) is presented here. An MLS policy consists of mandatory and discretionary portions. A mandatory security policy controls the flow of information based on the perceived trustworthiness of an individual while a discretionary security policy controls the flow of information based upon user identity. This paper considers mandatory security only. In systems enforcing multilevel security, objects represent elements of information and subjects represent active entities such as processes. Subjects and Objects are assigned security levels. The Bell-LaPadula model[3] provides a concrete method of enforcing mandatory access control policy. It defines allowable read and write accesses to data objects in the form of the simple security and *-property[3]. The simple security requires that a subject be allowed to read an object only if the security level of the subject dominate that of the object. The *-property requires that a subject only be allowed to write objects with security levels dominating its own. In our work here, we restrict this further, and only a allow a subject to write objects at its own level. An implication of this is that transactions accessing a database at a lower security level appear to the lower database as a query.

10.1.2 Multiversion Databases

Versions are retained not for the sake of satisfying temporal queries but for concurrency purposes. This type of versioning is called transient versioning [5]. This means that at startup the database is single versioned. After recovery, the database is single-versioned once again. In a multiversioned system, transactions are assigned a timestamp value when they enter the system. Each version maintains both the timestamp of the transaction that created it and the maximum of the timestamps of all transactions that read it. These timestamp values are called the write and read timestamp respectively. When a query, wishes to access a record, it is provided with the version having the largest write timestamp less than or equal to its own.

Versions are created by update transactions. The update operation results in the creation of a new version of the tuple with the appropriate fields modified. The previous version is also maintained. Versions thus created can be chained together as shown in Figure 10.1. The primary version has the largest write timestamp within the version chain. As can be seen, besides data, a read and write timestamp and a pointer to the previous version are stored with each version. This is the overhead required to implement a multiversioned element. For effective version management, it is essential to limit the storage requirements as far as possible.

The remainder of this paper is organized as follows: First, related work is presented in Section 10.2. Section 10.3 presents a new type of on-page caching
called dynamic on-page caching. Section 10.4 looks at efficient means for accessing a version and maintaining timestamp information necessary for concurrency control. Implementation issues are dealt with in Section 10.5. Experimental results obtained from the implementation are presented in the Section 10.6 followed by concluding remarks in Section 10.7.

10.2 RELATED WORK

Early multiversioned systems stored primary and secondary versions in two different database files [7]. The files containing the secondary versions is referred to as the version pool. However, in view of the inefficiencies arising from this storage arrangement, Bober and Carey suggested on-page caching [5]. In this approach, part of the version pool resides in the data pages of the main database itself.

On-page caching as suggested by [5] assigns a fixed portion of every data page to hold the cache. Consider for example, an update to a record. When this record is updated, the current primary version is copied into the cache before the new primary is created. If the cache is already full, garbage collection is attempted to remove versions in the cache that are no longer needed. Versions that would never be appropriate to transactions now or in the future can be deleted. If garbage collection is unsuccessful in freeing the required space, then a version in the on-page cache is chosen for replacement. This version is pushed to the version pool file thus creating space in the data page. Notice that a version is pushed when the on-page cache is full, not when the data page is full. One side effect of on-page caching is an improvement in data page utilization in the main database. This is because, more versions are made to reside in the main database itself using the space already available. The effect of on-page caches on performance is discussed in detail in [5].

Let us examine how a record is typically accessed using a B\(^+\)-tree. When a record has to be accessed by, say, a query operation, the location of the primary version is obtained from the leaf page in the B\(^+\)-tree. Then, starting from this primary version, the version chain is traversed until the appropriate version is found. Observe that retrieving each version in the chain can entail additional disk I/O. So, even after the leaf page is reached, retrieving the appropriate version might require several additional disk I/O operations. The access path to a version is determined by the storage organization of the database. On-page
caching tends to shorten the length of this path. For example, without on-page caching all secondary versions would reside in the version pool. However, even with on-page caching, overflow from on-page cache causes secondary versions to be pushed to the version pool. Alternative storage arrangements for faster access to a version were proposed in [6]. Three techniques were proposed and the performance of these techniques were evaluated using simulation. The method with the best overall performance was data page version selection (DP). In this approach all version information, including timestamp and version pointers are maintained along with the primary version in the same page in the data file. This ensures that any version can be accessed in at most two disk accesses. We propose to store this information in memory thereby reducing the number of disk accesses to one. In this regard, we assume a B+-tree with clustering index.

A feasibility study of multiversioned databases enforcing MLS was reported in [15]. The focus was on mechanisms to provide efficient access to multiple versions of data. In this regard, the authors studied in detail the storage and access costs associated with multiversioning. An analytical performance model was developed to predict the penalty of retaining earlier versions for the sake of queries and the model was validated using measurements from an experimental prototype. However, the model did not address on-page caching. It was assumed that all secondary versions were maintained in the version pool. Both [15] and [7] assume versions at the granularity of pages. In this work, we adopt a tuple-level granularity and maintain version location information in memory.

10.3 DYNAMIC ON-PAGE CACHING

In our scheme, no space is dedicated to an on-page cache. The size of on-page cache is allowed to grow dynamically to accommodate the workload requirements. Versions are pushed to the version pool only when the data page is full as opposed to the on-page cache as recommended in [5]. However, we still retain a version pool that would be used when a data page becomes full. Whenever a page becomes full, we check if any of the versions can be collected. If not, then the oldest version is selected for replacement. We adopt a write-one policy, i.e., only one version is written to version pool each time. Note that writing one version at a time to a version pool does not lead to an I/O for every replacement as buffering can be used. For recovery purposes, however, we utilize a separate log file. This would store the updates to the databases before the transaction commits. This makes it unnecessary to flush the versions pool writes to disk before committing a transaction.

Dynamic on-page caching allows the benefits of on-page caching described above to be more fully utilized. Results in [5] suggest that queries execute faster as the size of the on-page cache is increased. By allowing this size to be determined dynamically, we can accommodate secondary versions more efficiently. This method also adapts well to nonuniform access patterns. For pages that see little or no update activity, the portion of the page that otherwise would be set aside for the cache is available for primary versions. However, when a page
is updated frequently it becomes a hotspot [11]. In this case, with a fixed cache size, cache overflow would occur frequently and the performance benefits of on-page caching are reduced. To address this problem, the size of the dynamic on-page cache is controlled dynamically based on the update frequency of the page.

One important side effect of a dynamic on-page caching scheme is the capability to tailor the cache size to meet the needs of a known work load. For example, when the workload is dominated by update operations, then the cache size will be adjusted to accommodate a sufficient number of secondary versions. When the workload is dominated by sequential scan queries, a smaller cache size will result. This will tend to preserve locality among the primary versions and allow these types of queries to complete faster. Databases inevitably experience non-uniform access patterns resulting in the creation of hot spots in certain regions of the database. An inability to adapt the cache size as required will tend to reduce the throughput of the database system. Fixed size on-page caches behave as if there is no cache at all once they become full. This is because, every update causes some version to be pushed to the version pool. Hotspots can provide sufficient update activity to fill an on-page cache and lead to reduced effectiveness. Dynamic modification of on-page cache size can adapt to such non-uniform access patterns. Below, we present a strategy for controlling on-page cache size to address this problem.

Hotspots are characterized by rapid version creation. We propose to use the following measure to characterize the intensity of update activity.

\[ \text{hot}_\text{rate} = \frac{num\_vers}{curr\_timestamp - ver\_timestamp} \]

In the definition, \( num\_vers \) is the number of versions created, \( curr\_timestamp \) is the current timestamp, and \( ver\_timestamp \) is the time at which \( num\_vers \) was last set to zero. Thus, this measure approximates the update rate for the page. We mitigate the effect of a hot spot by splitting the corresponding B+-tree index page early. That means, a page that meets our criterion would be split even before it is full. This splitting is triggered when \( \text{hot}_\text{rate} \) reaches some predefined limit.

Splitting a page early causes updates to complete faster. This is because the updates are now distributed to the two pages that resulted from the split. This leads to less contention and thus higher throughput. Further, immediately after a page split, the amount of free space in the page increases to 50%. So, more versions can be accommodated. The positive effect of on-page caching on utilization of disk blocks is offset by splitting the page early. However, as we expect only a relatively small portion of the database will meet our criteria for a hot spot, this reduced utilization will only apply to a small portion of the database.

10.4 VERSION DIRECTORY
We propose storing the timestamp information and version pointers in memory. A similar idea for storing timestamp information in a single versioned system is proposed in [4]. After the tuple identifier for a key value is located using the index, we can use the in-memory structure to determine where the appropriate version resides with no additional I/O operations. In this scheme, at startup, the database is single versioned and all tuples have a default timestamp. At this moment, no timestamp information is required. Since all tuples have the same default timestamp, it need not be stored with each tuple. As updates and inserts occur, the version directory is used to store information about the version chains that are being formed. So, the version directory only needs to store information about version chains that do not have a default timestamp on the primary version. Thus, we assume that if information about a tuple is not available in the version directory, then it has a default timestamp. Note that the size of the version table is proportional to the number of active transactions and not the size of the database. This is because we only need to maintain information about versions that have been updated recently. From time to time, the default timestamp can be reset to a higher value. This allows us to collect some of the memory tied up in the table. When the default timestamp is changed, all versions with smaller timestamps can be removed from the version directory. We ensure that no active transaction exists with a timestamp below the default timestamp. Any such transactions are aborted. For more details refer to [14].

Storing version information in memory improves the performance of transactions at dominating security levels. We use a secure timestamp generator based on the protocol described in [10]. As explained earlier, due to the *-property [3], a database can only be queried by transactions at dominating security levels. Combined with our timestamp generation method this forces high-level transactions to access older versions. If the appropriate version for these queries resides in the version pool then it would require multiple disk I/O for retrieving that version. Using a version directory we can avoid this bias against high-level transactions and ensure that all versions can be accessed with at most one disk access.

We can think of the version directory as a more efficient method of storing and caching, in memory, the timestamp and version chain information. To see the advantage of this approach, consider the following example. Assume that tuples require, on average, 200 bytes of storage and that timestamps requires 16 bytes each. The two timestamps associated with each tuple amounts to an overhead of about 15%. Thus, each page in the buffer pool is only 85% effective. This is especially troubling when we realize that the majority of the timestamps are old enough to be replaced by a single default value. Thus by maintaining only those timestamps that are actually necessary, we reduce this overhead considerably. This leads to higher effective I/O rates, and a better use of memory.

A hash table is used for storing the version information. Hashing is done in such a way that all tuples lying in the same page would have their information
stored in the same hash chain. During garbage collection, versions in a page can be collected by looking at one hash chain.

10.5 IMPLEMENTATION ASPECTS

The design discussed above was implemented as part of the *-DBS project. The prototype is hosted on Distributed Trusted Operating System (DTOS) [12]. DTOS is an experimental prototype operating system developed at Secure Computing Corporation. It provides mechanisms to implement multilevel security on the CMU Mach Microkernel [1] [8]) and provides policy-based control over all Mach services.

Figure 10.2 shows the architecture of the Star-DBS prototype. The trusted components are shown shaded. The prototype adopts a client/server architecture. A transaction executing at a client begins by contacting a transaction manager (TM) which assigns it a timestamp. For details on the protocol governing secure timestamp generation refer to [10]. The TM provides timestamps to transactions at all security levels. The transaction then proceeds to make service requests to one or more resource managers (RM). When the transaction is complete, it contacts the transaction manager again to request that its work be committed.

Each RM is implemented as an untrusted subject performing operations on behalf of clients at a single level. The RM implements a restricted SQL-like RPC level interface (i.e., no nested queries, no aggregates, no sortby, and no support for groups). The RM makes pin/unpin requests to the buffer manager (BM) [2]. The buffer manager controls the movement of data between the persistent and volatile portions of the database for all security levels. It also coordinates logging with page flushes to enforce the write ahead logging (WAL) protocol[9]. Each RM is multithreaded allowing it to service requests from mul-
multiple clients concurrently. The log manager [13] writes uninterpreted undo/redo records to the log on behalf of RMs, writes commit and abort records on behalf of the TM and controls the flushing of log records to disk.

The designs described in this paper were implemented on the DTOS operating system and consists of approximately 5000 lines written in C. The role of this implementation is to act as a file manager within the architecture shown in Figure 10.2.

One of the important implementation challenges was the version table. The version directory is organized as a hash table. Each security level maintains an independent version directory for the versions residing at its security level. This version information is accessed by all transactions at the same security level as well as by transactions at dominating security levels. Thus, this involves realizing a logically single version directory with independent version directories at each security level. In our prototype, mandatory access control is enforced by a trusted component of the buffer manager. We utilize this component to allow high-level transactions to access version directories at lower levels. Each RM creates a file that holds the version directory for that level. Transactions at higher level thus retrieve the buffer containing the entry they wish to access. In case, a subject at the lower security level tries to pin this page in write mode, the page is copied to another buffer [2]. Retrieval and traversal through the hash list are done transparently through an interface implemented within the RM. This interface abstracts away the security related issues and provides functionality allowing all standard operations on a hash table. So, RMs need not explicitly do anything special to retrieve an entry in the version directory, even if it resides in another security level.

Our RM is implemented as an untrusted subject. A single version directory for all the RMs at different security levels would have required a trusted implementation. An untrusted implementation eliminates the need for formal security evaluation of the component and allows simpler prototyping.

10.6 EXPERIMENTAL RESULTS

The implementation provided a means to test the feasibility and performance of the ideas we developed. We were interested in the effect of not storing version information in stable storage on performance.

The first test conducted was to observe how the size of the on-page cache would vary if no limit was placed on its size. In particular we wanted to observe the variance of on-page cache size. A wide variance in on-page cache size across the database suggests that dynamic control will be effective. For this purpose, a database was populated with tuples whose key values were generated randomly with uniform distribution from the set 1, ..., 100,000. Then, tuples were chosen following a uniform distribution from this set for update. Selection is done with replacement so that one tuple may be updated several times during an experiment. On average, for every ten tuples in the database, one update operation was applied. This means the average size of a version chain is 1.1 versions. This value was motivated by results of a performance study described
Page size for the database was 4096 bytes. Tuple size was chosen to allow 30 tuples per page. Tuples were inserted until the database consisted of about 100 data pages. At the end of all insert and update operations, the number of primary and secondary versions in each data page was measured.

The distribution of on-page cache size is given by the histogram shown in Figure 10.3. On the x-axis is shown the size of the on-page cache as a percentage of the page size. The percentage of pages that have a particular cache size is shown on the y-axis. We repeated the experiment five times. For each experiment we compute the mean, i.e., $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_5$. We then calculate the mean and standard deviation for this collection of five samples. Assuming a normal distribution, we calculated 90% confidence intervals for the sample mean using the formula:

$$\left( \bar{X} - \frac{1.64\sigma}{\sqrt{n}}, \bar{X} + \frac{1.64\sigma}{\sqrt{n}} \right)$$

In the expression $\bar{X}$ represents the mean of the five sample means, $\sigma$ represents their standard deviation and $n$ represents the number of measurements (i.e., five). The mean cache size is 8.816% with a 90% confidence interval of (8.415, 9.218).

As can be seen the size of on-page cache in each page varies widely. This significant variation in the size of on-page cache makes it extremely difficult to predefine a particular size for the on-page cache. Also, a significant portion of the database is populated with pages which have no secondary versions at all. This is indicated by the number of data pages with zero on-page cache size. This shows that a significant number of elements experienced no updates at all and thus validates our assumptions that storing timestamps in the data page is not efficient.

Another test was devised to observe the savings in disk I/O due to the version table. The scheme we compare our savings against is Data Page scheme (DP) [6]. In this method, all of the version information is stored with the primary version. So, the number of disk I/Os required to retrieve the primary version would be one, and overall, the number of accesses needed to retrieve any version in the version chain would not be more than two. A database was created as discussed above. However, to remove the effects of dynamic on-page caching on the version table we set the maximum size of the on-page cache at 10% of the data page size. The same set of key values were used in both cases. Then, a query is run to execute a table scan over the tuples in the database. The disk I/O required when the version directory was used is measured and the disk I/O with DP is also measured. The mean savings obtained are 7.556% with a 90% confidence interval of (6.920, 8.192). With dynamic on-page caching enabled even more versions would reside in the data page itself and this would help to increase the savings in disk I/O.

To examine the combined effect of dynamic on-page caching and the version table, tests were conducted using a non-uniform access pattern. We characterize the amount of nonuniformity or access skew as $x\%$, implying that $x\%$ of access
requests are directed to $100 - x\%$ of the data elements in the database [11]. The database is divided into two parts, the first constitutes $x\%$ of the data items and the second represents the reminder ($100 - x\%$). With probability $\frac{100 - x}{100}$ a transaction accesses the first part. An element of this set is chosen based on a uniform distribution. Thus, for a 70\% Skew, 70\% of the accesses are to 30\% of the data elements. Our tests ranged from a uniform distribution (a skew of 50\%) to a 90\% skew.

So, for each level of skew considered, we measured the disk I/O that was saved by a query scanning the entire relation. Again, a database was created as described above. Only the updates to the database were skewed. Then a query was run in isolation when all the update and insert activity in the database was complete. The timestamp of this query is between the timestamps selected for updates and inserts. This means, if no update was applied to a version then the primary version is the appropriate version for this query. If an update operation was applied, then the secondary version is the appropriate version. The results obtained are shown in Figure 10.4. The difference between the disk I/O required for a query with a fixed on-page cache size of 10\% using DP, and the disk I/O for our design is expressed as a percentage plotted on the $y$-axis. For each access skew, the test was repeated five times and the average and variance are shown on the plot. The error bars represent 90\% confidence intervals.
As can be seen, as the amount of skew increases, the disk I/O saved also increases. This is because, due to the formation of hotspots, the pages are split earlier. This leads to more versions being held within the data page and hence a saving in disk I/O. Also, savings accrue due to the version directory that reduces disk I/O required when the appropriate version for the query resides in the version pool. As noted earlier, in a hot spot, the performance of fixed size on-page caches degrades. This effect becomes more pronounced as access skew is increased. This is because more versions are pushed to version pool. As disk I/O for retrieving versions in version pool is reduced with our scheme, the corresponding savings in disk I/O increases.

10.7 CONCLUDING REMARKS

We have presented a design for version management in a multilevel secure database system and described a prototype based on this design. In addressing the issues relating to storage of versions we found that a dynamic on-page caching scheme can effectively adapt to non-uniform access patterns. The version directory improves performance by reducing the overhead of maintaining version information. However, the version directory is constructed from a set of independent version directories each associated with a RM at that security level. This was done with support from DTOS and the buffer manager.
The combined effects of dynamic on-page caching and the version table show a reduction in I/O of between 32 and 47% over the DP method of [6].

References


11 SACADDOS: A SUPPORT TOOL TO MANAGE MULTILEVEL DOCUMENTS
Jérôme Carrère, Frédéric Cuppens, and Claire Saurel

Abstract:
This paper describes SACADDOS, a decision support tool to derive the sensitivity of a document when this document is transmitted, and to control the evolution of this sensitivity over time. For this purpose, SACADDOS manages a set of classification security policies. A classification security policy corresponds to a set of rules which are used to derive, from the content of a document, the classification of this document at a given time. SACADDOS includes an intelligent document management tool to analyze the content of a document in order to derive which classification rules apply to this document. When several contradictory rules apply, SACADDOS suggests to solve the conflict by defining an order of preference between the contradictory rules.

11.1 INTRODUCTION
Many organizations, especially intelligence services, have to manage huge numbers of documents, some of them which are sensitive. Generally, these organizations define security policies to derive a classification level from the content of a document. These security policies correspond to sets of rules, for example:

1. Documents in nuclear transport must be confidential.

2. Mission plans in weapons delivery for Bosnia must be secret.

3. Mission plans about hostage liberation in Bosnia must be top secret.

Moreover, sensitivity of a document may change over time. Generally, security policies manage the evolve in classification depending on the type of the document and content. These security policies define rules, for instance:
1. A document classified at the secret level needs to be downgraded after 10 years.

2. A document classified at the confidential level needs to be downgraded after 5 years.

3. An occasional mission plan must be downgraded the day after the completion of this mission.

These are examples of downgrading rules. These rules specify a reduction of document sensitivity. In addition, there are rules to specify that the classification of a document must be upgraded, for example:

1. In the case of a conflict with a particular country, every confidential corresponding document must be upgraded at the secret level.

Choosing the actual sensitivity of a document and defining how this sensitivity is to be changed in time requires an analysis of the documents content. When the number of documents becomes significant, determining their sensitivity becomes a long and tedious task. Therefore, in many organizations which have to deal with sensitive information, there is a clear need for a tool which provides automated capabilities for classification and downgrading of documents.

However, there has been little research done in this direction. The first system proposed by McHugh [6] provides only online assistance to a human to downgrade text. Another published paper on automatic classification/downgrading of text is by Lunt and Berson [5]. This paper describes Classi, an expert system to classify and sanitize text based upon content, context or information source. As noticed by [8], the main drawbacks of this system is that document analysis capabilities in Classi are only based on keywords and associations within a sentence. This approach does not provide sufficient understanding of “natural language” so that we can expect to obtain good result when automat ing the classification or downgrading of text.

Fortunately, for several years, “intelligent” support tools for document management have been designed. They provide functions to analyze both the syntactic and semantic content of a document. Such tools for instance, can automatically determine if a document is dealing with nuclear transport. This technique is sufficiently powerful to remove most ambiguities and to dynamically find main concepts in the text.

The purpose of this paper is to present a tool called SACADDOS. SACADDOS automatically determines the document sensitivity and controls evolution of this sensitivity over time. It is a decision support tool which suggests classification and downgrading choices to users. It also provides explanations for this choice by providing traces of derivation and enlightening the relevant portions

---

1 Common written or spoken language.
of the text. But, of course, users are always responsible for the final decision. The basic principle of SACADDOS is to combine a module for management of security policies used to classify and change the classification of documents, with a tool for document management.

The remainder of this paper is organized as follows. Section 2 presents the main objectives and functionalities of SACADDOS. Section 3 shows how knowledge, especially classification and downgrading security policies, are represented and managed in SACADDOS. Section 4 describes its logical architecture. Finally, section 5 concludes this paper by investigating several issues in this work.

Notice that it is not the purpose of this paper to investigate the problem of enforcing high security assurance when automating the classification and downgrading processes. This is an important but complex problem which represents further work that remains to be done.

11.2 Objectives of SACADDOS

The work presented in this paper applies to the context of multilevel security policies. In a multilevel security policy, every piece of information is associated with a classification level and every agent is associated with a clearance level. Classification and clearance levels are taken from a set of security levels associated with a partial order relation. The confidentiality property in the multilevel security policy states that an agent can only know a given piece of information if the clearance level of this agent is higher than or equal to the classification level of the information.

In the remainder of this paper, we shall consider four security levels: NC (public), CD (confidential), SD (secret) and TSD (top secret). This set of security levels is actually associated with a total order relation: NC ⪯ CD ⪯ SD ⪯ TSD. However, the work presented here also applies to the case of a partial order relation.

We shall consider that information is represented by the content of full text document in "natural language". The first problem to be solved by the transmitter of a document is to decide how to classify this document. For this purpose, we assume that some security policies have been defined to derive the classification of a document from the content of this document. Currently, classifying a document is performed by manually analyzing the document content and manually applying the classification security policies to this document. However, due to the ceaseless increase in the bulk of information and because of the existence of intelligent document management tools, it is necessary and possible to design a tool to provide agents with assistance in this classification process. This is the first objective of SACADDOS.

The classification of a document does not remain unchanged over time. After varying duration, the document content becomes obsolete and the document classification is to be downgraded. Therefore, after a document has been classified, the transmitter of a document has to fulfill the following tasks:
1. To choose a downgrading type for this document. In accordance with the security policies, there are two types of downgrading (see also [2]):

- **At a time:** this corresponds to a decision of downgrading the document classification at a specific time.
- **By order:** in this case, the document can be downgraded only if the transmitter of this document gives the order to downgrade it.

2. If the decision is to downgrade a document at a time, then the transmitter must specify the time at which the next downgrading will occur and what will be the new classification of the document at this time. This corresponds to the definition of planning to control the evolution of document classification over time.

As mentioned in the introduction, SACADDOS is a decision support system. When a user of SACADDOS has to decide how to classify or downgrade a document, SACADDOS can advise the user by suggesting classification or downgrading choices. However, a user is not obliged to follow the suggestions of SACADDOS. The user is not even obliged to ask SACADDOS for advice. In this last case, SACADDOS simply enables this user to manually classify or downgrade documents.

Let us now describe the main functions provided by SACADDOS.

**Inserting a document in the document base.** When new electronic sensible documents are created, they have first to be integrated in SACADDOS's document base. Related to each document, SACADDOS builds a description form with the documents main characteristics (for performance care); depending on the structure of the document, this form may or may not be filled up automatically. In some cases there needs to be an interactive help tool for the user. SACADDOS can then perform sensitivity management operations on documents.

**Classification.** Following classification policies, the first step is to provide the document with a classification level. Using the querying functionality of the document management tool, SACADDOS applies classification rules in accordance with the content of the document, in order to derive this classification level. When several different classification levels can be derived from the classification rules (this situation is called a conflict), SACADDOS can help to reduce this conflict by applying some strategies that can be combined within a meta-strategy by the user. The classification level finally approved or decided by the user is stored in the description form related to the document.

**Choice of downgrading type.** The downgrading type of a document is defined by the security policy according to the nature or content of the document. The process of choosing downgrading type is similar to the classification one: SACADDOS selects the applicable rules and suggests a conclusion with or without applying meta-strategies. According to the security policy we consider, the
selected downgrading type will imply different behaviors. If the downgrading type is "at a time", there is still a related downgrading planning to fill. If it is "by order", the process is completed.

**Definition of downgrading planning.** To start this process, it is required that the previous step of choosing document classification is completed (possibly manually), and that the downgrading type is set to "at a time" for the considered document.

The result of the classification process is a pair \( P_0 = (\text{initial classification date}, \text{initial classification level}) \). Starting with this pair, the downgrading planning process aims at defining another pair \( P_1 = (\text{first downgrading date}, \text{first downgrading level}) \). Then, the downgrading planning process iterates by computing a new pair \( P_2 \), and so on until the public level is achieved or there is no more rule that can be applied.

The characterization of conflict is different from the classification process. The pair corresponding to \((i\text{-th downgrading date}, i\text{-th downgrading level})\) has to be unique. Therefore, there is a conflict if it is possible to derive several different pairs from the set of applicable rules, that is pairs with different downgrading dates or pairs with different downgrading levels or both.

**Management of downgrading.** Once the planning is built, the downgrading management consists in checking if the document is properly classified in accordance with the downgrading plan and to the current date. If not, the operator is asked for downgrading the document according to its planning.

Another aspect of downgrading management is to record downgrading orders when received and to update the classification level as specified by the order.

**Updates.** An update corresponds to a modification of the current classification level, the downgrading type or the downgrading planning. This update is due to new applicable rules, abrogated rules, or occurrence of events (since the occurrence of an event can activate existing rules and make them applicable to some given document). In order not to have obsolete data about document protection, SACADDOS finds for each action (e.g. inserting a rule, removing a rule, adding an event, deleting an event, modifying an event, etc.) which documents are concerned by the related modification.

### 11.3 KNOWLEDGE REPRESENTATION AND MANAGEMENT

#### 11.3.1 A logical framework

Here we present the language used to describe classification and downgrading security policies. Since we want to be able to both represent knowledge and to compute derived information, our language is based upon first order logic. Within this language, we will have to represent objects (e.g. documents), events (e.g. a conflict between two countries), and also operational rules (e.g. classification rules).
For objects and events, we take our inspiration from the object-oriented paradigm. In this framework, an entity is represented by an object. Any object belongs to a “class of objects” which is an abstraction of it. The set of classes is structured in a hierarchical way, for instance, a class may inherit from another class. By doing so, we can represent different abstraction levels. A class of objects is defined by:

- its name,
- the names of the classes it inherits from,
- its attributes; attribute values are objects (so that we can define structured objects).

In our logical language, a class of objects is represented as follows:

- name: a symbol of unary predicate.
- attribute: a symbol of binary predicate. The arguments of this predicate will respectively receive the identifier of a given object, and its corresponding value for the attribute.

A given object is identified by a constant (constants are denoted by capital letters) – different constants identify different objects. We represent any inheritance link between two classes of objects \( C_1 \) and \( C_2 \) with the following rule:

\[
\forall x, C_1(x) \rightarrow C_2(x).
\]

We call entity the highest class in the hierarchy of objects. This class entity is then split into two disjoint classes: object and event. So we have the following axioms:

- \( \forall x, \neg (object(x) \land event(x)) \)
- \( \forall x, entity(x) \rightarrow (object(x) \lor event(x)) \)
- \( \forall x, object(x) \rightarrow entity(x) \)
- \( \forall x, event(x) \rightarrow entity(x) \)

Class event is defined with the following attributes: reference, event type, beginning date, end date. This class is introduced in our language because many classification or downgrading rules can only be applied when some event occurs, for instance, at the end of a given mission, or at the beginning of a conflict. In this case, classes mission and conflict are to be defined as sub-classes of class event.

11.3.2 Classification and downgrading policies

The purpose of security policies we consider is to specify rules to classify and downgrade documents. Therefore, we define classes as follows.
- **Class Document** has attributes: title, reference, transmitter, content (full text in "natural language"), transmission date, downgrading type, sensitivity level, classification history, downgrading planning.

A document may be structured in parts, chapters or sections. Therefore, class **Document** also includes the following attributes: a set of contained Parts, a set of contained Chapters, and a set of contained Sections.

**Note:** The content of a given document is supposed to be invariant in our system. If the content of a document changes, then a new document having this new content is created.

Security policies are modeled as a sub-class of documents:

- **Class Policy** inherits from class **Document** and has the following attributes: coming into effect date, set of contained Rules.

A policy may also abrogate other policies, or parts, chapters or sections coming from other policies. Therefore, class **Policy** also includes the following attributes: set of abrogated Policies, set of abrogated Parts, set of abrogated Chapters, set of abrogated Sections.

Finally, a policy defines a set of **rules**:

- **Class Rule** has attributes: name, transmitter, transmission date, coming into effect date, source (structural position in the policy which may contain the rule), content (original text in natural language), and logical expression (logical representation of the rule).

According to the policies we consider, rules can be sorted into three classes:

- Classification rules, for instance: "Documents which deal with nuclear transport must be confidential."
- Downgrading type rules, for instance: "A mission plan dealing with a cancelled mission must be downgraded by order."
- Downgrading rules, for instance: "A document classified at the secret level needs to be downgraded after 10 years." Such rules are used to planning the sensitivity evolution of documents which are to be downgraded at a time.

To be able to automatically derive which classification level and when it has to be assigned to a given document, we have to give logical representations to rules. Most premisses of rules are conditions either about the content of the document (especially about its themes), or about occurrence of events. In our logical language, we introduce the following predicates:

---

2This modeling choice makes it possible to apply the classification/downgrading process on the documents containing the security policies.
• \( \text{classif}(\text{doc}, t, n) \): the classification level of a document \( \text{doc} \) which was transmitted at time \( t \) has to be \( n \).

• \( \text{downgrading_type}(\text{doc}, t, c, \text{type}) \): the downgrading type of the document \( \text{doc} \) transmitted at time \( t \) with a classification level \( c \) must be \( \text{type} \).

• \( \text{downgrade}(\text{doc}, t, n, [t_1, n_1]) \): the document \( \text{doc} \) classified at level \( n \) at time \( t \) must be downgraded at level \( n_1 \) at time \( t_1 \).

• \( \text{deals_with}(\text{doc}, \text{theme}) \): the document \( \text{doc} \) deals with the theme \( \text{theme} \).

Since all classification and downgrading rules aim at giving only one conclusion, the logical representation of rules corresponds to Horn clauses.³

We need to define an adequate representation of time in order to express the notions of date (for instance, a document transmission date) and duration (for instance, “ten years”, “5 days”). Our temporal representation partly takes its inspiration from [4]. We model time as a linear and continuous time, in which we distinguish some particular points which delimit time intervals. Dates and durations are denoted by triples \( \langle \text{day}, \text{month}, \text{year} \rangle \). For instance, \( \langle 18, 2, 1998 \rangle \) denotes the date “February 18th, 1998”, and \( \langle 1, 3, 2 \rangle \) denotes a duration of two years, three months and one day.

We also define an additive function \( \text{add}_d \) between dates and durations (e.g. to compute the date corresponding to “ten years after the date of document transmission”). The algorithm of this function is specified to fit with our human intuitive way of computing such operations, especially when taking into account variable lengths of months and years. Due to the space limitation, we cannot develop this extended work here (see [3] for a more detailed presentation).

Using this language, we now give logical representations of several rules:⁴

• Documents which deal with nuclear transport must be confidential.

\[
\text{deals_with}(\text{doc}, \text{NUCLEAR.TRANSPORT}) \rightarrow \text{classif}(\text{doc}, t, \text{CD})
\]

• A mission report must be downgraded by order.

\[
\text{deals_with}(\text{doc}, \text{MISSION.REPORT}) \rightarrow \text{downgrade_type}(\text{doc}, t, \text{SD}, \text{ORDER})
\]

• A document classified at the secret level needs to be downgraded to the confidential level after 10 years.

\[
\text{add}_d(t, \langle 10, 0, 0 \rangle, t_1) \rightarrow \text{downgrade}(\text{doc}, t, \text{SD}, [\text{CD}, t_1])
\]

• An occasional mission plan must be downgraded the day after the end of the corresponding mission:

³ A Horn clause is a clause in which there is at most one positive literal.
⁴ The constants \( \text{SD}, \text{CD}, \text{NC} \) respectively represent the secret, confidential and public classification levels.
\[ \text{mission \_plan}(\text{doc}) \land \text{mission}(m) \land \text{end}(m, t) \land \\
\text{add} \_d(t, (1, 0, 0), t') \land \text{deals \_with}(\text{doc}, m) \\
\rightarrow \text{downgrade}(\text{doc}, t, l, [t', NC]) \]

### 11.3.3 Rules applicability

A rule can be transmitted alone, for example, by the security administrator of an intelligence service. It can also belong to a classification or downgrading policy, which has a transmitter, a transmission date and a coming into effect date (from which rules inherit). In both cases, a rule has a *validity period*: it can be applied only from its coming into effect date, to the date at which another policy or rule coming into effect will abrogate it.

Notice that a rule might concern documents which were transmitted before the rule coming into effect date. This means that a rule may come into effect retroactively, applying to documents which were transmitted before the coming into effect date of the rule (see the next subsection for an example).

Notice also that a rule may be abrogated. This does not mean that the rule is physically deleted. Actually, the rule is kept by SACADDOS, and SACADDOS implements a procedure to determine which rules apply at a given time, that is rules whose validity periods include this given time. When SACADDOS tries to perform a given derivation, it only considers this subset of applicable rules. This makes it possible to reason at different times, for instance past and future times.

In some cases, there is no applicable rule to classify or downgrade a given document. In such situation, SACADDOS provides the user with a “Don’t know” answer. It may also happen that an applicable rule includes reference to a not yet occurred event, for instance reference to the end of a mission. If, when the rule is applied, the end of the mission is unknown, SACADDOS provides the user with an “unknown” date (actually an existential null value).

### 11.3.4 Downgrade planning and classification history

To control the sensitivity evolution of documents over time, for which the downgrading type of a document is at a time, it is necessary to compute and then to store a downgrade plan. This plan matrix forecasts *when* and at *which* classification level a document has to be downgraded.

Nevertheless, since we design a decision support tool and not an automatic tool, a downgrading operation for a document will not necessarily be performed at the scheduled time, but sometimes later (e.g., scheduled time corresponds to vacation for the security administrator). So the downgrading dates for a document may actually not fit with the content of the downgrade plan. To be able later to know or reason about the history of the document sensitivity evolution another piece of information is necessary. This additional information is a classification history, which is another matrix similar to the downgrade plan, but with a content corresponding to the real evolution of classification. Note that ideally, downgrade planning and classification history should be identical.
182 DATABASE SECURITY XII

11.3.5 Update plan

At any given time, new applicable rules may lead to update an already computed downgrade plan:

- either a new rule comes into effect (e.g., a new classification policy) dealing with documents which were transmitted before the coming into effect date of the rule (see example 1 below),

- or a rule dealing with a given event which could not be applied to documents, because of the unoccurrence of the related event, can now be applied because of the actual occurrence of the event (see examples 2 and 3 below).

In both cases downgrade planning has to be updated since the new rules have to be taken into account. Here we give some examples of such updates.

1. Let us assume that the following rule R1 comes into effect on February 1998: every document transmitted before 1990 and dealing with Bosnia must be classified at the public level. Let us consider a document D dealing with Bosnia, transmitted at the secret level in 1989. Then, due to the general downgrading rules expressed in the introduction, this document first ought to be downgraded at the confidential level on 1999, then at the public level on 2004 (see figure 11.1). When R1 comes into effect, the classification level and the downgrading planning for D must be updated, although D was transmitted before February 1998. However, since we cannot change the actual past, this cannot affect the sensitivity levels the document actually got before February 1998. That is to say that document D may be, at best, downgraded in February 1998.

![Figure 11.1](image)

**Figure 11.1** Initial planning (solid line) and updated planning (dashed line)

2. Let us assume we have the two following rules R2 and R3. R2: A mission plan has to be downgraded at a time. R3: A mission plan which deals with a mission which has been cancelled has to be downgraded by order. For a mission plan which deals with a mission which turns out to be cancelled,
R3 must apply and will cause both an update (i.e., an adjournment) of the downgrading planning initially filled up with R2, and an update of the downgrading type.

3. Let us assume the rule: In case of conflict with a given country X, every confidential document about country X must be upgraded at the secret level. Let us consider an American document D, dealing with weapon delivery in Iraq, that was transmitted at the confidential level on January 1998; according to its initial downgrade plan and the rules presented in the introduction, it needs to be downgraded at the public level on January 2003. But, on February 20th, 1998, there is a conflict between USA and Iraq. So the planning must be updated; the classification of D must be upgraded to the secret level and the planning becomes: downgrade to the confidential level on February 20th, 2008, and then to the public level on February 20th, 2013 (see figure 11.2).

![Figure 11.2](image-url) Initial planning (solid line) and updated planning (dashed line)

11.3.6 Conflict resolution

Suppose that at a given time, for a given document, several different rules apply giving different conclusions. Since it is assumed that at any time a given document can have only one sensitivity level, only one downgrading type and if the downgrading type is on a time, only one downgrading planning, we call such a situation a conflict. Due to the semantic nature of data classification, this problem was already identified in [7] and [5]. We distinguish three kinds of conflicts: classification level, downgrading type, and conflicts about downgrade planning.

Depending on its nature, there are several ways of managing a given conflict. It consists in defining partial preference orders between rules:

- Conflict between specific rules, coming from a same policy. Suppose a policy which regulates mission plans, and in particular mission plans about
some sensitive countries, with different particular objectives (hostage liberation, weapon delivery...). A first strategy consists in preferring the conclusions derived from the rules whose premisses express the most specific conditions about document content.

- Conflict between rules coming from different specific policies. Suppose you have a general policy (including rules which specify downgrading after 5 or 10 years), and a policy concerning documents dealing with occasional missions. In this case, the second policy is more specific than the previous one, yet another strategy consists in preferring the conclusion coming from the most specific policy.

- Conflict between derived sensitivity levels: in some applications, a good strategy is to prefer to minimize security risk by choosing a document assignment level at the highest level.

- Conflict between downgrading dates for the same sensitivity level: another strategy consists in preferring to minimize security risk by choosing the latest date of downgrading.

- Conflict between “old” rules: one may prefer to choose the rule which came into effect most recently.

- Conflict between rules which were transmitted by an author with more or less high rank: one may prefer to choose the rule coming from the author with the highest rank.

- and so on...

As one can see, the preference order between rules can be defined upon the conclusions they give, and upon some intrinsic characteristics of the rules, which are stored in the rule descriptions (see section 11.3.2). So we have provided some corresponding basic strategies as the ones described below, which could be enriched by users. Since a strategy fits with a partial order defined on rules, its application aims at reducing the number of proposed conclusions for a given document, by preferring a subset of the set of applicable rules. In combining several strategies according to the choice of the user, we obtain a meta-strategy. Using meta-strategies, it becomes possible to gradually reduce the set of suggested conclusions and sometimes get only one solution. This conflict resolution provides a help to users who in some cases can suggest to add some rules specific to their application but not included in the rule base of SACADDOS.

Due to space limitation, we do not further develop the problem of conflict resolution in this paper, but see [1] for a more detailed presentation and formalization of this problem.

---

5Since SACADDOS is a decision support tool, users are free to choose to apply or not basic strategies in whichever order.
In summary, for a given document and a given functional operation of SACADDOS to be performed on this document, our process includes three consecutive steps:

1. selection of all rules for which the validity period includes the time of the evaluation,

2. from these rules, a selection of applicable rules for the given document (i.e., making all rule premisses true),

3. if these applicable rules give only one common conclusion, this defines the suggested result, if not, then strategies apply to reduce the set of suggested conclusions.

11.4 LOGICAL ARCHITECTURE OF SACADDOS

SACADDOS was designed as a client/server application. The whole core of SACADDOS runs on the server. On the client side runs only the Graphic user interface. The core of SACADDOS is composed of a knowledge base, a document database, a database for events management, and user profile descriptions. The graphic user interface runs independently from the server on a separate computer. This architecture requires us to define communication mechanism between the different modules, especially between the core of SACADDOS and the user interface.

11.4.1 SACADDOS module description

We first present the SACADDOS core and then the user interface. Each knowledge item considered in SACADDOS is represented as an object. The implementation is in PROLOG and keeps this principle of object knowledge representations.

11.4.1.1 Knowledge base management. The SACADDOS knowledge base includes: security policies, strategies and meta-strategies for conflict solving process. The SACADDOS knowledge base management includes processes for classification and downgrading management process for update management. The overall knowledge is managed with PROLOG.

11.4.1.2 Database for event management. Events are considered in SACADDOS as special objects that can influence the applicability of classification or downgrading rules. Ideally, the set of events may be managed by an object-oriented database management system. However, in the present prototype, they are simply simulated by a set of PROLOG facts.

11.4.1.3 Document database. The document database is managed by the document management tool. SACADDOS queries this tool to create new
document description forms\(^6\) which include all information required to apply classification and downgrading rules. For this purpose, the document management tool selects all the themes mentioned in the document and which are involved in the rules and sends them to SACADDOS. If they are provided in the document, SACADDOS can automatically fulfill other required attributes in the document description form such as the document type, reference, transmission date and transmitter. If the document management tool does not find some information in the document file, SACADDOS warns the user about this so that the user can manually insert the missing information.

**11.4.1.4 User profile management.** User profiles are introduced to manage different users that are playing different roles in SACADDOS with multiple clearances. User profiles are used to restrict access to classified documents or to any other classified item stored in SACADDOS. For instance, a user with clearance secret cannot have an access to top secret documents. We also assume that the document description form has the same classification as the document it comes from. SACADDOS performs access controls so that the user only receives information it is cleared to observe according to its user profile.

\(^6\)These forms are created for efficiency purpose.
11.4.1.5 Graphic user interface. We designed an OSF/Motif based graphic user interface that locks/unlocks functionalities according to the user profile. The user interface adapts to the users who have authorization to work in SACADDOS.

11.4.2 Communications between different modules

The communication protocol depends on the modules to be connected:

1. Between the graphic user interface and SACADDOS core, the message sent is directly a PROLOG query. SACADDOS core then sends preformatted results to the user-interface. Communication between that Graphic user interface and the core of SACADDOS is based on sockets.

2. Between SACADDOS core and the document management tool, the communication is through API provided by the document management tool. Depending on the content of classification and downgrading rules, SACADDOS generates queries to be evaluated by the document management tool.

11.5 CONCLUSION

All the functionalities of SACADDOS which have been described in this paper are implemented. We have tested and compared several document management tools before choosing the one that will be used in SACADDOS. We started with a tool that only provided keyword oriented analysis and the possibility to define networks of concepts. However, this comes out to be not sufficient for SACADDOS. Therefore, it is currently being changed by a new system which includes a module for semantic analysis of full text documents. Using this system should greatly enhance the capability of SACADDOS. Due to modularity and since most work is performed on document description forms instead of documents themselves, there is no difficulty in changing the document management system used in SACADDOS. SACADDOS is being tested by security officers; the downgrading function ought to incite them to actually downgrade over-classified documents, that was not the case up to now.

There are several issues to this work. A first issue is that SACADDOS does not take into account subjective rules, i.e. rules whose conditions require evaluation of a user’s subjective judgment. Another issue of this work would be the problem of sanitizing a document, that is retrieving a lower classified subpart from a highly classified document.

We are investigating several evolutions of SACADDOS. A first one is a module called COLCHIC, which is currently being designed. COLCHIC is based on the same principles as SACADDOS, because COLCHIC manages a set of classification and downgrading security policies and includes a document management tool. The two tools provide different functionalities, whereas, SACADDOS provides assistance to users in their task of classifying and downgrading documents, COLCHIC is a module which analyzes secondary storage memory
to perform audit of information which are abnormally classified. COLCHIC can typically be used by security administrators to know what information is under or over classified in the system for which they are responsible. Another possible evolution of SACADDOS and COLCHIC would be to design a module which is inserted in a network to analyze communications and perform an audit of sensitive communications. The main problems when designing such a tool are exhaustivity and performances: this module must analyze and control every communication without hindering network performances. Finding solutions to these problems represents further work that remains to be done.

Acknowledgements

The work presented in this paper was supported by the CASSI under the Convention 95/7313929.

References


VIII  Role-based Access Controls
Role-based access controls (RBAC) have been proposed as a design and implementation approach to discretionary access controls (DAC) more apt to the requirements of commercial enterprise environments. As advantages can be mentioned centralized security administration, separation of duty and least privilege properties. However, the nature of enterprises often entails recurring sub-structures like departments, projects etc. that cannot yet be handled adequately by the available concepts for role-hierarchies. Therefore, we propose an additional mechanism for administrating role-hierarchies called role-templates. This mechanism allows to specify a generic sub-hierarchy (e.g. a department role-hierarchy) that may be instantiated for each department of the enterprise resulting in an automatically generated, concrete role-hierarchy for the particular department. Furthermore, role-templates may be specialized and have aggregations and associations to other templates making the concept more flexible and semantically expressive. The proposed ideas will be implemented as a prototype within OASIS (Open Architecture Security for Information Systems) dealing with enterprise-wide security, which demands highly configurable access controls for multiple heterogeneous information systems.

12.1 INTRODUCTION

The concept of role-based access controls (RBAC) has been proposed in the early nineties as a design and implementation approach to the common discretionary access control models (DAC, see [4]). RBAC is an approach that is more central to the processing needs of commercial environments. It has
been motivated from the fact that information is often not owned by particular individuals. Instead, the entire enterprise serves as the central owner of information and access to it should be administered more centrally, removing the burden to control multiple grant and revocation chains. Other key features of RBAC include the separation of duties, stating that different kinds of tasks should be carried out by different organizational units (roles), and the least privilege paradigm, saying that a role should be supplied with only those privileges necessary to fulfil the role’s tasks. Recently, RBAC has been found useful as an underlying security mechanism for intranets (see [20]), which are enterprise-wide information systems using internet technology. RBAC allows to define role-hierarchies representing the organizational and functional structure of an enterprise and relieving the administrational burden of formulating authorizations for each user, individually. However, recurring structures, as can be found within a lot of enterprises, still need to be modeled by copying certain parts of the role-hierarchy, manually. For instance, enterprises commonly are divided into departments that often embody similar structures: a department head, a department secretary, and department members etc. Furthermore, enterprises frequently organize their work within projects which, again, may be structured in a similar way, having a project manager, project members etc. An administrator is forced to model the part of a role-hierarchy representing a department for each department of the enterprise. In addition, the sub-tree of the role-hierarchy representing a project structure has to be copied for each new project the enterprise is about to start. Our contribution within this work is the definition of role-templates, that allow to generically define recurring role-structures for multiple instantiation. The concept has several advantages: on the one hand, the administrational effort for defining recurring role-hierarchies is reduced, significantly. On the other hand, the process of defining role-hierarchies is less vulnerable to errors, since instantiating a role-template that generates the required part of the role-hierarchy, possibly together with essential user assignments and authorizations, can be done automatically and need not to be copied manually by the administrator.

Within MeSMo (Meta Security Model), a project funded by the Austrian FWF\(^1\), we develop a generic security model, that can be configured with great variety. The model is integrated within OASIS (Open Architecture Security for Information Systems) which especially concentrates on providing enterprisewide security, dealing with the problems of providing authentication and access controls across multiple heterogeneous and distributed information systems. Role-templates are a part of MeSMo as supplement to regular RBAC and DAC mechanisms.

The remainder of this paper is structured as follows: section 12.2 provides a quick overview of role-based access controls in general. Section 12.3 goes into detail with role-templates, introducing template-roles, the template definition

---

\(^1\)This work is supported by the Austrian FWF (Fonds zur Förderung der wissenschaftlichen Forschung) under project number P12314-TEC.
view, and the embedding of role-templates within the role-hierarchy. Section 12.4 investigates possible relationships among role-templates which are associations, aggregations, and specializations respectively generalizations. Finally, section 12.5 concludes and addresses some open issues and future research directions. We use UML (Unified Modeling Language, compare [3]) throughout this paper for illustrating role-hierarchies and conceptual modeling.

12.2 ROLE-BASED ACCESS CONTROLS (RBAC)

This section describes the basic terms and ideas of role-based access controls. For a more detailed discussion we refer to the work of [15], [14], or [5], among others.

A basic role-based security model consists at least of the following elements: U (a set of users), R (a set of roles), and A (a set of authorizations) related as shown in Figure 12.1.

![Figure 12.1 Elements and relationships of the basic RBAC model.](image)

A role can be regarded as a job describing what has to be done regardless of who has to do it. Each user is associated to a set of roles within the membership relationship, saying that the particular user is allowed to activate those roles, he is a member of. Furthermore, each role is associated to a set of authorizations determining the access rights applicable for the user who activates the particular role. Membership and authorization are static relationships defined on administration-time whereas activation is dynamic and changes during run-time. RBAC models may also allow a user to run an arbitrary number of sessions with a diverging set of activated roles expressed by the session element in Figure 12.1.

Role-based security models often allow roles to be structured within an acyclic, directed graph called the role-hierarchy reflecting parts of the functional and organizational structure of an enterprise. The notion \( r \rightarrow r' \) determines that \( r \) is the super-role of \( r' \) respectively \( r' \) is the sub-role of \( r \), or in other words: \( r' \) is a more specific, \( r \) a more general role. The sub-role relationship is used to provide inheritance along the role-hierarchy in analogy to class-hierarchies within object-oriented systems. Inheriting user-membership states, for instance, that a user who is member of \( r \) automatically is a member...
Inheritance concerning authorizations states, for instance, that an authorization specified for \( r \) is also applicable to \( r' \). The concept of inheritance is particularly useful for reducing the expenditure of administrating an RBAC model. Figure 12.2 illustrates an example role-hierarchy showing a typical sub-hierarchy within a project-oriented enterprise. Note, that we consistently show the most general roles at the top and the most specific roles at the bottom of the hierarchy, following the notion for class hierarchies within object-oriented systems.

Several constraints are used in order to augment the semantic expressiveness of RBAC security models. Mutual exclusion constraints, for instance, may be applied to the membership as well as the activation relationship between users and roles. The former constrains an administrator concerning assignment of users to roles, the latter constrains the user concerning role activation. As an example, the roles *Programmer* and *Tester* could be mutually exclusive preventing users to simultaneously activate them. Quantity constraints are particularly useful for expressing concepts like the 4-eyes-principle, or the least-privilege-principle. For example, users could be forced to activate at most 1 role, as defined, for instance, within the FADAC security model (compare [6] and [7]), since users tend to activate as many roles as possible in order to achieve the maximum set of privileges. On the other hand, particular roles could be constrained in the way that at least 2 users (4-eyes-principle) have to activate a role in order to utilize the roles privileges. The activation of particular roles may also be influenced by time and/or location constraints, saying that role *ProjectStaff* (including sub-roles) could only be activated on Mondays to Fridays, between 9 a.m. and 5 p.m., for example, or that role *Administrator* could only be activated from a highly trusted host. When specified on authorizations time constraints lead to temporal authorization models, as defined, for instance, in [2] and [1].

Within DAC, information is assumed to be owned by individual users and that access rights are administrated under the discretion of the owners, therefore. In contrast, RBAC has been motivated by the fact that information is owned by the whole enterprise rather than by individual users. Thus, administration within role-based security tends to be more centralized than in
ownership-based models and includes the following particular tasks: (1) modifying the role-hierarchy (modeling the structural aspect of an enterprise), (2) assignment of authorizations to roles (modeling the functional aspects of an enterprise), (3) assignment of users to roles (specifying who has to fulfil which of the specified tasks), and (4) assignment of constraints (for role membership, role activation, and authorizations).

12.2.1 Related Work

In the early 1990s role-based access controls (RBAC) were proposed as a type of non-discretionary access controls more central to the secure processing needs of non-military systems. Key features supported by RBAC were central administration and separation of duties. [12] refines the role of roles as a job describing what must be done regardless of who does it. Constraints concerning location, time, and data could be specified when defining a role. [5] presents an authorization mechanism for URBS (user role based security) within an object-oriented design model. [13] extend the URBS specifications for relational and active database systems. [19] presents a ground-work for developing a consensus on the meaning of "role-based". A multi-dimensional view on RBAC comprising the nature of privileges and permissions, hierarchical roles, user assignment, privilege and permission assignment, role usage, and role evolution. [15] presents a taxonomy of role-based access control models ranging from a basic model to models supporting role hierarchies and/or constraints.

[14] presents in detail the organizational aspects of role-hierarchies using graph theory. The authors introduce a minimum and maximum role for completeness of the role-graph and present different forms of role organization comprising inheritance with partial, common and augmented privileges. [18] distinguishes two kinds of role hierarchies, the regular hierarchy and the administrative hierarchy. The latter comprises exclusively roles having the administrative task to associate users to roles. The set of roles an administrative role is allowed to administrate is determined by either enumeration or by providing a range of roles within the role-hierarchy. The set of users that an administrative role is allowed to associate is determined by the notion of a pre-requisite role.

[17] uses special constraints in order to enforce lattice-based (mandatory) access controls with RBAC components. [20] examines RBAC for intranet security. The authors distinguish global and local role-hierarchies, the former specify access to resources throughout the intranet and the latter specify the individual role-hierarchies on the intranet's component servers. Furthermore, relationships between global and local roles are described. [9] examines roles in connection with workflow security especially concentrating on alter-egos - a concept representing (and not merely identifying) individuals in Cyberspace.
Within earlier work, the authors already treated some aspects concerning role-based access controls. [6] presents design choices for the IRO-DB\textsuperscript{2} security policy, being role-based with mixed administration and ownership paradigm for authorization propagation. [7] proposes the extensions to the global architecture of IRO-DB in order to include security enforcement. [8] presents object-oriented access controls (OOAC) as a concept providing a strictly object-oriented common ground for implementing access control mechanisms like role-based security, for instance.

12.3 ROLE-TEMPLATES

In this section we introduce the concepts role-template and template-role as mechanism to handle recurring role structures within an enterprise. These concepts intend to assist the security administrator in the efficient and consistent design and creation of role-hierarchies. Furthermore, we discuss the instantiation of role-templates requiring two different views of the role-hierarchy, namely, the template definition view and the template instantiation view. Various kinds of embedding template-roles within the role-hierarchy are feasible which are presented in sub-section 12.3.4. Finally, an example is given in order to demonstrate the proposed concepts for which Figure 12.3 provides an overview illustration.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{roles_template roles.png}
\caption{Roles, Template-Roles and Role-Templates.}
\end{figure}

12.3.1 Template-Role

Conventional roles, as described in section 12.2, are the basis for access control decisions within RBAC systems. Any user who is a member of a particular role may activate that role in order to receive the desired authorizations. Template-roles are in contrast to conventional roles not directly used for access control; they reside at a superior level for role design we call the template definition view (compare sub-section 12.3.3). Template-roles reside within role-templates and become concrete only by instantiation. Nevertheless, users may be assigned to template-roles expressing the fact that a particular user shall become a member

\textsuperscript{2}IRO-DB (Interoperable Relational and Object-Oriented Databases), partially supported by the European ESPRIT III program under project Nr. 8629.
of each instantiation of that particular template-role. For instance, a chief secretary within an arbitrary enterprise shall automatically become a member of each project secretary role within the enterprise. Since template-roles are used to handle recurring structures it is reasonable to assign authorizations to them, too. At each instantiation of the template-role the specified authorizations are assigned automatically to the generated concrete role and are available for any of the assigned users.

12.3.2 Role-Template

A role-template is a named set of template-roles and defines a specific part of a role-hierarchy that can be applied multiple times within an enterprise for diverging purposes. In the simplest form, a role-template consists of only one template-role and is therefore equal to that template-role. In case of multiple template-roles, a template-role-hierarchy may be specified although the concept of role-hierarchies is not a prerequisite for the application of role-templates. Relationships do not only occur within role-templates, respectively between the template-roles of a role-template, but also across role-templates as well as between role-templates and normal roles as specified in section 12.4.

12.3.3 Instantiation of Role-Templates

A role-template has to be instantiated in order to produce concrete roles within the role-hierarchy. The role-template may be instantiated several times which generates concrete roles that have unique names for all the template-roles defined within the role-template. Role-Templates may contain optional template-roles which are instantiated on demand rather than automatically. Thus, the administrator must explicitly affirm or negate the instantiation of an optional template-role when instantiating the appropriate role-template. By applying this mechanism it is possible to specify roles that belong to templates, but are not necessarily instantiated together with the other template-roles. We define two different views representing the administrational states before and after role-template instantiation, which are:

- **Template definition view**: contains role-templates including their template-roles as well as concrete roles defining their embedding within the role-hierarchy. Instantiated template-roles do not occur within this view.

- **Template instantiation view**: consists of the actual concrete roles together with those template-roles that have already been instantiated. Thus, the template instantiation view corresponds to the role-hierarchy within systems not using role-templates.

12.3.4 Embedding of Template-Roles within the Role-Hierarchy

Template-roles may be embedded within a given role-hierarchy and therefore be related to other concrete roles. The different kinds of embedding and their consequences are described in the following sub-sections. For simplicity and demon-
stratification reasons we assume a role-template consisting of only one template-role, as shown in Figure 12.4.

12.3.4.1 Independent Template-Role. In the simplest case, a template-role \( B \) is not embedded into the role-hierarchy, as shown in Figure 12.4a. Consequently, the concrete roles generated by instantiating the template-role \( B \) are not related to any other existing roles within the role-hierarchy.

12.3.4.2 Sub-Template-Role. A template-role \( B \) may be embedded within the role-hierarchy as a sub-role of a concrete role \( A \) (compare Figure 12.4b). After instantiating \( B \) the specific role generated from the template \((B')\) becomes a sub-role of the concrete role \( A \). The behavior concerning inheritance of authorizations and/or user-membership for \( B' \) is just the same as for any other sub-role within the role-hierarchy.

12.3.4.3 Super-Template-Role. A concrete role \( B \) may have a template-role \( A \) as its super-role (compare Figure 12.4c). When the template is instantiated \( B \) becomes the sub-role of the newly generated role \( A' \). Since templates may be instantiated several times this kind of embedding requires multiple inheritance: as soon as the template is instantiated the second time the originally existing role receives a second super-role.

12.3.4.4 Super- and Sub-Template-Role. Template-roles may have concrete roles as both, super- and sub-roles (compare Figure 12.4d). As long as the template is not instantiated \( C \) remains the direct sub-role of \( A \). Once the template is instantiated the newly generated role \( B' \) (the instantiation of \( B \)) is inserted between the two concrete roles and \( C \) becomes the direct sub-role of \( B' \) respectively \( B' \) the direct sub-role of \( A \).

12.3.5 Example

We now illustrate role-templates as well as the embedding of template-roles within a role-hierarchy by means of an example. The template definition view of Figure 12.5 shows a role-template for a project structure consisting of three
template-roles, namely, *project staff* (PS) having two sub-template-roles *project manager* (PM) and *project team* (PT).

Since project managers from different projects may have certain common properties and duties a role *common project manager* (CPM) is defined as super-role of the template-role PM in order to propagate the properties of CPM to any project manager. Furthermore, a role *tester* (T) is defined which enables its members to test results of any of the projects. For this reason, T shall inherit all properties and authorizations that are assigned to the project teams of the various projects and is therefore defined as sub-role of the template-role PT.

The template instantiation view shows the role-template instantiated with the projects 1 and 2. Both project managers PM<1> and PM<2> are sub-roles of the common project manager role CPM. Furthermore, the role tester (T) is a sub-role of both project teams PT<1> and PT<2>.

### 12.4 RELATIONSHIPS BETWEEN ROLE-TEMPLATES

Besides relationships between template-roles and concrete roles there may also be relationships across role-templates. As soon as a template-role is a sub-role of a template-role that belongs to different role-template, the two templates have a relationship. Since we only regard inheritance within the role-hierarchy, relationships between templates also always affect the *is-a* relationships between roles. We distinguish three different kinds of relationships between templates, namely, *association*, *aggregation*, and *inheritance*. Regarding association and aggregation the cardinality specifies the number of templates that participate in the relationship whereas the conditionality specifies whether the participants may or have to take part in the relationship.

#### 12.4.1 Association

Associations between objects indicate an *is-related-to* relationship. For our purpose an association between role-templates means that one of the template-roles has an *is-a* relationship to a template-role of a different template. The
specification of cardinality has an impact on the instantiation. To illustrate this impact, consider an example consisting of two templates, namely a template \textit{Department} and a template \textit{Project} and possible associations between them. Figure 12.6 shows the template definition view of these two templates.

![Figure 12.6](image)

\textbf{Figure 12.6} Relationship between two role-templates.

Now let us regard a \textit{one-to-zero-or-one}, a \textit{one-to-zero-or-more} and a \textit{one-or-more-to-zero-or-more} relationship. In the first case, departments may be instantiated without instantiation of projects, which is not true for the reverse order.

When instantiating a \textit{Project} the generated role for \textit{ProjectStaff} is inserted as sub-role of a concrete \textit{DepartmentStaff} role. The \textit{one-to-zero-or-one} association furthermore indicates that each department may only run one project and vice versa one concrete project may only be carried out by one department (compare Figure 12.7), thus, each instantiation of the template-role \textit{ProjectStaff} (\textit{PS}) is related to one instantiation of the template-role \textit{DepartmentStaff} (\textit{DS}).

![Figure 12.7](image)

\textbf{Figure 12.7} 1-to-0..1 relationship between role-templates.

Figure 12.8 shows a \textit{one-to-zero-or-more} relationship and the corresponding template instantiation view. In this example departments may run several projects, and therefore each instantiation of the template role \textit{DepartmentStaff} may have several instantiations of \textit{ProjectStaff} as sub-roles.

The last example considers a \textit{one-or-more-to-zero-or-more} association (compare Figure 12.9). This indicates that projects may be carried out by several departments. Thus instantiations of \textit{ProjectStaff} may have several instantiations of \textit{DepartmentStaff} as super-roles.
12.4.2 Aggregation

The term aggregation denotes a part-of relationship, which means that components are part of an aggregate object. [16] define aggregation as a special strong form of association, which vary by a somewhat different semantic. However, no significant particular features exist to distinguish between them. Components may or may not exist independently from an aggregate and they may appear in multiple aggregates. The user is recommended to choose an aggregation if objects are tightly bound by a part-whole relationship, whereas independent objects shall be related via an association. [10] and [11] discusses different aspects that should be considered when distinguishing between association and aggregation. Since the distinction between associations and aggregations has no consequence on our approach, we refer to the discussion on cardinality of the previous section.

12.4.3 Specialization, Inheritance

Since it may be useful to define different versions of role-templates it is reasonable to build specializations respectively generalizations of them. Figure 12.10 shows a specialization of the role-template Department which contains the additional template-role DepartmentSecretary. According to the requirements Department or the specialization of Department may be instantiated. Specializations may also differ concerning user-membership and assignment of authorizations (compare section 12.3.1). In the case that role-hierarchies are not supported DepartmentSecretary could also be defined as optional template-role.
12.5 CONCLUSIONS AND FUTURE WORK

In this paper we presented the concept of role-templates, a mechanism to handle recurring role structures. Role-templates support an efficient and consistent management of roles by assisting the security administrator regarding their design and generation. They allow to generically define recurring role structures, consisting of template-roles that may also form role-hierarchies. It is possible to specify the embedding of template-roles within the concrete role-hierarchy and to define association, aggregation and specialization/generalization relationships between role-templates.

Role-templates, together with their embedding and their relationships are represented within the template definition view. They may be instantiated several times, whereby the newly generated roles are shown within the template instantiation view, which corresponds to the normal role-hierarchy endorsed with the instantiated template-roles.

The approach of role-templates shall not lead to rigid instructions concerning security aspects. It is the task of a concrete implementation or configuration facility to keep it flexible enough to be applicable and useful. Therefore, role-templates could be handled as proposals and not as strict defaults, allowing the administrator to change or revoke properties of generated roles, as relationships or assigned authorizations.

Further research will concentrate on a detailed consideration on open issues, like the influence of negative authorizations, for instance. Concerning the assignment of authorizations to role-templates more advanced concepts, like generic authorizations, are conceivable. Such mechanisms would allow to handle flexible authorizations for recurring object structures that correspond to role-templates. The proposed approach will be implemented as a prototype within the project MeSMo (Meta Security Model).

References


Abstract: In the past two years, Java has exploded onto the computing landscape, offering an object-oriented language and environment that is suitable for a wide variety of application domains. Java is targeted for applications that include: advanced capabilities in WWW browsers via applets; enterprise computing with database connectivity, CORBA, and RMI; usage in personal, commercial, and consumer market products; embedded computing applications with real-time constraints; and, smart card technology. Security is an integral component of many of these applications, to control access and prevent misuse. The purpose of this chapter is to focus on the security capabilities and potentials of Java. There must be an understanding of the available security primitives in Java, an investigation of the ability of Java to support existing object-oriented security approaches, and a consideration of potential security solutions for distributed object computing applications.

13.1 INTRODUCTION

The Java object-oriented programming language and environment first appeared commercially in early 1996, and in just over 2 years time, there has been an explosive interest and growth of Java across the computing landscape. Java is utilized for distributed, Internet-based applications of all types, including: Web browsers, graphical user interfaces (GUIs), programming environments, mixed-programming language applications, upgrading and interfacing to legacy systems.

*The work of S. Demurjian and C. Bastarrica has been supported, in part, by a contract from the Mitre Corporation, Eatontown, NJ.
From a security perspective, the usage of Java for the design and development of large-scale, multi-processor, distributed applications, is of paramount concern. Successful distributed object computing (DOC) can be addressed from three perspectives. First, when developing new applications, it is often the case that multiple programming languages and varied paradigms must work together, e.g., a Java GUI, a C I/O package, and an SQL database system. This motivates the second perspective, involving the integration of Java with commercial-off-the-shelf (COTS) systems. Of course, when integration occurs, it may be necessary to be innovative and creative to allow interactions with legacy applications, which is the third perspective. In all three perspectives, security must be considered, to insure that interoperating legacy, COTS, and database systems can satisfy the security policy of distributed applications. There have been efforts to begin to address security for DOC [4, 9].

The purpose of this chapter is to examine and detail the security capabilities and potentials of the Java language/environment. Java provides a robust set of security capabilities as part of the Java Security Application Programmers Interface (API). These capabilities include digital signatures, message digests, key management, and access control lists. A first goal of this chapter is to detail these capabilities, so that the security community can understand the available functionality. Java, as an object-oriented programming language, is of interest from a user role-based security (URBS) perspective, to determine the potential to realize discretionary access control (DAC). Many researchers, including ourselves, have studied this problem for object-oriented/C++ applications and systems [1, 2, 5, 7]. A second goal of this chapter is to consider the applicability of our previous approaches to Java. This leads to a third goal, an exploration of the unique features of Java that can be used to enhance existing URBS/DAC/OO approaches or to support new approaches for distributed object computing security.

To meet these goals, the remainder of this chapter is organized into four sections and a conclusion. In Section 2, a brief overview of the Java language/environment is provided. In Section 3, the security capabilities in the Java Security API, are examined, targeting the first goal described previously. Section 4 explores the realization of URBS/DAC approaches in Java, from prior work [2], targeting the second goal. Section 5 examines advanced security capabilities, concentrating on the potentials of Java, and targeting the third goal. Finally, Section 6 concludes this chapter and outlines future work.

13.2 BACKGROUND: AN OVERVIEW OF JAVA

Java is a third generation, general-purpose, platform-independent, concurrent, class-based, object-oriented language and associated environment. Java can be used to write special programs called applets that can be downloaded from the Internet and displayed/manipulated safely within a Web browser, or to develop
standalone applications, with a wide range of capabilities and functionality. Java has two main components, the Java Development Kit (JDK) and the Java Runtime Environment (JRE). The JDK is a package of programs and support files which is needed to develop Java programs. The JDK contains the command-line driven javac Java compiler. The Java Debugger (JDB) is included with the JDK. The JRE used to execute Java applications, and consists of the bytecode interpreter and other files such as the code verifier. Version 1.1.6 of both the JDK [12] and the JRE [14] are available from Sun for Microsoft Windows 95/NT 4.0 and Solaris, with third-party ports [13] to a wide variety of other OSs.

In order to support platform independence, Java provides an environment to oversee the execution of applets and applications, the Java Virtual Machine (JVM). JVM is a program which runs on a particular hardware/OS platform (or ‘real’ machine) which interprets and executes a Java applet/application that is contained in a .class file. The .class file contains both executable JVM instructions (called bytecodes), and additional information such as the class structure, method and data member visibility, and superclass information. Since each JVM interprets the same set of bytecodes, true program portability is achieved by implementing JVMs for a wide variety of platforms.

The main modeling capability is the Java class, which is similar to a C++ class. Within a Java class, a member (method or variable) can be tagged as private, public, protected, or package (default). A class for prescriptions in a health care application is given below:

```java
public class Prescription
{
    public String Get_Prescription_No( ... ) { ... }
    public void Set_Prescription_No( ... ) { ... }
    public String Get_Pharmacist_Name( ... ) { ... }
    public void Set_Pharmacist_Name( ... ) { ... }
    public String Get_Medication( ... ) { ... }
    public void Set_Medication( ... ) { ... }

    private String prescription_no;
    private String pharmacist_name;
    private String medication;
}
```

Classes that are related to one another can be grouped together into the package abstraction, to be discussed in Section 5 of this chapter. Inheritance is supported in Java by using the extends keyword when declaring a class.

Java, through its public interface capabilities and package concepts also requires a clear definition of the exported portion of all classes/packages, which requires software engineers to specifically enumerate which packages, classes, and/or methods are imported. Thus, like Modula-2 and Ada83 (and of course, Ada95), Java provides a set of application programming interface (API) packages. The Java Platform 1.1.6 Core API is available online [10]. Each API
contains a complete description of the package, classes and public methods that can be imported and utilized to develop Java applications.

13.3 JAVA AND SECURITY

Java provides transparent, general, and open security mechanisms which do not require any knowledge or action on the part of the software engineer. The sandbox is Java’s basic security mechanism, which forces downloaded applets to run in a confined portion of the system, and allows the software engineer to customize a security policy. One result of this approach is that the security policy is hard coded as part of the application, providing little or no flexibility either to modify the policy or to have discretionary access control. The Java language/environment has features that assist in protecting the integrity of the system and preventing several common attacks. This section describes the security capabilities in the Java Security API [11].

Sandboxes: An applet’s actions are restricted to its sandbox, a dedicated area of the Web browser. The applet may do anything it wants within its sandbox, but cannot read or alter any data outside it. The sandbox model supports the running of untrusted code in a trusted environment so that if a user accidentally imports a hostile applet, that applet cannot damage the local machine. To implement sandboxes, the Java platform relies on three major components: the class loader, the bytecode verifier, and the security manager. Each component plays a key role in maintaining the integrity of the system, assuring that only the correct classes are loaded, that the classes are in the correct format, and that untrusted classes will neither execute dangerous instructions nor access protected system resources. Java’s Protected Domains constitute an extension of the sandbox, and determine the domain and scope in which an applet can execute. Two different protected domains can interact only through trusted code, or by explicit consent of both parties.

The class loader determines how and when applets can load classes, and is responsible for: fetching the applet’s code from the remote machine, creating and enforcing a namespace hierarchy, and preventing applets from invoking methods that are part of the system’s class loader. An executing Java environment permits multiple class loaders, each with its own namespace, to be simultaneously active. Namespaces allow the JVM to group classes based on where they originated (e.g., local or remote). Java applications are free to create their own class loaders. In fact, the JDK provides a template for a class loader to facilitate customization. Before a class loader may permit a given applet to execute, its code must be checked by the bytecode verifier. The verifier insures that the applet’s code, which may not have been generated by a Java compiler, adheres to all of the rules of the language. In fact, in order to do its job, the verifier assumes that all code is meant to crash or penetrate the system’s security measures. Using a bytecode verifier means that Java validates all untrusted code before permitting execution within a namespace. Thus, names-
paces insure that one applet cannot affect the rest of the runtime environment, and code verification insures that an applet cannot violate its own namespace.

**Security Managers:** The *security manager* enforces the boundaries around the sandbox by implementing and imposing the security policy for applications. All classes in Java must ask the security manager for permission to perform certain operations. `SecurityManager` is an abstract class of the `java.lang` API, and provides the programming interface and partial implementation for all Java security managers. By default, an application has no security manager, so all operations are allowed. But, if there is a security manager, all operations are disallowed by default. Existing browsers and applet viewers create their own security managers when starting up.

When there is a security manager, each operation or group of operations will have its own `checkXXX` method. There are `checkXXX` methods for operations on sockets, threads, files, networking, windows, etc. To write a security manager, it is necessary to create a subclass of `SecurityManager` and override most or all of its methods: `class MyPolicy extends SecurityManager { ... }`. Once a new security manager is created, it can be installed with the `setSecurityManager` method from the `System` class. The security manager will remain active until the end of the application. A method that opens a file for reading invokes the `checkRead` method of the security manager. A method that opens a file for writing invokes the `checkWrite` method. If the security manager approves the operation, the `checkXXX` method returns, otherwise, it throws a `SecurityException`.

**Digital Signatures and JAR files:** If a particular publisher is trusted, and a signed applet from that publisher has arrived over the Internet and been authenticated, then the Java Security Manager could allow that applet out of the sandbox, and treat it as an application. The first task of any security system is to be able to assure that who or whatever is on the other side of a connection is who or what the user expected to be there, i.e., the host that they have connected to is the host they contacted and not an impostor, or the module that they have loaded is really the one they expected to run and not a substitute. This is of particular concern in downloaded environments where there is a constant threat of a *Trojan Horse*.

The man-in-the-middle/middleman is a type of attack to which all network-based systems might be vulnerable, and proceeds in a number of steps. First, a client application requests some service from a legitimate server. Unknown to both client and server, an attacker application observes this request and waits for the server to respond. When it does, the attacker intercepts the server's response and replaces it with one of its own, one that the client may assume came from the original server. The way to prevent this type of attack is to ship code contained within a digital shrink-wrap, which is achieved in Java using *signed applets*. A supplier bundles Java code (and any related files) into a JAR
(a Java Archive), and then signs the file with a digital signature. The client can verify the authenticity of the supplier by verifying the signature.

**Key Management:** The Java Security API provides support for integrated key management in Java programs and applets. Keys are generally obtained through key generators, certificates, or the various Identity classes used to manage keys. There are no provisions yet for the parsing of encoded keys and certificates. An identity certificate is a guarantee by a principal that a public key is that of another principal. The KeyPairGenerator class is used to generate pairs of public and private keys.

**Message Digests:** Cryptographers have developed a way to generate a short, unique representation of your message, called a message digest, that can be encrypted and then used as your digital signature. The MessageDigest class provides the functionality of a message digest algorithm, such as MD5 or SHA. Message digests are secure one-way hash functions that take arbitrary-sized data and output a fixed-length hash value. Like other algorithm-based classes in the Java Security API, MessageDigest has two major components: the methods called by applications needing message digest services and the interface implemented by providers that supply specific algorithms.

**Access Control Lists:** Every authenticated principal will have a level of accessibility: highly trusted resources should be granted more access than those of more dubious origin. Access Control Lists (ACL) are data structures used to guard access to resources, and allow users to define read/write permissions based on users and groups. Each ACL entry contains a set of positive or negative permissions associated with a particular principal (an individual user or a group). Individual permissions (either positive or negative) override the groups' permissions. The java.security.acl package provides the interfaces to the ACL and related data structures (ACL entries, groups, permissions, etc.), and the sun.security.acl API provides a default implementation.

### 13.4 JAVA AND USER-ROLE BASED SECURITY

Security issues in distributed object computing are difficult to address since security of individual systems (legacy, COTS, database) must be supported in the distributed and interoperating application. In this section, we consider the ability of Java to support user-role based security (URBS) approaches, where permissions and right to access are be assigned based on the individual roles, rather than to specific users. URBS is a realization of discretionary access control (DAC), that assigns rights and permissions to roles rather than to individual users, with users assigned to specific roles [2, 3].

The premise of our efforts is that the public interface provided by object-oriented programming languages is not suited for the customized approach that is needed for supporting URBS/DAC. The public interface of a class is the union of all privileges (methods) needed by all users of each class. This
allows methods intended for only specific users to be available to all users. Our past approaches have strengthened the public interface concept, promoting the idea that different subsets of the public interface are available to specific users based on role, thereby providing a means to realize URBS/DAC. We have detailed a number of extensible and reusable URBS/DAC enforcement mechanisms that utilize inheritance, generics, and exception handling for the automatic generation of code for DAC policies [2, 3]. This section begins by providing background material on our URBS model. Then, this section reviews the realization of three of our prior URBS approaches in Java. We complete this section with remarks on the limitations of Java in support of our URBS.

### 13.4.1 User-Role Based Security

To support URBS, the user-role definition hierarchy (URDH) characterizes the different kinds of individuals (and groups) who require different levels of access to an application. In a health care application (HCA) there would be user roles (UR) such as Staff_RN, Attending_MD, Education, etc., that can be grouped under a single user type (UT) (e.g., Nurse). When multiple UTs share privileges, a user class (UC) can be defined (e.g., Medical_Staff). To define, UCs, UTs, and URs, we utilize a node profile (NP): 1. a name for the node; 2. a prose description of its responsibility; 3. a set of assigned methods (the positive privileges); 4. a set of prohibited methods (the negative privileges); and 5. a set of consistency criteria for relating URDH nodes. Assigned and prohibited methods are of primary interest for our discussion, since they focus on what actions are allowed/denied for each UR.

### 13.4.2 User-Role Subclassing Approach

In the user-role subclassing approach, URSA, each application class has a group of subclasses, based on the different roles that have some subset of assigned and/or prohibited methods from the class. As subclasses, the basic concept is to inherit and turn off the prohibited methods.

```java
public class Prescription { // As given in Section 2 }

public class Staff_RN_Prescription extends Prescription
{
    public void Set_Prescription_No(....)
    { return; // Prohibit access to this method - Turn Off }

    public void Set_Pharmacist_Name(....)
    { return; // Prohibit access to this method - Turn Off }

    public void Set_Medication(....)
    { return; // Prohibit access to this method - Turn Off }
}

public class Attending_MD_Prescription extends Prescription
{
    public void Set_Pharmacist_Name(....)
```
If the Set_Prescription_No method is requested by an individual whose role is Staff.RN, then the method associated with the Set_Prescription_No of the Staff.RN_Prescription subclass is executed and no value is returned.

### 13.4.3 URDH Class Library Approach

In the URDH class library approach (UCLA), a new inheritance hierarchy is used, where each class is a URDH node. For each URDH node, positive methods access is defined based on the assigned methods that have been specified. As the application executes, methods must validate against the current UR.

```java
public class Root { All Check Methods defined to return False;
public class Users extends Root {}
public class Medical_Staff extends Root {}

public class Nurse extends Medical_Staff
{ public boolean Check_Prescription_Get_Medication()
  { return True; }
}

public class Staff_RN extends Nurse
{ public boolean Check_Prescription_Get_Prescription_No()
  { return True; }

  public boolean Check_Prescription_Get_Pharmacist_Name()
  { return True; }
}

The Root class includes new Check methods, which are defined for all application methods to return False. These check methods will be turned on at lower levels (UC/UT/UR) by the assigned methods of the URDH. These Check methods are also utilized to change the code that can be generated for each class:

```java
class Prescription extends Item
{ public Prescription(String N, String D, int No, String N1, String M)
  { // initialize variables }
public int Get_Prescription_No()
  { if (current_user.Check_Prescription_Get_Prescription_No())
    return (Prescription_No);
  else
    return NULL;
}
public void Set_Prescription_No(int No)
{ if (current_user.Check_Prescription_Set_Prescription_No())
   Prescription_No = No;
}
Once the user role has been determined, a new global `current_user` object will be created at run-time and casted to the selected user role.

### 13.4.4 Basic Exception Approach

Exception handling in Java is similar to that of C++, where the try construct is utilized to encapsulate a block of code that has the potential to raise an exception. As the code within the try block is executing, various conditions can be checked, and when the correct situation occurs (e.g., unauthorized UR or a call by an authorized UR to a prohibited method), an exception can be thrown and processed by the catch block (e.g., to process the security violation). In the basic exception approach (BEA), each class is modified to include a set to methods for exception handling. This is illustrated below.

```java
public class Prescription extends Item {  // Private data has been omitted

    public Prescription(String N, String D, int No, String N1, String M)
    { // Assign Prescription variables, call Item constructor }

    public int Get_Prescription_No()
    { return(rtn_int_check_valid_UR(Prescription_No)); }

    // All Other Prescription methods

    public int rtn_int_check_valid_UR(int rtn_int_ck)
    { 
        try { Check_UR(); }

        // Catch block to process raised exceptions
        catch (Unauthorized_UR UR_exception)
        { System.out.println("Attempt to access unauthorized UR"); }
    }

    // All other data type check_valid_UR methods

    public void Check_UR() throws Unauthorized_UR
    { if ((compareTo(current_user.Get_User_Role(),"Staff_RN") != 0) 
        (compareTo(current_user.Get_User_Role(),"Attending_MD")!= 0))
        throw new Unauthorized_UR;  // throw raises exception
    }

    Check_UR is needed to verify that the current UR can invoke the desired method via a table lookup. The class `Unauthorized_UR` is an exception handling class where code can be provided to handle security violations.
```
13.4.5 Limitations of Java in Support of URBS

While Java appears to easily support the various approaches given in Sections 4.2, 4.3, and 4.4, in actuality, Java has some limitations:

- UCLA, as presented in Section 4.3, is in fact, not fully supported. UCLA as originally conceptualized in C++ [2] requires multiple inheritance, which is needed to define the URDH. While Java can realize UCLA through the replication of privileges from User into either the user types or user classes, it is not an ideal solution. The interface capability of Java, which supports design-level multiple inheritance, is also not appropriate, since interfaces do not allow implementations to be inherited.

- UCLA and BEA, as presented in Sections 4.3 and 4.4, respectively, have corresponding approaches (GUCLA and GEA) that utilize generics [3]. Java, without generics, the ability to reuse security definition and enforcement code is a major drawback of the language.

While Java appears to have stabilized from a language design perspective, the user community may call of the inclusion of both multiple inheritance and generics, since both concepts are fundamental to software reuse.

13.5 ADVANCED SECURITY FEATURES AND URBS

This section focuses on the third goal of the chapter, the ability to utilize security features of Java for URBS, thereby truly exploring the potentials of the language. The remainder of this section considers four advanced capabilities of Java and their potential for supporting URBS: packages for encapsulating security definition and enforcement code; access control lists; the Class class of the Java Language API; and, software agents which are supported by Java aglets.

Packages in Java The highest level of abstraction/encapsulation in Java is the package, which allows collections of one or more classes to be bound into a single named unit. For example, consider a PatientInfo package:

```java
package PatientInfo;
class Prescription { ... };
class PatientGUI { ... };
class MedicalRecord { ... };

package PatientInfo;
public class Prescription { ... };
public class PatientGUI { ... };
public class MedicalRecord { ... };
... 
```

In the version on the left, the classes are only visible within the package in which they are defined. In the version on the right, classes tagged with the public qualifier are visible within the package and externally.

The package construct can be instrumental in encapsulating the security definition and enforcement code that is required for the different URBS approaches. In UCLA (see Section 4.3 again), the entire URDH class library can be encapsulated into a single package, allowing changes to the URDH to be localized to
a single, controlled package. In URSA (see Section 4.2 again), Prescription, StaffRN.Prescription, and AttendingMD.Prescription can be encapsulated into a single package, with Prescription not tagged as a public class. This would mean that only the other two user-role subclasses, tagged as public, are visible externally, which would further protect unauthorized access to Prescription, since all access must go through the user-role subclasses.

Access Control Lists  The main purpose of the URDH is to allow methods that are defined on classes throughout the application to be assigned and/or prohibited to various user classes, user types, and user roles. The privileges associated with the URDH are directly supported in Java via the Access Control List (ACL). Each ACL entry contains a set of permissions (access to methods) and for a particular principal (UR or UT). Privileges are assigned when the principal is allowed to access a method and prohibited otherwise. The individual permissions (for URBS, the UR) will override permissions of the group (for URBS, the UT) to which an individual belongs. The following methods from java.security.acl.ACL are required for the support of URBS:

1. **addEntry():** Adds an ACL entry to the Access Control List. This entry contains the specified user and a list of methods which are assigned or prohibited for this user, according to the user role that is being played.

2. **checkPermission():** Returns true if the input user has permission to access the input method, false otherwise.

3. **getPermission():** Returns an enumeration of all methods which are assigned or prohibited for the input user. The assigned/prohibited methods are determined by first obtaining the assigned/prohibited methods for the group (in our case the UT) and then determining the assigned/prohibited methods for the individual (in our case the UR). The final permissions are then determined by allowing the individual permissions to override the group permissions for both the assigned and prohibited methods. The assigned permission set is returned.

The following methods from java.security.acl.ACLEntry are required to build a URBS ACL entry:

1. **addPermission():** Adds a permission (method) to the ACL Entry.

2. **checkPermission():** Determines if a permission (method) is already part of the ACL Entry.

3. **removePermission():** Removes a permission (method) from the entry.

4. **setPrincipal():** Specifies the user role or user class for which the permissions (methods) are assigned or prohibited.

5. **getPrincipal():** Returns the user role or user class for which these permissions (methods) are assigned or prohibited.
6. **setNegativePermissions()**: Set the ACL entry to be a list of negative permissions (prohibited methods).

For URBS, we would specify the permissions of all methods so that the method `CheckPermission()` could be invoked to accurately determine both the assigned and prohibited methods. As we stated earlier, the URs inherit all of the permissions of the parent UT. The Java API `java.security.acl.Group`, can be used to assign URs to the UTs, via the methods `addMember()` and `removeMember()`, where the UT would be the Group.

For URSA, UCLA, and BEA, ACL can be utilized to track the information required for an authorization list, that would bind users to their associated roles upon login. For BEA, the ACL has the most significant potential use, namely for the `Check.UR` method, that is able to verify which URs have access to which methods. Using an ACL, this information could be dynamically changed, whenever the security requirements cause the addition/deletion of roles or changes in application classes. While an ACL can be implemented in any language, having one designed, implemented, tested, and with a standard interface, is a definite advantage to Java.

**The Class Class in Java** In the `java.lang` API, the `Object` and `Class` classes have a large set of methods defined that are accessible to software engineers for obtaining information about any system- or user-defined class in Java. For instance, `Class` has methods that can be invoked to return, for a specific user or system class, a list of its public methods, member variables, declared constructors, etc.

The `Class` class can be used by URSA, UCLA, and BEA for the dynamic retrieval of all public methods for each class. The retrieved methods would have a default permission of assigned and only the links to the prohibited methods would need to be removed. For example, the `Check.UR` method of BEA, if implemented with ACL as described earlier, could utilize the `getMethods` method of `Class` whenever the security policy was updated. This would allow the revised/updated entries of the ACL (that contain, for each role, the assigned methods) to be automatically and dynamically compared against the actual methods defined on each class. Similarly, whenever a class was altered, this verification could also occur. In both situations, the maintenance of the security policy is greatly simplified.

**Java and Aglets** Mobile *software agents* are defined in formal terms as objects that have behavior, state and location [8]. Agents can move from place to place and have a specific function or responsibility to perform. Agents are like other objects in that they can be created and destroyed, but they can also migrate to a new location, execute their required responsibilities, and process incoming messages from other agents. Agents cannot interact by invoking each others methods, rather, they communicate via message passing.

IBM terms these mobile agents of Java, "aglets", combining the terms of "agent" and "applet" [16]. Unlike a Java applet, an aglet continues execution
where it left off (upon reaching a new location). This is possible because an aglet is an executable object (containing code and state data) which moves from host to host across a network [15]. Karjoth describes a proxy as a representative of an aglet which serves as a shield to protect the aglet from direct access to its public methods [6]. The proxy used for the aglet has the responsibility to prevent the access of unauthorized users (or agents).

Like applets, aglet actions should be restricted to a sandbox (see Section 3 again). The sandbox model supports the running of untrusted code in a trusted environment so that if a hostile aglet is received, that aglet cannot damage the local machine. For applets, this security is enforced through three major components: the class loader, the bytecode verifier, and the security manager. Aglets would require the same level of security as applets. The aglets would need to ask permission from the security manager before performing operations, thus allowing the security manager to know the identity of the aglet.

Mobile aglet security is progressing with the use of the Java sandbox mechanism and separation execution environments [6]. Java security mechanisms such as cryptography and authentication are also be investigated to ensure security of both the aglet and the messages transported between aglets. Aglets offer the opportunity to rethink our URBS security approaches, which are class/method based for user roles, and whose definition process is focused on type-level concerns. In distributed object computing, it is critical to explore the security of runtime objects, as they are accessed by users playing roles. Aglets may provide active objects that monitor and/or enforce security, from the perspective of the user, the user role, the object, or any/all combinations. As security needs change, security aglets can be dynamically updated to maintain their oversight and enforcement capability. Aglets in Java must be examined for their potential to support security in distributed object computing.

13.6 CONCLUDING REMARKS AND FUTURE WORK

This chapter has examined the security capabilities and potentials of the Java object-oriented language/environment. There are a wide range of security capabilities, provided in the Java Security API, including digital signatures, message digests, key management, and access control lists, which all function under the control of a security manager, as described in Section 3. From an object-oriented/programming language perspective, Section 4 examined the ability of Java to support our previous URBS approaches. While some of the approaches were realizable in Java, others that utilize multiple inheritance and generics could not be fully attained. Section 5 examined advanced security capabilities of Java in support of URBS. Specifically, we considered the package abstraction for encapsulating URBS code, Java's access control lists for realizing important components of our URBS approaches, and the Class class for performing automatic and dynamic verification of security privileges.

One of the more interesting potentials of Java is related to our future work, namely, the utilization of Java agents, or aglets, for supporting security in distributed object computing. Another future related area is the support of
security within the CORBA/ORB framework, which is the only available standard for distributed computing. A third related area is security capabilities offered by emerging object-oriented database platforms, including the recently announced Jasmine by CAI.

References


IX Mobile Databases
Abstract: Mobile computing and communication is a rapidly developing area. But mobility is associated with problems for security and privacy beyond those in open networks. A well known threat is tracking user movements. New risks are caused by the mobility of users, the portability of computers, and wireless links which include dynamics, resource dependencies and additional information to ensure the communication. This paper surveys the new challenges and the research on security issues in mobile data management, access and transfer. We investigate the issues concerning database specific security which have to be reconsidered. We will identify a basic characteristic of these security issues, adaptability, to answer the dynamics.

14.1 INTRODUCTION

The development of mobile devices make new applications conceivable through ubiquitous computing. For example, mobile work “on-the-spot” like disaster recovery and maintenance tasks as well as business trips are possible. Mobile computing and communication start up to be an important factor in business. On the one hand we have really inspiring possibilities, but on the other hand, security and privacy becomes more eminent with wireless computing and communication. Dynamics of the mobile environment is confronted with static security services, often scarce resources hinder the correct application of security mechanisms, and additionally managed information needs particular protection. Moreover, it is obvious that there is a chance to integrate security and privacy issues in an early design phase of this new kind of computing.
14.1.1 Mobility

In the first place, we have to explain our interpretation of mobility. While it is beyond the scope of this paper to present mobile agents, we focus on user and terminal mobility. In the case of terminal mobility a user is identifiable through his mobile terminal [10]. User mobility keeps users in the foreground. The customers roaming to pursue their aims are mobile with respect to their environments, to their locations, other persons, and terminals. They are not fixed to use one and the same device and sort of link. Every arrangement of terminal kinds (fixed or portable) and networks (wired or wireless) is conceivable (see figure 14.1). The user can for example use a laptop with a fixed network in a hotel or the same mobile device in a radio network environment. To manage user mobility, detailed information of the current computing and communication environment is necessary besides the user location information. The protection comprises safeguard of content data and the described environment data.

![Figure 14.1 General Mobile Scenario](image)

14.1.2 Mobile Databases

We assume a distribution of the database content over the whole network, e.g. there is no central database on a fixed host which will be accessed from fixed and mobile hosts, but a distribution (or replication) of fragments over both mobile and fixed hosts.

We will organize this paper according to groups of the security issues. One possible grouping contains the consideration of security goals and associated threats. The basic objectives are confidentiality, integrity and availability including accountability and non-repudiation. Main threats are information leakage, integrity violation, denial of services, illegitimate use, and unaccountability. Such a classification seems to be too general because most of security problems are confidentiality related. [3] focuses on communication problems and proposes a grouping into

- content privacy,
- unlinkability of sender and recipient and
location privacy.

In [7] a framework of the categories of

- mobility,
- disconnection,
- data access mode and
- scale of operation

is used.

We introduce in this paper a more database-related approach to mobile security. We ask what are the objects which have to be protected, and in which situations. Even in mobile environments, there are risks

- for the information itself and
- for the metadata.

The metadata concern in the mobile environment in particular the additional data accompanying the communication (also: telemetadata [16] or communication context) which are personal data and have to be protected. The safeguard should effect the data

- management and access and
- transfer.

We distinguish between actions on the mobile and the fixed site. In the next chapter we will survey the security threats for data and metadata in the data transfer (communication) over a wireless link. Most of them are problems of the underlying operating system and network layer. Also the following chapter about security issues contains the rather database-related security issues, e.g. data and metadata protection in the management and access operations and we will specify the dangerous situations in such an environment. Afterwards we show the necessary preconditions and protection approaches like contradiction between transparencies, different levels of anonymity and separation of metadata. The conclusion closes this paper with some resulting remarks.

14.2 SECURITY ISSUES IN MOBILE DATABASE ENVIRONMENTS

14.2.1 Data Security in Mobile Data Transfer

Disconnections occur often in wireless communication. They can be forced by the user because of saving communication costs or be induced by faults. This situation can endanger the data consistency, even without considering replicas. Disconnections are primarily a problem of the underlying layers of a database, but the database system is also responsible for avoiding data loss in case of such
unexpected disconnections with the help of transaction recovery. The higher frequency of network partitioning requires a more powerful error recovery than in fixed networks. Besides error recovery, this situation offers attackers the possibility to masquerade as either the mobile unit or the base station. With the help of masking the identity, data are at risk to be released improperly. Moreover, the use of a wireless link facilitates eavesdropping, because air-emitted information is accessible in a simple way without any additional effort required. This kind of security violation is hard to detect. In both cases, security relies on cryptography to achieve user authentication and data privacy. Mobile users are registered their real identity or with a pseudonym with that domain's authentication server. The authentication service should provide to the communicating parties the confidence that they are in fact communicating with each other [8, 12]. The subsequent communication should protect the data transfer content against attacks and eavesdropping. Authentication in mobile environments is e.g. described in [6, 13, 17, 18]. Most of the authors propose to use asymmetric encryption for the authentication and symmetric cryptography for a secure communication.

It is essential, that also inner-database communication between distributed fragments has to be realized securely.

14.2.2 Metadata Security in Mobile Data Transfer

We named the metadata in a mobile communication area (see figure 14.1) the mobile context. It consists of a user profile, information about the current resource situation, information characteristics, location and time. The current whereabouts and especially the movement of users are a matter of privacy, and ideally only the user herself should have knowledge about these data [7, 17]. Its protection is regarded as the main special mobile security aim. The threat of keeping user whereabouts appears on different layers. On the network layer, user location or presence in a particular radio cell, respectively, is managed in order to reach a mobile user to communicate with him. All user identification information including message origin and destination have to be protected with the help of cryptography to conceal a communication from other network users. In order to achieve anonymous communication, aliases or pseudonyms are used. Furthermore the identity of users should be kept secret against the service provider, even if they consume services which have to be paid. It is not necessary to know the identity of users to get solvency information from their home base node. The home site is informed about the aliases and the real identity. Safeguarding of anonymity additionally against the home base node requires a trusted third party to manage the pseudonyms.

An implicit method to disclose the location lies in the possibility to carry out a traffic analysis. Prevention against traceability of network connections in mobile environments can be offered through either MIXes [2, 14] or the Non-Disclosure-Method [5]. Both methods use cryptography. MIXes delay and collect different messages and send it in a shuffled sequence to the receivers or another MIX. Using MIXes requires only a modification of the available networks, but a sufficient amount of messages with equal length is necessary,
otherwise dummy traffic will be created.
In the Non-Disclosure-Method, introduction of Security Agents (SA) is proposed. The communication path is masked through a sender selected route (with detours) over a number of different SA's. Each SA only knows his predecessor and successor in the routing chain. Security increases in case of widely scattered SA's, possibly among different providers. But the detours assume an intact and wired network. In case of database requests, MIXes and detours extend the response time in a dynamic way and hinder an efficient optimization. Another aspect of wireless communication security, the permanent reachability of persons, endangers the user’s claim of self-determined communication. In [15], an implemented approach for personal reachability management is proposed. The main idea is to evaluate and negotiate a communication request and to decide automatically by a Reachability Management System whether a personal contact is made or not, to allow chosen calls or avoid disturbances. The connection with the called subscriber will only be established on certain situations, namely if the negotiated communication context has fulfilled certain conditions, which can contain information for example about communication partners and the urgency of requests. This aspect increases the problem of keeping data consistency in often partitioned mobile networks.
While a user crosses cell boundaries, his information - the telemetadata - like location and user profile will be transferred and replicated to the adjacent Base Stations. That way, risks for the very sensitive personal data are increased due to “the multiplication of the points of attack” [7] and the possibly different trust levels afforded by each node. The difficulties will be stronger with respect to different security models.

14.2.3 Data Security in Mobile Database Management and Access
The effects of disconnections as a special resource condition are described above. An often neglected aspect in mobile communication contains the loss of mobile units. They are more likely to get lost than fixed hosts and the consequences are lost data and confidentiality. The only means to prevent loss of confidentiality is the usage of encryption and powerful identification, authentication and access control mechanisms. These are no specific challenges to mobile computing. Just mobile devices are provided only with a very simple protection.
In particular situations, isolated computing without communication and its range of security threats is necessary. But scarce resources like small storage or power capacity could prevent such a computing situation. In addition, scarce resources may cause faulty situations. The system may not be able or the user may renounce from carrying out security methods. Both user or resource driven security can lead to restricted or dismissed protection. A decision instance is required to establish what is to do, or to omit in such a situation. Another problem can consist in a disproportion between the amount of requested data and the available resources, which can lead to a violation of availability or integrity.
14.2.4 Metadata Security in Mobile Database Management and Access

As mentioned above, there is a security threat because of different trust levels of the base stations. In database environments, we have to extend our attention on the one hand from Base Stations to all concerned fixed and mobile hosts and on the other hand to access control models. We have to take into account heterogeneity of access control models (multilevel, discrete, role-based) and heterogeneous integration of data in homogeneous models (apart from heterogeneous security aims and strategies). The same information may be classified different in distinct systems. We will call this effect security model incompatibility.

We indicated tracking user movements as the central mobile security issue. The whereabouts and movements can be taken from the communication overhead or deduced from traffic analysis. But there is also another indirect way to detect them. It is obvious, that mobile users working on databases access data necessary in their current computing environment, e.g. at their current location like the city or the building where they are, dependent on communication partners and so on. The information which data the user has accessed (created, read or modified) at which time make a deduction of his movements possible because of the location dependencies of data. This is a totally new threat we are confronted with in mobile database access.

14.3 SECURITY APPROACHES FOR MOBILE DATABASE ENVIRONMENTS

Now we have spread out a wide range of security problems and challenges. While there is a broad research effort in the area of network security for mobile environments, databases in connection with mobility is rather neglected badly. Even we assume a secure and confidential data transfer there are various database security problems. We will offer in the next chapter safeguards to resolve some problems of data management, access and transfer security. First, we investigate the difference between database systems and security related transparencies. Then we explain location and user movement security and after this we present an approach to answer the dynamic and resource restricted mobile environment.

14.3.1 Transparencies

There are basic security challenges tightening up due to mobility. Included in these challenges is the contradiction of transparencies, the transparency in the database sense against the transparency in the privacy sense. The first one means the user will be relieved from internal system knowledge. For example, he sees his database query and the related result, while the operations in the system like parsing and optimization are hidden from the customer. We can compare it with a view through a window, where the window glass is transparent and not visible. The contrary privacy influenced transparency requires
to expose operations for user views. Users can view the structure and operations in the system to control it. They want to know the nature of the window to find out whether there is a transparent glass, not a mirror, or whether the window distorts the real world behind the glass improperly. Mobile used systems are intended to support the user, to reduce remote query processing and to avoid discrepancies between the amount of data and available resources through intelligent preprocessing and influencing during the processing phase. The management and evaluation of context data is a necessary precondition [9]. These metadata are on the one hand pretty sensible (see also [11]) and on the other hand users have to be granted the right to read and influence context data. This is important since a transparent adaptation of query process is not understandable and can lead to misinterpretations of query results. Moreover, the user must have the possibility to influence the adaptation process (see figure 14.2) carefully. The database transparency needs to be restricted for the benefit of privacy related transparency.

14.3.2 Secure Locations and Movements

The location is a sensible information only in connection with user identities. The best protection of user whereabouts consists in the avoidance of management of location information or user information, respectively. Movement information can be achieved by location information in relation to time information, since movement is defined by changing locations in time. Mobile computing should work as much as possible data thrifty, e.g. as anonymous as possible. Data thrift is a concept in the privacy area and addresses a thrifty management and use of personal data. “Personal data shall mean any information relating to an identified or identifiable natural person” ([4], article 2). The usage of pseudonyms represents a weak kind of data thrift. Since database systems do not support anonymous or pseudonymous computing, pseudonyms must be created either outside of the database system, or users have to act in roles (as described in [11]). We recommend a design of data thrift during the design phase of database systems. In a matrix connecting different levels of data thrift with data and metadata (context), a detailed overview of the necessary and possible data thrift is definable. This model is tested for communication systems in [1].

Let us assume, that data thrift is not applicable. It is possible to deduce the user location

- directly from its management on the network layer and in the mobile context and
- indirectly through traffic analysis and access compromising.

Directly available location information has to be protected with the help of cryptography and suitable access control techniques. In the case of a correctly working system, only the adaptation systems use the location information. Such a mobile context therefore should be accessible only by that system with
exception of user accesses for ensuring privacy intended transparency. Indirect location inquiries can be avoided by means of disguising real information flows. We have described above disguising techniques in data transfer. In database systems, the information flow between sender and receiver is asynchronous because of storing the data in the database between writing and reading them. That is why we are interested in information about data accesses. To avoid deducing whereabouts by accessed data we will consider location dependencies of database data. The protection of user locations against such a compromising attack requires the knowledge of location attributes in the databases. Databases can include

- location attributes like city, address, district, country etc.,
- attributes relating to identifiable locations like special sights (Akropolis, Statue of Liberty, Brandenburg Gate, etc.).

Location dependencies moreover are based on the user context knowledge. Location attributes mostly build a location hierarchy. Assume further, that the location and location referable attributes are well known. We now define

- aggregation separation,
- vertical separation and
- horizontal separation

to achieve the guarding of directly managed whereabouts in the mobile context as well as indirectly managed locations.

**Aggregation Separation** As we mentioned above, location and time information endangers privacy only if it is joined with user identities or information relating to identifiable users. Let \( \{p\} \) be the managed context or audit properties, respectively. Then the separation divides the properties into \( \{u, l, t, r\} \), where \( u \) are user identification properties or contexts, \( l \) the location and identifiable location properties, \( t \) the time information and \( r \) the remaining properties or context information. The protection is obtained by an projective separation.

The aggregation of these data should be accessible just for authorized users. Authorized users are administrators, but with the restriction of vertical separation, and each affected user ("data subject"). The protection is achieved by a separation of user identities and/or location and time. It has to be realized with the help of access control, but also thanks to their physical separation. While in general user identities are simply determined, the establishing of location attributes is a very complex process and needs knowledge discovery methods. General mobility introduced in chapter 1 is a concept to support this separation, thanks to modeled separation of user and location in contexts. The movement of a site does not actually disclose the user movement.
Vertical Separation  The more information a user is accessing, the more complete information about user movements and local activities results. A separation of the personal data is advisable to prevent a generation of an extensive user view. The separation can be vertical, or selective. This means the audit information or mobile contexts, respectively, must be guarded with views based on database selections. The selections split location data for example according to their activated role into accesses to databases, fragments, domains, devices, systems and so on. Consequently, only small sections of user whereabouts are visible.

Horizontal Separation  The requirement of horizontal separation represents a classical database challenge. Data are handled through the underlying operating system and network to store and transfer them. There must not exist an opportunity to undermine the database security through services of underlying layers. In other words, user dependent location information should not cross the database boundaries. The mobile context is of course common for various applications and layers, but should be accessible only in particular views.

14.3.3 Dynamic and Resource Restricted Mobile Environment

Moreover, security and privacy methods are often very static whereas the mobile communication environment is dynamic and necessitates adjustments of queries and results. As we have explained above, automatical adjustment is a basic concept in mobile computing research within the MoVi-project (see [9] and figure 14.2). The adaptation is applied dynamically. Based on the mobile context, the suitable component will be selected.

\[\text{Figure 14.2  Basic Adaptation Concept}\]

The dynamics of mobile database environments arises from the changing mobile context; namely the changing location, dynamic and scarce resources and the varying user and application context. Summarizing, the security and privacy problems caused by the dynamics are:

- heterogeneous access control models on the mobile and the fixed site, heterogeneous integration of information into the same access control model,
- isolated computing,
- no or reduced application of security measures because of scarce resources.
According to the adaptation of database functionality we will try to use the adaptation concept to respond to security problems in the mobile environment.

An access from a database system on the mobile site to a fixed database can raise the problem of heterogeneous access control models. The information is managed for example in a matrix model while the access control model of the accessing mobile site is realized multilevel. A similar problem concerns several introduction of data in the same model. For example, the address of employees are accessible for the personnel department in one database system. In another database, a list of special employees is determined who have access.

These model incompatibilities are not specific for mobile computing. They are characteristic for distributed and especially federated databases. But the very heterogeneous mobile hard- and software preconditions increases the problems. An adaptation process is needed to select the suitable model and to execute a model adaptation. The precondition for an adaptation process is the existence of additional security information. We recommend a pick-a-back security, e.g. a transfer of information about the original access control model, and data integration in it in connection with the data itself. If an adaptation is not possible (the differences of security levels are unbridgeable), the access fails. The adaptation process can ensure that no data will be accessed from or transferred into an unsecured domain. The other favorable effect of an adaptation process is, the burden of security controls is not only of the user alone.

Moreover, an adaptation component could control stand alone computing. This implies to disallow an access to a remote database. A monitoring of the current application is necessary to decide on disconnecting a network connection. The adaptation component has to cooperate with the underlying system layers to realize this task.

Another task for an adaptation process is conceivable, the resource related adaptation. It adjusts the database accesses to the available resources. Small mobile devices have likely frequent a lack of resources. Current techniques are not able to recover from these errors. Another effect is, that the user decides in such a case to perform the intended operation by dismissing security measures. The adaptation process can reduce security methods according to reduced functionality and still maintain a minimal and obligatory security.

14.4 SUMMARY AND CONCLUSIONS

We have described in this paper mobility related security and privacy issues. We grouped the problems by risks for data and metadata in their management, access and transfer. A basic challenge in this environment is to bridge the gap between different transparencies, namely the database-related and the privacy-related transparency. We identified the protection of user whereabouts and movements to be a central threat in mobile environments and propose three kinds of separation in access and management to guard this additional information. In the last chapter we considered with the adaptability concept to respond to dynamic and often scarce resources. We are going to develop
the security adaptation model and to do small implementations to test this concept.

References


X Inference
Abstract:
We apply Bayesian estimation and network techniques to the database inference problem. Bayesian analysis permits the realistic estimation of probabilities of missing data as well as insight into how prior knowledge and observed data interact. We urge our community to exploit this powerful tool.

15.1 INTRODUCTION
We take a Bayesian approach to the database inference problem (inference problem for short). Although the database community has analyzed the inference problem in many different ways (e.g., [5], [10], [9], [14], [18], [20], [22], [28]), other researchers have never formally used Bayesian techniques to study the problem (although several papers e.g., [8], [20], [28] have alluded to this method). Our main contribution is to apply standard Bayesian estimation theory to the inference problem. In particular, we use the method of inductive learning with a Bayesian network e.g., [7]. We analyze the type of inference problems to which our method is applicable.

The ability of a low-level user (Low) to infer higher-level information is the MLS database inference problem. We assume that the low user has the entire low-view of the database at its disposal. If the low database is, in fact, a high database with certain entries blocked out we have the missing data [16] approach to the inference problem (this term is also called incomplete data [27]). The data that is missing is the hidden high data. If the low database
Table 15.1  Simple Relational Database with an Unknown Value

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bond</td>
<td>Russia</td>
<td>Spy</td>
</tr>
<tr>
<td>Smart</td>
<td>France</td>
<td>Cook</td>
</tr>
<tr>
<td>Poirot</td>
<td>France</td>
<td>Cook</td>
</tr>
<tr>
<td>Sandiego</td>
<td>Russia</td>
<td>Spy</td>
</tr>
<tr>
<td>Christie</td>
<td>England</td>
<td>Clerk</td>
</tr>
<tr>
<td>Goldfinger</td>
<td>Russia</td>
<td>Spy</td>
</tr>
<tr>
<td>Holmes</td>
<td>England</td>
<td>Cook</td>
</tr>
<tr>
<td>Ames</td>
<td>Russia</td>
<td>?</td>
</tr>
</tbody>
</table>

is not part of a high database but by Low following implications between the database attributes, it is possible for Low to come up with a new relationship between the data that is, in fact, high information then we are dealing a *logical inference* approach [23]. These are two approaches to the inference problem to which Bayesian techniques can be usefully applied. We only study missing data in this paper and will address the later in future work. We are not concerned with other approaches such as the well-studied statistical/query based approach e.g., [10], [22]¹.

Let us consider the following missing data example put forth by Marks [20]. We assume that Table 15.1 is the low database. The columns are the attributes. The challenge is to determine which job Ames holds, which is the missing value. That is high information since we are assuming that Table 15.1 with the last entry filled in is the high database. The obvious, though simplistic, answer might be to say that Ames is definitely a spy. In fact, this fact seems to be tacitly assumed in some papers, e.g. [20]. Why should we accept this? Do we have enough data upon which to base our decision? If we flip a coin twice, it comes up heads once, and tails once, can we say that it is a fair coin? If the first time I buy a lottery ticket or play a slot machine I win would you give me your home to gamble with on my second attempt? The problem is that we are dealing with a very small sample that is not statistically significant. Certainly, not everyone in Russia is a spy. There must be cooks and there must be clerks. *A fortiori*, if we know that only a small percentage of the people presently in Russia are spies, then we should not assume that Ames is a spy.

¹In the statistical/query based approach the low user is not allowed full knowledge of the database. The complete database is considered high information. The low user is allowed to ask certain questions of the database and/or to know certain statistical information (e.g. max, min, mean, variances, etc). From this partial information the low user then tries to glean high information. Of course, there might be some intersection between the different approaches.
Therefore, we see that deterministic approaches to the inference problem can give skewed results and may be too limited to meet our security concerns. Certainly, a probabilistic leak of high information can be a cause for concern.

We wish to develop a means for analyzing the inference problem when:

Condition 1-The low database may have missing values, and
Condition 2-The low database may have nondeterministic rules and probabilistic relationships between attributes.

The inference problem fits into the bigger scheme of data mining [18], [15], [12]. With this in mind, we use the data mining (learning) technique of Bayesian networks to analyze the inference problem, meeting conditions one and two as above. The example in Table 15.1 is a toy problem that can be expressed as a simple Bayesian network. Before giving formal definitions we will work through the toy problem.

The problem is to determine the job of Ames. We have three choices: Spy, Cook, or Clerk2. In Table 15.1 the first column (name) identifies the sample and designates the row (tuple). We take columns two and three and form a contingency table as shown in Table 15.2.

<table>
<thead>
<tr>
<th></th>
<th>Russia</th>
<th>France</th>
<th>England</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cook</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Spy</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Clerk</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

We take a subjective or "degree of belief" approach to probability. In a belief based approach, we use our prior knowledge of a situation, along with observed data, to arrive at a probability. The prior knowledge is referred to as the prior distribution (simply called the prior), the data on hand is referred to as the observation. The idea of combining the prior and the observation to give us the posterior distribution, by using Bayes' theorem, is called the Bayesian method.

\[ P(E|F) = \frac{P(F|E)P(E)}{P(F)} \]

When attempting to fit observed data to into a probabilistic model (such as what we are doing with the low view of the database), we must be careful not to over-fit the data. Certainly, given two points representing a function, we should not say, in general, that the function is a straight line. Similarly, we can not always assume a deterministic interpretation of the data that is given. The Bayesian approach is very good for this.

2Note we are assuming that there are only three choices of jobs. If there were more (but not represented in the observed data) our Bayesian approach would still work by increasing the number of parameters.
We need to determine \( P(A = i) \), where \( A \) represents the (categorical) random variable “Job of Ames” which can take on the values \( i = \text{cook}, \text{spy}, \text{clerk} \). It is meaningless that the name is Ames. What matters is that there is a Russian with an unknown occupation. The occupations of the French or English are not germane to our analysis. However, we must consider the Russian data. This is why we condition upon it. The event representing the data from the first column of Table 15.2 is represented by \( D_r \). Our goal is to determine \( P(A = \text{spy}|D_r) \). Our prior distribution is the distribution describing the occupation of a Russian. The prior distribution has three values. For a discrete random variable the range values are the parameters. We let \( \theta_1 = P(\text{cook}), \theta_2 = P(\text{spy}), \) and \( \theta_3 = P(\text{clerk}) \). Since probabilities must sum to one, we really only have two parameters (\( \theta_3 \) can be written as: \( \theta_3 = 1 - (\theta_1 + \theta_2) \)). This is, in effect, a second order probability analysis—we are assigning probabilities to probabilities. We decompose \( P(A = \text{spy}) \) as follows (All integration is definite, for notational simplicity, we often do not write out the region of integration): we have

\[
P(A = \text{spy}|D_r) = \int \int \frac{P(A = \text{spy}|D_r, \theta_1, \theta_2)P(D_r|\theta_1, \theta_2)f(\theta_1, \theta_2)}{P(D_r)} \, d\theta_2 \, d\theta_1
\]

[1.1]

What is \( P(D_r|\theta_1, \theta_2) \)? We assume that the occurrences making up Table 15.2 are independent. Therefore, since there are three spies:

\[
P(D_r|\theta_1, \theta_2) = \theta_1 \theta_2 (1 - \theta_1 - \theta_2)^0 = \theta_2^3
\]

[1.2]

Consider the term \( P(A = \text{spy}|D_r, \theta_1, \theta_2) \). This is the probability of \( A = \text{spy} \) conditioned on the data \( D_r \) and the the priors having the values \( \theta_1 \) and \( \theta_2 \). (Note we are abusing notation by sometimes not distinguishing between the random variable describing the prior \( \theta_i \) and the values that the parameter may assume.) This tells us that we are to assume that the parameters are taking on those particular values. Therefore, since the second parameter is the prior for \( \text{spy} \), we have no choice but to assign the conditional probability \( P(A = \text{spy}|\theta_1, \theta_2) \) the value \( \theta_2 \). Also note that the event \( D_r \) does not influence this conditional probability. Therefore, \( P(A = \text{spy}|D_r, \theta_1, \theta_2) = \theta_2 \) also.

15.2 TOY EXAMPLE: NON-INFORMATIVE PRIOR AND TWO DISCRETE PRIOR EXAMPLES

Next, we need to determine the density function \( f(\theta_1, \theta_2) \). We have many choices for what the distribution of these parameters should be. For now we take a non-informative view of the prior and assume that the parameters are jointly uniformly distributed. Consider the parameter \( \theta_1 \). It gives the possible values for the probability of \( A = \text{cook} \). Therefore, \( \theta_1 \) can take on any value between zero and one. Given a value of \( \theta_1 \) the parameter \( \theta_2 \) can be between 0 and \( 1 - \theta_1 \) (of course the third parameter is \( 1 - \theta_1 - \theta_2 \)). Therefore, the integration is taken over the region given by the right triangle \( 0 \leq \theta_1 \leq 1, 0 \leq \theta_2 \leq 1 - \theta_1 \). Since the area of the triangle is \( 1/2 \) we see that \( f(\theta_1, \theta_2) = 2 \). At this point, we do a standard Bayesian trick and treat \( P(D_r) \) as a normalizing
constant $k^{-1}$ and do not calculate it at this time. Therefore Eq. [1.1] simplifies to

$$P(A = \text{spy}|D_r) = 2k \int_0^1 \int_0^{1-\theta_1} \theta_2 \theta_2^3 d\theta_2 d\theta_1 = \frac{k}{15}$$

Now let us consider $P(A = \text{cook}|D_r)$. We see that $P(A = \text{cook}|D_r, \theta_1, \theta_2) = \theta_1$ and we have

$$P(A = \text{cook}|D_r) = \int \int P(A = \text{cook}|D_r, \theta_1, \theta_2)P(D_r|\theta_1, \theta_2)f(\theta_1, \theta_2) d\theta_2 d\theta_1 = \frac{k}{60}$$

Using $P(A = \text{clerk}|D_r, \theta_1, \theta_2) = 1 - \theta_1 - \theta_2$ we see that $P(A = \text{clerk}|D_r) = \frac{k}{60}$. Since $P(A = \text{cook}|D_r) + P(A = \text{spy}|D_r) + P(A = \text{clerk}|D_r) = 1$ we have that $P(D_r) = 1/10$ and therefore $k = 10$. Therefore, $P(A = \text{spy}|D_r) = 2/3$, $P(A = \text{cook}|D_r) = 1/6$, and $P(A = \text{clerk}|D_r) = 1/6$. We could have easily calculated $P(D_r)$ directly in this case since

$$P(D_r) = \int \int P(D_r|\theta_1, \theta_2)P(\theta_1, \theta_2)d\theta_2 d\theta_1 \quad [2.1]$$

What would happen if our data were different? What if there were $n$ spies and no cooks or clerks in first column of the database table? In that case Eq. [1.1] would become

$$P(A = \text{spy}|D_r) = 2k \int_0^1 \int_0^{1-\theta_1} \theta_2^{n+1} d\theta_2 d\theta_1 = \frac{2k}{(n+2)(n+3)} \quad [2.2]$$

where, as before $k^{-1} = P(D_r)$. Since Eq. [2.1] gives us $P(D_r) = 2/[(n+1)(n+2)]$ we see that Eq. [2.2] reduces to $P(A = \text{spy}|D_r) = (n+1)/(n+3)$. This tells us that as $\lim_{n \to \infty} P(A = \text{spy}|D_r) = 1$. This agrees with our intuition—the number of spies is getting larger and larger and still no clerks or cooks appear in the data set. The data set lets us adjust our views on the prior and gives us what we hope is a better guess at the posterior distribution $P(A = \text{spy}|D_r)$.

What if we used a different prior? What would happen if our prior knowledge was in fact definite knowledge? By this we mean that the prior is given by one (non-trivial) value, $P(\theta_1 = a, \theta_2 = b) = 1$. These statements are so strong that they overrule any influence from $D_r$. Obviously with such priors, one would not need to calculate anything. We have no choice but to say that $P(A = \text{spy}|D_r) = b$, regardless of what $D_r$ is. Let us see if our equations give us this result. Recall that the purpose of playing with this toy example is to give us insight into the Bayesian technique. Since we have probability mass functions $P(\theta_1 = i, \theta_2 = j)$ instead of, as before, probability density functions $P(\theta_1, \theta_2)$, the integration becomes summation. Therefore, we now have

$$P(A = \text{spy}|D_r) = \frac{P(A = \text{spy}|D_r, \theta_1 = a, \theta_2 = b)P(D_r|\theta_1 = a, \theta_2 = b)}{P(D_r)} \cdot 1$$

In our initial example with three spies, no cooks or clerks, then both $P(D_r) = P(D_r|\theta_1 = a, \theta_2 = b) = b^3$. Therefore, these two terms cancel out. This holds
true as long as $b \neq 0$. In that case, we would be dividing by zero. In this case, it would also be impossible to have a data set with spies in it! So our mathematics and intuition agree (luckily). So we are left with $P(spy|D_r, \theta_1 = a, \theta_2 = b)$ which is just $b$. So, when there is just one value for the prior, it is also the value for the posterior distribution.

Now let us consider the example where the prior $\theta_2$ can take on two non-trivial values $b$ and $d$, without loss of generality $b < d$. We take as a probability mass function:

$$P(\theta_1 = a, \theta_2 = b) = u, \ P(\theta_1 = c, \theta_2 = d) = 1 - u \text{ where } a \leq 1 - b \text{ and } c \leq 1 - d.$$  

Therefore, we have that $P(A = spy|D_r) = \frac{1}{P(D_r)} \{P(A = spy|D_r, \theta_1 = a, \theta_2 = b)P(D_r|\theta_1 = a, \theta_2 = b)P(\theta_1 = a, \theta_2 = b) + P(A = spy|D_r, \theta_1 = c, \theta_2 = d)P(D_r|\theta_1 = c, \theta_2 = d)P(\theta_1 = c, \theta_2 = d)\}.$

Which gives us $P(A = cook|D_r) = \frac{1}{P(D_r)} \{ub^3 + (1 - u)d^3\}$ and similarly $P(A = clerk|D_r) = \frac{1}{P(D_r)} \{u(1 - a - b)b^3 + (1 - u)(1 - c - d)d^3\}.$

Since all three terms must add to one, we have that $P(D_r) = ub^3 + (1 - u)d^3$. Note that, due to the simplicity of the priors, $P(D_r)$ could have been directly calculated as $P(D_r|\theta_1 = a, \theta_2 = b)P(\theta_1 = a, \theta_2 = b) + P(D_r|\theta_1 = c, \theta_2 = d)P(\theta_1 = c, \theta_2 = d).$ In the next section we will calculate $P(D_r)$ in this latter method. We include the discussion on the normalizing constant because this is a very common way of dealing with Bayesian problems [26]. Therefore,

$$P(A = spy|D_r) = \frac{ub^4 + (1 - u)d^4}{ub^3 + (1 - u)d^3} \quad [2.3]$$

We see by Eq. [2.3] that $P(A = spy|D_r)$ is a function of $u, u \in [0, 1]$ so we will write it as $f(u)$. Obviously $f(0) = d$, and $f(1) = b$. Since $b < d$, we see that the derivative of $f$ w.r.t. $u$ is negative. Therefore, $f(u)$ is a function with a maximum of $d$ and a minimum of $b$. The value of $u$ determining the priors and the data set $D_r$ determine the probability of $P(A = spy|D_r)$ but we know it must be in the range $[b, d]$.

We have played with this toy problem enough for now. Our goal with this exercise was to give the intuition behind the Bayesian method before we presented the full theory. That presentation is the next section.

### 15.3 OUTLINE OF BAYESIAN ANALYSIS

This section is based on work of Anderson [1] and of Heckerman [13].

Bayesian estimation (or prediction) of the value of a random variable $X$ deals with computing the posterior probability of $X$ equaling a certain value, based (conditioned) on the observed data $D$. The general approach is to derive an estimated probability distribution for the random variable based on the available data, and then to obtain the information about a particular value of interest from this derived probability distribution. The probability distribution is, in general, described by a family of parameters $\Theta$. We assume that $X$ is discrete and denote the parameter set by $\theta_1, \ldots, \theta_{|\Theta|}$, where $|\Theta|$ is the number of non-trivial values $X$ may obtain. In other words, the non-trivial values that
X takes are $v_k$ and the parameter corresponding to that value is $\theta_k$. Note that each $\theta_i$ is itself a (usually continuous) random variable. The $\theta_i$ are constrained by the equation $\sum_{i=1}^{\Theta} \theta_i = 1$, since the $\theta_i$ represent probability values. The posterior probability of the $k^{th}$ value $v_k$ of $X$ is

$$P(X = v_k | D) = \int P(X = v_k | \Theta, D) P(\Theta | D) d\Theta.$$  

The total number of independent parameters is $|\Theta| - 1$, because $\sum_{i=1}^{\Theta} \theta_i = 1$. Without loss of generality, we view the last parameter $\theta_{|\Theta|}$ as $1 - \sum_{i=1}^{\Theta-1} \theta_i$. Thus the integral is a $(|\Theta| - 1)$-fold integral, and the region of integration is $0 \leq \sum_{i=1}^{\Theta-1} \theta_i \leq 1$. $D$ can be dropped out from the first term of the last integral because it no longer affects the probability of $v_k$ once the parameters are known. Thus, we have

$$P(X = v_k | D) = \int P(X = v_k | \Theta) P(\Theta | D) d\Theta$$  

[3.1]

To compute the posterior probability of $P(\Theta | D)$, we need $P(D)$, $P(\Theta)$ and $P(D | \Theta)$. This allows us to rewrite Eq. [3.1] as

$$P(X = v_k | D) = \int \frac{P(X = v_k | \Theta) P(D | \Theta) P(\Theta)}{P(D)} d\Theta$$  

[3.2]

Under the Bayesian assumption, each datum is independently drawn and the conditional probability of the data, given parameters, obeys the multinomial distribution. Thus, the likelihood of data is given by $P(D | \Theta) = \prod_{k=1}^{\Theta} \theta_k^{n_k}$ where $n_k$ is the number of samples in $D$ matching the $v_k$ value (compare to Eq. [1.2]).

We use the Dirichlet distribution for the prior probability $P(\Theta)$. $P(\Theta) = \frac{\Gamma(\alpha)}{\prod_{k=1}^{\Theta} \Gamma(\alpha_k)} \prod_{k=1}^{\Theta} \theta_k^{\alpha_k - 1}$ where $\alpha_k > 0$, $\alpha = \sum_{k}^{\Theta} \alpha_k$, and $\Gamma(\cdot)$ is the Gamma function. Note that:

1. When there are only two parameters this is also called the Beta distribution.
2. When $\forall k$, $\alpha_k = 1$ this special Dirichlet distribution becomes a uniform distribution (not over the unit hypercube, since we must account for the Gamma functions). This is called the non-informative prior (as in the toy problem).

Keep in mind that in our Bayesian analysis the “last” parameter $\theta_{|\Theta|}$ is taken to be $1 - \sum_{i=1}^{\Theta-1} \theta_i$.

The use of the Dirichlet distribution (a form “conjugate” to the multinomial distribution) is a standard Bayesian technique for several important reasons [2], [11], [1]. The expected values of the Dirichlet distribution (w.r.t. each parameter) give us a frequentist interpretation of the various coefficients. Our posterior probabilities of $X$ will be in a form that lets us see the influence of the weighting of each prior, via the coefficients $\theta_k$, and the contribution from the observed data. Not all priors are given by the Dirichlet distribution but for this paper, they are.
P(D) is given by (it is basically integration by parts \textit{ad nauseam})

\[ P(D) = \frac{\Gamma(\alpha)}{\Gamma(\alpha + N)} \prod_{k=1}^{|\Theta|} \frac{\Gamma(\alpha_k + n_k)}{\Gamma(\alpha_k)} \]  

where \( N = \sum_{k} n_k \).

Finally, using the fact that \( P(X = v_k|\Theta) = \theta_k \), we arrive at:

\[ P(X = v_k|D) = \frac{\alpha_k + n_k}{\alpha + N} \]  

Eq. [3.4] is very interesting and one of the reasons that the Dirichlet distribution is used (of course we are not advocating using a certain method/technique simply because it results in the “correct” answer). Eq. [3.4] combines two ratios. One ratio is that of the weighting, via the coefficient \( \alpha_k \), of the given prior parameter against the sum of the coefficients \( \alpha \). The second ratio is that of the number of occurrences of the value in question in the observed data against the total number of observations. Applying Eq. [3.4] to our toy problem with the non-informative priors, we have that \( P(A = \text{spy}|D_r) = \frac{4}{10} \) (which is also what we got before!). What if, instead of a uniform prior, we had a generalized Beta distribution? If, when assigning our priors, we have more confidence in “spy” we can show this in the priors by letting \( \alpha_k \) be large. As \( \alpha_k \) increases we see that \( P(A = \text{spy}|D_r) \to 1 \). Similarly, even with the non-informative (uniform) priors if we had 300 data elements and all of them were spies (in the Russian column) we would have that \( P(A = \text{spy}|D_r) = \frac{1 + 300}{1 + 300} = 1 \). In addition, if the observed data has a small amount of non-spy observations, but the number of spies is large, we see that the non-spy observations have a small influence on our posterior probability. Therefore, Eq. [3.4] provides a good intuitive interplay between our assumptions about the prior distribution of the value parameters and the observed data set.

15.4 NETWORK MODEL

We now examine scenarios that are more complicated. We start with a motivating example similar to our toy problem (see Table 15.3).

15.4.1 Example

Now we do not know that Ames is Russian, nor do we know what job he has. We use the random variable \( J \) which represents the jobs that a person from any country (Russian, England, or France) can have. Now we must use the entire contingency table (Table 15.2) since we do not know the country. We want to know the probability of Ames being a spy based on the available data and our estimation techniques. Unfortunately, we now have two missing attributes—Location and Job. Therefore, we must generalize our Bayesian technique from the previous section. This generalization takes us into the area of Bayesian networks [26]. We will work this example and then give the full theory. Our approach is based upon the work done by Cooper [7] in Bayesian learning.
Table 15.3  Simple Relational Database with Two Unknown Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bond</td>
<td>Russia</td>
<td>Spy</td>
</tr>
<tr>
<td>Smart</td>
<td>France</td>
<td>Cook</td>
</tr>
<tr>
<td>Poirot</td>
<td>France</td>
<td>Cook</td>
</tr>
<tr>
<td>Sandiego</td>
<td>Russia</td>
<td>Spy</td>
</tr>
<tr>
<td>Christie</td>
<td>England</td>
<td>Clerk</td>
</tr>
<tr>
<td>Goldfinger</td>
<td>Russia</td>
<td>Spy</td>
</tr>
<tr>
<td>Holmes</td>
<td>England</td>
<td>Cook</td>
</tr>
<tr>
<td>Ames</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Figure 15.1  A Simple Bayesian Network

The term $P(J = \text{spy}|D)$ is what we want. Now $D$ is the data: three Russian spies, two French cooks, one English clerk and one English cook. We decompose $P(J = \text{spy}|D)$ as $P(J = \text{spy}|D) = P(J = \text{spy}, L = R|D) + P(J = \text{spy}, L = F|D) + P(J = \text{spy}, L = E|D)$ where $L$ is the random variable representing country. Note we do not expect this answer to be the same as $P(A = \text{spy}|D_r)$ because we are not sure of the country and are basing our estimations upon the entire observed data.

We form a Bayesian net $B_n$ as shown in Fig. 15.1. The arrow going from $L$ to $J$ tells us that if we change our belief in $L$, we must change our belief in $J$.

Let us start out by calculating $P(J = \text{spy}, L = R|D)$. According to our Bayesian technique, we write this as $P(J = \text{spy}, L = R|D, B_n)$. This emphasizes the net that we are using. The parameter set $\Theta$ now consists of nine parameters. Note that we will (subtly) choose our parameters to be consistent with our underlying net topology. Also, all integration is only over the relevant independent parameters. Unfortunately, no notation seems to capture this in general, so all parameters appear. We have the parameter set consisting of the parameters $\theta_R$, $\theta_E$, and $\theta_F$ for the three different locations. We only concern ourselves with the set $\Theta_L$ made up of $\theta_R$ and $\theta_E$ ($\theta_F = 1 - (\theta_R + \theta_E)$ ). Given a location only two parameters are needed to describe the priors for occupation we have the set $\Theta_J|L$ made up of the six parameters $\theta_{\text{spy}|R}$, $\theta_{\text{cook}|R}$, $\theta_{\text{spy}|E}$, $\theta_{\text{cook}|E}$, $\theta_{\text{spy}|F}$, and $\theta_{\text{cook}|F}$.

$$P(J = \text{spy}, L = R|D, B_n) = \int P(J = \text{spy}, L = R, |\Theta, D, B_n)P(\Theta|D, B_n)d\Theta$$
This follows by the rules of conditioning upon $\Theta$. Recall that when we condition upon both $D$ and $\Theta$ that $\Theta$ subsumes $D$. Thus, we see that the above equals
$$\int P(J = \text{spy}, L = R|\Theta, B_n)P(\Theta|D, B_n)d\Theta .$$
Write $P(J = \text{spy}, L = R|\Theta, B_n)$ as $P(J = \text{spy}|L = R, \Theta, B_n)P(L = R|\Theta, B_n)$. Now we make the local network structure assumption [27] which gives us
$$P(J = \text{spy}|L = R, \Theta, B_n)P(L = R|\Theta, B_n) = P(J = \text{spy}|L = R, \Theta_{J|L}, B_n)P(L = R|\Theta_{L}, B_n).$$
Actually we only use the two parameters $\theta_{\text{spy}|R}$ and $\theta_{\text{cook}|R}$, we do not change our terms for notational simplicity. This assumption makes sense because in each node only the local parameters matter. We can now write the above as
$$\int P(L = R, \mid \Theta_{L}, B_n)P(J = \text{spy}|L = R, \Theta_{J|L}, B_n)P(\Theta_{L}, \Theta_{J|L}|D, B_n)d\Theta_{L} d\Theta_{J|L}$$

Now we make use of the parameter assumption [27]
$$P(\Theta_{L}, \Theta_{J|L}|D, B_n) = P(\Theta_{L}|D, B_n)P(\Theta_{J|L}|D, B_n),$$
this gives us that the above simplifies to
$$= \int P(L = R|\Theta_{L}, B_n)P(\Theta_{L}|D, B_n)d\Theta_{L}$$
$$\cdot \int P(J = \text{spy}|L = R, \Theta_{J|L}, B_n)P(\Theta_{J|L}|D, B_n)d\Theta_{J|L}$$

Thus, we see that the Bayesian network boils down to the non-network formulas that we had before. We make the same assumptions about multinomial sampling and the Dirichlet distribution. The $B_n$ terms have just really come along for the ride. Let us consider the first term
$$\int P(L = R|\Theta_{L}, B_n)P(\Theta_{L}|D, B_n)d\Theta_{L} = \frac{a_R + n_R}{\alpha + N}.$$ The coefficient $\alpha_R$ corresponds to the prior for location Russia, which we assume to be one (non-informative prior). The sum of the location coefficients is $\alpha$, since we are assuming they are all one, this sum is three. $N$ is the total number of observations (7) and $n_R$ is the number of Russians (3). Hence the first integral is 4/10. The second integral follows as before from Eq. [3.4] and is 4/6. So $P(J = \text{spy}, L = R|D) = 8/30$. Similarly we find that $P(J = \text{spy}, L = F|D) = 3/50$, and $P(J = \text{spy}, L = E|D) = 3/50$. Therefore, $P(J = \text{spy}|D) = 58/150 < 2/3$. It is not surprising that it is less than 2/3 because this was $P(A = \text{spy}|D_r)$ and the $D$ data has more non-spies than the $D_r$ data (which has none).

We now present the complete development of Bayesian networks, following [7], [13], [19].

15.5 BAYESIAN NETWORK THEORY

A Bayesian network model is used to deal with samples drawn from an $m$-dimension sample space, with the dimensions corresponding to the attributes. Let $\mathbf{X} \overset{\text{def}}{=} (X_1, \cdots, X_m)$. A sample from a database is an instantiation of the set of attributes and is denoted by the bold face vector $\mathbf{X}$. This is done to simplify the notation; the value of the random vector is implicitly assumed. A
Bayesian network can be viewed as a collection of local networks. Each consists of a child node $X_i$ and its (immediate) parent nodes, $pa_i$. It is an acyclic graph with each node corresponding to an attribute and a (directed) link indicating the conditioning probability, $P(X_i = x|pa_i^j)$. Note that an instantiation of the parent variables, $pa_i^j$, defines a (conditional) probability distribution for $X_i$. (In our previous example, $X_i$ was the random variable $J$ with values $x \in \{R, E, F\}$, and $pa_i$ was the random variable $L$ with the values spy, cook or clerk corresponding to the values of $j$ in the term $pa_i^j$.) An $n$-node network has $n$ local networks. The posterior probability of the entire network is simply the multiplication of posterior probabilities of those local networks.

The Bayesian network model requires two layers of evaluation. At one layer, we evaluate the parameters associated with each local network. The second layer selects the best topological structure for the network $B_n$. We compute the probability distribution for each local component. $\Theta$ is used to denote all parameters. $\theta_i$ is the collection of the parameters associated with the local network that has child node $X_i$. $\theta_{ij}$ is the set of parameters with parent variables of $X_i$ taking the $j^{th}$ instantiation (this is the set of all possible values of the parent nodes). Keep in mind that the set $\{j\}$ depends on which node $X_i$ we are using. To keep the notation at a minimum, we do not write $j$ as a function of $i$, but it should always be kept in mind. $\theta_{ijk}$ is the parameter which associates with the variable $X_i$ that takes its $k^{th}$ value, given that its parent variables are at the $j^{th}$ instantiation. We have that $\theta_{ijk}$ is the distribution modeling $P(X_i^{k}|pa_i^j, \theta_{ij}, B_n)$. For a local network, we have $\sum_{k} K_i \theta_{ijk} = 1$. Notations $J_i$ and $K_i$ stand for the number of instantiations associated with parent variables and the number of different values of variable $X_i$, respectively. So we now examine $P(\tilde{X}|D, B_n)$ because we must also condition upon the underlying net structure,

$$P(\tilde{X}|D, B_n) = \int P(\tilde{X}|\Theta, D, B_n)P(\Theta|D, B_n)d\Theta \quad \text{[5.1]}$$

We assume that at the local network of $X_i$, the probability $P(\theta_{ij}|D, B_n)$ has the Dirichlet distribution. By the parameter independence assumption $P(\Theta|D, B_n)$ is equal to $\prod_{i} \prod_{j} I_{J_i} P(\theta_{ij}|D, B_n)$. The first term of the integral in Eq. [5.1] can be simplified, with the local network structure independence assumption so that Eq. [5.1] can be written as follows (As discussed in the example all integration is only over independent parameter sets. Also the parameters have been chosen to be consistent with $B_n$. If they were not, we would still get the same answer but our integration would be over a different region and the Jacobian from the change of variables would normalize things out.):

$$P(\tilde{X}|D, B_n) = \prod_{i=1}^{n} \int \cdots \int P(X_i^{k}|pa_i^j, \theta_{ij}, B_n)P(\theta_{ij}|D, B_n)d\theta_{ij}, \text{ therefore}$$

$$P(\tilde{X}|D, B_n) = \prod_{i=1}^{n} \int \cdots \int \theta_{ijk}P(\theta_{ij1}, \cdots, \theta_{ij|K_i}|D, B_n)d\theta_{ij1}, \cdots, d\theta_{ij|K_i}$$

$$= \prod_{i=1}^{n} \frac{\alpha_{ijk} + n_{ij k}}{\alpha_{ij} + n_{ij}}$$
15.6 **MISSING VALUES OVER MULTIPLE TUPLES**

Consider the following from Table 15.4 which shows a more complicated situation than what we have previously discussed because we have data missing in more than one tuple (row).

Our approach is largely based on the Gibbs sampling Monte Carlo [25], [3] method where a missing value is repeatedly assigned with a new estimation conditioned on the current values of all other data. The estimation of missing values in our approach includes the following steps: 1- Initialize missing values and the network model. 2- For each attribute, estimate its value with a sample if its value is missing and replace the initial value with this new estimate. 3- Evaluate the network model. 4- Repeat the last two steps for all attributes. 5- Stop when reassignments are not required.

Let $D^I$ and $D^C$ denote the original incomplete database and the database with missing values assigned. The above approach is summarized by:

$$C^{\text{ini}}_{I} = \{D^I, B_n \leftarrow \max_{x_i} P(D^C_{x_i} | B_n)\}$$

where $C^{\text{ini}}_{I}$ means that the value of attribute $X_i$ in sample $C_I$ is missing, and $D^C_{x_i}$ means that $x_i'$ is assigned to $X_i$ at the $i$th attribute for the sample $C_I$.

Details can be found in the presentation version of this paper (available on the web).

15.7 **CONCLUSIONS AND FUTURE WORK**

Our Bayesian method is a double-edged sword (both sides useful). If Low is attempting to determine (probabilistically) high information, it can use Bayesian techniques to guesstimate the correct probability. On the other hand, with knowledge of Bayesian methods, High can introduce spurious data to confound Low's estimation techniques. The idea of padding data is not new. By using the Bayesian formulas, however, we can develop a framework on how to introduce this padded data judiciously, in order to conserve resources. In particular,
we see how the data can influence the final expressions (e.g., the $n_k$ terms) of the posterior probabilities.

We wish to emphasize that our Bayesian techniques certainly call into question assumptions that the given data implies with certainty the occurrence of an event. Of course, the more data observed (low-view) and the more confidence we have in our prior probability distributions, the more we can accurately predict the missing high data. This is best summed up as Small amounts of data imply questionable decisions, while large amounts of data imply better (but still not perfect) predictions. This is certainly not a new thought but one that can be lost amidst elaborate new theories and notations. In this paper (as in [24]), we try to show the advantages of using powerful, well-analyzed methods that already exist in other fields. Bayesian techniques are well-studied and have been successfully used in software debugging and information retrieval [4], [21]. This is close in spirit to the inference problem in MLS database design. Our method is useful when dealing with multiple missing attribute values, and our future work will deal with more complicated databases.

We wish to continue our work by studying more complicated network topologies and determining under what conditions the Bayesian technique may fail. We also wish to explore the High-padding countermeasures discussed above.

We also believe that Bayesian analysis can complement the database search algorithm (e.g., [14]), where a path connecting one entity (e.g., the company table) to another one (e.g., the project table) can be constructed from multiple tables. Once a plausible path is found, Bayesian analysis can carry out inferencing by determining the causal dependency relationships among attributes. This is a logical approach which we feel can complement our statistical approach. We also feel that our approach can also complement the rough sets approach put forth by others [17] [28]. We also feel that a decision tree analysis can be useful in analyzing database inferences (see [6]).

Acknowledgments

We thank Ruth Heilizer along with Judy Froscher, Myong Kang, Carl Landwehr, and Cathy Meadows. We also thank the anonymous referees and workshop participants for their helpful comments. Research supported by the Office of Naval Research.

References


250 DATABASE SECURITY XII


Abstract: Inference is a way to subvert access control mechanisms of database systems. Most existing work on inference detection relies on analyzing functional dependencies in the database schema. This paper is an extension to our earlier effort in developing a data level inference detection system [13]. In this paper, we introduce the split query inference rule, make an extension to the overlapping inference rule, and provide an in depth discussion on the applications of the inference rules on union queries. Data level inference detection is inevitably expensive. We have developed a prototype of the inference detection system to evaluate its performance. The result shows that the system performs better with larger number of attributes and records in the database, and smaller number of projected attributes and return tuples of the queries. Therefore, the inference detection system could be practical when users retrieve a small amount of data compare to the size of the database.

16.1 INTRODUCTION

An inference occurs when a user infers data that the user is not allowed to access. In multilevel secure database systems, early work on inference detection employs a graph to represent the functional dependencies among attributes in the database schema. An inference occurs if there exists two paths between two attributes (or composite attributes), and the two paths are labeled at different
classification levels [5, 1, 10]. The detected inference channel is eliminated by redesigning the database schema [8] or upgrading the paths that lead to the inference [11]. There is also work on incorporating external knowledge in detecting inference [12, 6, 3]. Detecting inference at the schema level is efficient as the detection is performed at the database design time. However, it has two drawbacks. First, the database schema does not capture all dependencies that occur in an instance of the database. Second, the existence of inference paths in the database schema does not necessarily imply the users are making use of them to perform inference.

More recently, researchers look at the instance of the database to generate a richer set of functional dependencies for detecting inference. Hinke et al. use cardinality associations to discover potential inference channels [7]. Hale et al. incorporate imprecise and fuzzy database relations into their inference channel detection system [4]. Marks develops an inference detection system that prevents all possible inference by monitoring user queries with select clauses of the form \( A_i = a_i \), where \( a_i \) is a constant [9]. Chang et al. use Bayesian estimation and network techniques to estimate missing data in the database [2].

In this paper, we describe our effort in developing a data level inference detection system. We have identified six inference rules that users can use to infer data: split query, subsume, unique characteristic, overlapping, complementary, and functional dependence inference rules. Essentially, the six inference rules cover the set-subset, intersection, difference and union relationships among return tuples of queries. These rules are sound and they can be applied in any number of times, and in any order. The existence of these inference rules illustrates the inadequacy of the schema level inference detection approach.

However, data level inference detection is inevitably expensive, as it needs to keep track of all user queries and their return tuples. We have developed a prototype of the data level inference detection system to evaluate its performance. An earlier version of this paper is reported in [13]. In this paper, we introduce the split_query inference rule, make an extension to the overlapping inference rule, provide a detail description on the applications of the inference rules on union queries, and present a more complete experimental results. Because of lack of space, we omit the description of the unique characteristic and functional dependency inference rules. We also omit the use of examples to illustrate the inference rules. Interested readers can find them in [13].

This paper is organized as follows. In Section 2, we introduce the notations used in this paper. In Section 3, we present the inference rules. In Section 4, we discuss the applications of the inference rules on union queries. In Section 5, we outline the inference detection algorithm. In Section 6, we present our experimental results. In Section 7, we give a summary of the paper.

### 16.2 NOTATIONS

We consider a relational database that contains a single table. Multiple tables can be modeled as a universal relation as discussed in [9]. \( t[A_i] \) denotes the
A DATA LEVEL INFERENCE DETECTION SYSTEM

attribute value of the tuple $t$ over the attribute $A_i$. A query is represented by a 2-tuple: $(projected-attributes; selection-criterion)$, where $projected-attributes$ is the set of attributes projected by the query, and $selection-criterion$ is the logical expression that selects the return tuples of the query. No aggregation function (for example, maximum and average) is allowed to apply on the projected-attributes. Given a query $Q_i$, $|Q_i|$ denotes the number of return tuples of $Q_i$, and $\{Q_i\}$ denotes the set of return tuples of $Q_i$. Unless otherwise stated, a set of return tuples is indeed a multiset of return tuples, that is, duplicated return tuples are retained. For each query $Q_i = \{AS_i; SC_i\}$, $AS_i$ is expanded with $A_i$ when $'A_i = a_i'$ appears in $SC_i$ as a conjunct. An inferred query is a query that a user can infer its return tuples without directly issuing it to the database. A partial query $Q_i$ is a query that a user knows about $|Q_i|$ but not all the return tuples of $Q_i$. \(\cap\), \(\cup\), and "\-' stand for the set intersection, union, and difference operations respectively.

A tuple $t$ projected over a set of attributes $S$ satisfies a logical expression $E$ if $E$ is evaluated to true when each occurrence of $A_i$ in $E$ is replaced with $t[A_i]$, for all $A_i$ in $S$. $t$ contradicts $E$ if $E$ is evaluated to false. A return tuple $t_i$ of a query $Q_i$ is indistinguishable from another return tuple $t_j$ of $Q_j$ if 1) $t_i[A] = t_j[A]$ for each attribute $A \in (AS_i \cap AS_j)$, 2) $t_i$ does not contradict $SC_j$, and 3) $t_j$ does not contradict $SC_i$. A tuple $t_i$ relates to another tuple $t_j$ if the two tuples are projected from the same tuple in the database. If $t_i$ relates to $t_j$, then $t_i$ is indistinguishable from $t_j$; but the reverse does not necessary hold. Given two queries, $Q_1$ and $Q_2$, we say that $Q_1$ is subsumed by $Q_2$, denoted as $Q_1 \sqsubseteq Q_2$, if and only if 1) $SC_1$ logically implies $SC_2$ (denoted as $SC_1 \Rightarrow SC_2$), or 2) for each return tuple $t_1$ of $Q_1$, $t_1$ satisfies $SC_2$. \(\sqsubseteq\) is a reflexive, anti-symmetric, and transitive relation.

The goal of our inference detection system is to detect if a user can infer data using a series of queries. In particular, the system determines if a user can infer a return tuple of a query relates to a return tuple of another query. If so, the user can learn more about the return tuples.

16.3 INFRINGEMENT RULES

In this section, we present four inference rules. Unless otherwise stated, all queries appear in the inference rules are not partial queries. We assume all the queries are issued by a single user, and there is no change to the database content. When two users are suspected of cooperating in performing inference, we run the inference detection system against their combined set of queries.

16.3.1 Split Queries

A query $Q_i$ can be split into two smaller queries when a user can identify the return tuples of $Q_i$ that relate to the return tuples of another query.

Inference Rule 1 (Split Queries) Given two queries $Q_1$ and $Q_2$. Express $SC_2$ in disjunctive normal form. If there exists a disjunct of $SC_2$ such that the set of attributes appear in the disjunct is a subset of $AS_1$, then generate two
inferred queries: 1) $Q_{11} = (AS_1; SC_1 \wedge SC_2)$; and 2) $Q_{12} = (AS_1; SC_1 \wedge \neg SC_2)$. $Q_2$ may be a partial query. The return tuples of $Q_{11}$ are the return tuples of $Q_1$ that also satisfy $SC_2$. The return tuples of $Q_{12}$ are the return tuples of $Q_1$ that does not satisfy $SC_2$.

When $Q_1$ projects all attributes that appear in a disjunct of $SC_2$, a user can identify the return tuples of $Q_1$ that satisfy $SC_2$. Hence, the user can divide the return tuples of $Q_1$ into two sets: those that satisfy both $SC_1$ and $SC_2$, and those that satisfy $SC_1$ but not $SC_2$.

16.3.2 Subsume Inference

In this section, we describe inference making use of the subsume relations among queries.

**Inference Rule 2 (Subsume)** Given two queries $Q_1$ and $Q_2$, such that $Q_1 \sqsubseteq Q_2$.

**SI1** If there is an attribute $A$ in $(AS_2 \setminus AS_1)$, such that all return tuples of $Q_2$ take the same attribute value $a$ over $A$, then for each return tuple $t_1$ of $Q_1$, $t_1[A] = a$. $Q_1$ may be a partial query.

**SI2** If a return tuple $t_1$ of $Q_1$ is indistinguishable from exactly one return tuple $t_2$ of $Q_2$, then $t_1$ relates to $t_2$. $Q_1$ may be a partial query.

**SI3** Let $S$ be the set of return tuples of $Q_2$ that are distinguishable from the return tuples of $Q_1$. If $|S| = (|Q_2| - |Q_1|)$, generate two inferred queries from $Q_2$: 1) $Q_{21} = (AS_2; SC_2 \wedge \neg SC_1)$ with $S$ as the set of return tuples; and 2) $Q_{22} = (AS_2; SC_2 \wedge SC_1)$ with $\{Q_2\} \setminus S$ as the set of return tuples. If $|S| < (|Q_2| - |Q_1|)$, generate an inferred partial query: $Q_{23} = (AS_2; SC_2 \wedge SC_1)$ with $S$ as the partial set of return tuples, and $|Q_{23}| = (|Q_2| - |Q_1|)$.

$Q_1 \sqsubseteq Q_2$ implies that for each return tuple $t_1$ of $Q_1$, there is a return tuple $t_2$ of $Q_2$ such that $t_1$ relates to $t_2$. **SI1** says that when all return tuples of $Q_2$ share a common attribute value, say $a$, over an attribute $A$, a user can infer that each return tuple of $Q_1$ also takes the attribute value $a$ over the attribute $A$. This is because for each return tuple $t_1$ of $Q_1$, no matter which return tuple $t_2$ of $Q_2$ that relates to $t_1$, $t_2[A] = a$. Hence, $t_1[A]$ must be equal to $a$.

**SI2** says that if $t_1$ of $Q_1$ is indistinguishable from exactly one return tuple $t_2$ of $Q_2$, then $t_1$ relates to $t_2$. This is because $Q_1 \sqsubseteq Q_2$ implies that there is at least one return tuple of $Q_2$ that is indistinguishable from each return tuple of $Q_1$. Now, if $t_1$ of $Q_1$ is indistinguishable from one and only one return tuple $t_2$ of $Q_2$, then we can conclude that $t_1$ relates to $t_2$.

**SI3** says that if a user identifies all the return tuples of $Q_2$ that relate to the return tuples of $Q_1$, then the user can infer these two queries from $Q_2$: $(AS_2; SC_1 \wedge SC_2)$ which includes return tuples of $Q_2$ that relate to the return tuples of $Q_1$, and $(AS_2; SC_2 \wedge \neg SC_1)$ which includes return tuples of $Q_2$ that do not relate to the return tuples of $Q_1$. 
16.3.3 Overlapping Inference

In this section, we describe the overlapping inference rule.

Inference Rule 3 (Overlapping)

OI1 Given $Q_1 \sqsubseteq Q_2$, and $Q_1 \sqsubseteq Q_3$. Let $S_2$ be the set of return tuples of $Q_2$ that are indistinguishable from the return tuples of $Q_3$. If $|S_2| = |Q_1|$, and a return tuple $t_2$ of $Q_2$ is indistinguishable from exactly one return tuple $t_3$ of $Q_3$, then $t_2$ relates to $t_3$. Similarly, let $S_3$ be the set of return tuples of $Q_3$ that are indistinguishable from the return tuples of $Q_2$. If $|S_3| = |Q_1|$, and a return tuple $t_3$ of $Q_3$ is indistinguishable from exactly one return tuple $t_2$ of $Q_2$, then $t_3$ relates to $t_2$. Suppose $|Q_1| = |S_2| = |S_3|$. If a return tuple $t_1$ of $Q_1$ is indistinguishable from exactly one return tuple $t_2$ in $S_2$, then $t_1$ relates to $t_2$. Also, if $t_1$ is indistinguishable from exactly one return tuple $t_3$ in $S_3$, then $t_1$ relates to $t_3$. $Q_1$ may be a partial query.

OI2 Given a query $Q_1$, and a set of queries, $QS = \{Q_2, \ldots, Q_n\}$, where $n \geq 3$. Suppose $SC_1 \equiv (SC_2 \lor \ldots \lor SC_n)$, and for each $Q_i$ in $QS$, $Q_i \sqsubseteq Q_1$. If the number of distinguishable tuples in $QS = |Q_1|$, then any pair of indistinguishable tuples relate to each other.

OI3 When OI1 is applied and all the related return tuples between $Q_2$ and $Q_3$ have been identified, generate the following two inferred queries from $Q_2$: 1) $Q_{21} = (AS_2; SC_2 \land \neg SC_3 \land \neg SC_1)$ with $\{Q_2\}\backslash S_2$ as the set of return tuples; and 2) $Q_{22} = (AS_2; SC_2 \land SC_3)$ with $S_2$ as the set of return tuples. Similarly generate two inferred queries from $Q_3$. When OI2 is applied, generate possibly four inferred queries for each pair of queries that have overlapping return tuples.

Given that $Q_1 \sqsubseteq Q_2$ and $Q_1 \sqsubseteq Q_3$, the number of return tuples of $Q_2$ that relate to return tuples of $Q_3$ must be at least $|Q_1|$. OI1 identifies the cases where a user can infer the related return tuples among the three queries. When $Q_1$ implies three or more queries, OI1 is applied to two of them at a time.

We illustrate OI2 using three queries, $Q_1$, $Q_2$, and $Q_3$, where $Q_1 \sqsubseteq Q_3$, $Q_2 \sqsubseteq Q_3$, and $SC_3 \equiv SC_1 \lor SC_2$. Let $N$ be the number of indistinguishable tuples in $Q_1$ and $Q_2$. As $SC_3 \equiv SC_1 \lor SC_2$, each return tuple of $Q_3$ relates to a return tuple in $Q_1$ or $Q_2$. Hence, $N \geq |Q_3|$. Furthermore, as $Q_1 \sqsubseteq Q_3$ and $Q_2 \sqsubseteq Q_3$, each distinguishable tuple in $Q_1$ and $Q_2$ relates to a return tuple of $Q_3$. Hence, $N \leq |Q_3|$. Therefore, $N = |Q_3|$. When a user find out that the number of indistinguishable tuples in $Q_1$ and $Q_2$ equals $|Q_3|$, the user can infer that for each return tuple $t_1$ of $Q_1$ that is indistinguishable from a return tuple $t_2$ of $Q_2$, $t_1$ relates to $t_2$.

16.3.4 Complementary Inference

The complementary inference rule performs inference by eliminating tuples that are not related to one another.
Inference Rule 4 (Complementary Inference) Given four queries, $Q_1$, $Q_2$, $Q_3$, and $Q_4$, where $Q_1 \subseteq Q_2$, and $Q_3 \subseteq Q_4$. Also, the return tuples of $Q_1$ that relate to the return tuples of $Q_3$ are identified (for example using the overlapping inference rule), and the return tuples of $Q_2$ that relate to the return tuples of $Q_4$ are identified. If one of the following three conditions holds,

1. for each return tuple $t_1$ of $Q_1$ that does not relate to any return tuple of $Q_3$, $t_1$ is distinguishable from all return tuples of $Q_4$,

2. $Q_4 \sqsubseteq Q_3$, or

3. $|Q_3| = |Q_4|$, 

then $Q'_1 \sqsubseteq Q'_2$, where $Q'_1 = (AS_1; SC_1 \land \neg SC_3)$, and $Q'_2 = (AS_2; SC_2 \land \neg SC_4)$. \{$Q'_1$\} is the set of return tuples of $Q_1$ that do not relate to any return tuple of $Q_3$, and \{$Q'_2$\} is the set of return tuples of $Q_2$ that do not relate to any return tuple of $Q_4$.

As $Q_1 \sqsubseteq Q_2$ and \{$Q'_1$\} \subset \{Q_1\}, each return tuple of $Q'_1$ relates to a return tuple of $Q_2$. Condition (1) says that each return tuple of $Q'_1$ does not relate to any return tuple of $Q_4$. Hence, each return tuple of $Q'_1$ relates to a return tuple of $Q'_2$. Condition (2) or (3) implies \((Q_3 \sqsubseteq Q_4) \land (Q_4 \sqsubseteq Q_3)\). By removing from $Q_1$ and $Q_2$ the “same” set of return tuples, we have $Q'_1 \sqsubseteq Q'_2$.

It should be noted that in some cases, the inference as obtained from the complementary inference rule can also be obtained from the overlapping inference rule. For example, consider four queries $Q_1$, $Q_2$, $Q_3$, and $Q_4$, where $Q_1 \sqsubseteq Q_2$, and $Q_3 \sqsubseteq Q_4$. Suppose the overlapping inference rule can be applied to identify the related tuples between $Q_1$ and $Q_3$, and between $Q_2$ and $Q_4$. These result in the generation of two inferred queries: 1) $Q'_1 = (AS_1; SC_1 \land \neg SC_3)$; and 2) $Q'_2 = (AS_2; SC_2 \land \neg SC_4)$. If \((SC_1 \land \neg SC_3) \Rightarrow (SC_2 \land \neg SC_4)\), then we have $Q'_1 \sqsubseteq Q'_2$ which is the same result as obtained by applying the complementary inference rule to the four queries. However, $SC_1 \Rightarrow SC_2$ and $SC_3 \Rightarrow SC_4$ does not necessary implies \((SC_1 \land \neg SC_3) \Rightarrow (SC_2 \land \neg SC_4)\). When this implication does not hold, the complementary inference rule is needed to perform the inference.

16.4 INFEERENCE WITH UNION QUERIES

The inference rules can be applied to unions of queries. We call a union of queries a ‘union query’. In contrast, a user query or an inferred query is called a ‘simple query’. If $Q_u$ is a union query consists $Q_1$, …, and $Q_j$, then $AS_u = (AS_1 \cap \ldots \cap AS_j)$, and $SC_u = (SC_1 \lor \ldots \lor SC_j)$. Note that $AS_u$ might be equal to $\emptyset$. The applications of the split query, unique characteristic and functional dependency inference rules on union queries are similar to their applications on simple queries. Hereafter, we only discuss the applications of the subsume, overlapping, and complementary inference rules on union queries.
16.4.1 Subsume Inference Rule on Union Queries

Consider the applications of the subsume inference rule on union queries when the union queries are subsumed by other queries. Let $Q_u = \{Q_i, \ldots, Q_j\}$ be a union query, and $Q_u \sqsubseteq Q_1$. We show that inference obtained by applying the subsume inference rule on $(Q_i \cup \ldots \cup Q_j) \sqsubseteq Q_1$ can also be obtained by applying the subsume inference rule on $Q_i \sqsubseteq Q_1, \ldots, Q_j \sqsubseteq Q_1$.

Consider the applications of $SI1$. If there is an attribute $A$ in $(AS_1 \setminus AS_u)$, such that all return tuples of $Q_1$ take the same attribute value $a$ over $A$, then for each return tuple $t_u$ of $Q_u$, $t_u[A] = a$. This implies that for each return tuple $t$ of a simple query of $Q_u$, $t[A] = a$. This is the same as if the $SI1$ is applied to $Q_1$ and $Q_1$, where $Q_i \sqsubseteq Q_1$, for each simple query $Q_i$ of $Q_u$.

Consider the applications of $SI2$. If there exists a tuple $t_u$ in $Q_u$ that is indistinguishable from exactly one return tuple $t_1$ of $Q_1$, there exists at least one simple query $Q_i$ of $Q_u$ such that $t_u$ relates to a return tuple $t_1$ of $Q_i$. Now, $t_i$ is indistinguishable from $t_1$ of $Q_1$. Hence, when $SI2$ is applicable to infer that $t_u$ of $Q_u$ relates to $t_1$ of $Q_1$, it is also applicable to infer that $t_i$ of $Q_i$ relates to $t_1$ of $Q_1$.

Consider the applications of $SI3$. When all the related tuples between $Q_u$ and $Q_1$ are identified, two inferred queries are generated from $Q_1$: 1) $Q_{u1} = (AS_1; SC_1 \land \neg SC_u)$; and 2) $Q_{u2} = (AS_1; SC_1 \land SC_u)$. We show that these two queries can also be generated from the simple queries of $Q_u$ and $Q_1$. Note that when all the related tuples between $Q_u$ and $Q_1$ have been identified, all related tuples among the simple queries of $Q_u$ are also identified. Without loss of generality, suppose $Q_u = \{Q_2, Q_3\}$. The application of $SI3$ on $Q_1$ and $Q_2$ generates two inferred queries: 1) $Q_{21} = (AS_1; SC_1 \land \neg SC_2)$; and 2) $Q_{22} = (AS_1; SC_1 \land SC_2)$. Similarly, the application of $SI3$ on $Q_1$ and $Q_3$ generates two inferred queries: 1) $Q_{31} = (AS_1; SC_1 \land \neg SC_3)$; and 2) $Q_{32} = (AS_1; SC_1 \land SC_3)$. Now, $Q_{21}$ and $Q_{31}$ are both generated from $Q_1$, and we can generate the following inferred query for their related tuples: $(AS_1; SC_1 \land \neg SC_2 \land \neg SC_3)$ which equals $Q_{u1}$. $Q_{22}$ and $Q_{32}$ are both generated from $Q_1$, and we can identify the related tuple between them. The union of these two queries is $(AS_1; SC_1 \land (SC_2 \lor SC_3))$ which equals $Q_{u2}$. Therefore, we do not need to consider the applications of the subsume inference rule when the union query is subsumed by other queries.

Consider the case where union queries subsume other queries, say $Q_1 \sqsubseteq Q_u$. $SI1$ is applied as follows. If for each return tuple $t$ of any simple query of $Q_u$, $t[A] = a$, then $t_i[A] = a$ for each return tuple $t_1$ of $Q_1$. $SI2$ is applied as follows. If there is a return tuple $t_1$ of $Q_1$ that is indistinguishable from a set of return tuples $S$ from the simple queries of $Q_u$, where all tuples in $S$ relate to one another, then $t_1$ relates to each tuple in $S$. $SI3$ is applied similarly. Note that the subsume inference rule can still be applied when the simple queries of $Q_u$ have no common projected attribute.
16.4.2 Overlapping and Complementary Inference Rule on Union Queries

Consider the applications of \( O11 \). Given three queries, \( Q_1, Q_2, \) and \( Q_u \), where \( Q_u \) is a union query. Suppose \( Q_u \subseteq Q_1 \) and \( Q_u \subseteq Q_2 \). If \( O11 \) is to be applied to identify the related return tuples among \( Q_2 \) and \( Q_3 \), \( |Q_u| \) must be known. That is, the number of related tuples, if any, between the simple queries are identified. Now, suppose \( Q_1 \subseteq Q_u \) and \( Q_1 \subseteq Q_2 \). If \( O11 \) is to be applied to identify the related return tuples between \( Q_u \) and \( Q_2 \), then the user must has already identified those related tuples among the simple queries in \( Q_u \). Also, the user has to identify the return tuples of \( Q_u \) that are indistinguishable from the return tuples of \( Q_2 \), and the number of these return tuples equals \( |Q_1| \).

Consider the applications of \( O12 \). Suppose there is a set of queries \( QS = \{Q_2, \ldots, Q_n, Q_u\} \) such that for each query \( Q_i \in QS, Q_i \subseteq Q_1 \). \( O12 \) is applicable when the related tuples among the queries in \( QS \) are identified. That is, the related return tuples, if any, between \( Q_u \) and other queries in \( QS \) have to be identified. \( O13 \) is applied similar to the case with simple queries.

Note that the overlapping inference rule can still be applied when \( AS_u = \emptyset \). For example, let \( Q_u = \{Q_{u1}, Q_{u2}\} \). If \( SC_{u1} \land SC_{u2} = false \), the user can conclude that there is no related return tuple between \( Q_{u1} \) and \( Q_{u2} \), and \( |Q_u| = |Q_{u1}| + |Q_{u2}| \).

Consider the applications of the complementary inference rule on the union queries. Suppose there are four queries \( Q_1, Q_2, Q_3, \) and \( Q_u \), where \( Q_u \) is a union query, \( Q_1 \subseteq Q_2 \), and \( Q_3 \subseteq Q_u \). To apply the complementary inference rule on these four queries, the related return tuples among the simple queries in \( Q_u \) that also relate to return tuples of \( Q_2 \) must have been identified. Similarly for the case when \( Q_1, Q_2, \) or \( Q_3 \) is a union query.

16.5 Inference Detection Algorithms

In this section, we outline the inference detection algorithms. Figure 16.1 shows the main function \( INFERENCE(U, Q_i) \), which is called each time a user \( U \) issues a query \( Q_i \) to the database. The function maintains two sets: \( GEN \) and \( EXP \). \( GEN \) is initialized with the user issued query \( Q_i \), and is subsequently being added with inferred queries generated by the inference rules. Each query in \( GEN \) is compared with previously issued or inferred queries for user \( U \) (denoted as \( PREV.QUERY(U) \)) to determine if the inference rules are applicable to them. \( EXP \) is the set of tuples that are expanded during the applications of the inference rules. The results of the applications of inference rules are generations of inferred queries and expansions of some return tuples of queries. Given a tuple \( t_1 \) projected over a set of attributes \( AS_1 \), and another tuple \( t_2 \) projected over a set of attributes \( AS_2 \). If \( t_1 \) and \( t_2 \) are found to be related to each other, \( t_1 \) is expanded as follows: for each attribute \( A \in AS_2 \setminus AS_1 \), \( t_1[A] = t_2[A] \). \( t_2 \) is expanded similarly.

After a tuple is expanded, the query that returns the expanded tuple might be eligible in further applications of inference rules. Hence, the function checks if the inference rules are applicable to the query. \( INFERENCE \) is a terminating
function, as the number of inferences is bound by the size of the database. In each call to the INFEERENCE function, all queries in GEN are processed before the expanded tuples in EXP. This avoids repeatedly processing the same tuple which is expanded more than once after queries in GEN are processed.

**INFEERENCE** \((U, Q_i)\):

1. initialize \(GEN\) with \(Q_i\);
2. \(EXP \leftarrow \emptyset\);
3. \(GEN.Q \leftarrow \emptyset\);
4. \(EXP.Q \leftarrow \emptyset\);
5. while \((GEN \neq \emptyset \text{ or } EXP \neq \emptyset)\) do
   6. if \(GEN \neq \emptyset\) then
      7. \(Q_j \leftarrow\) a query in \(GEN\);
      8. remove \(Q_j\) from \(GEN\)
      9. \(GEN.Q \leftarrow GEN.Q \cup \{Q_j\}\);
   10. else if \(EXP \neq \emptyset\) then
       11. \(Q_j \leftarrow\) a query that returns a tuple in \(EXP\);
       12. \(EXP.Q \leftarrow EXP.Q \cup \{Q_j\}\);
       13. \(ts \leftarrow\) return tuples of \(Q_j\) in \(EXP\);
       14. remove return tuples of \(Q_j\) from \(EXP\);
       15. for each \(Q_k \in PREV\_QUERY(U)\) do
           16. \(EXP \leftarrow UNIQUE(Q_j, Q_k, ts, EXP)\);
           17. \(GEN \leftarrow SPLIT\_QUERY(Q_j, Q_k, GEN)\);
       18. if \(Q_j \sqsubseteq Q_k\) then
           19. \((GEN, EXP) \leftarrow SUBSUME(Q_j, Q_k, GEN, EXP)\);
           20. \((GEN, EXP) \leftarrow OVERLAP(U, Q_j, Q_k, GEN, EXP)\);
           21. \(GEN \leftarrow COMPLEMENTARY(Q_j, Q_k, GEN)\);
       22. else if \(Q_k \sqsubseteq Q_j\) then
           23. \((GEN, EXP) \leftarrow SUBSUME(Q_k, Q_j, GEN, EXP)\);
           24. \((GEN, EXP) \leftarrow OVERLAP(U, Q_k, Q_j, GEN, EXP)\);
           25. \(GEN \leftarrow COMPLEMENTARY(Q_k, Q_j, GEN)\);
       26. \(FIND\_UNION(U, GEN.Q, EXP.Q)\);

**Figure 16.1** The inference function.

The function \(UNIQUE\) has three input parameters: \(Q_j\), \(Q_k\), and \(ts\). The function checks if unique characteristic can be determined between the two queries \(Q_j\) and \(Q_k\). For each expanded return tuple in \(ts\), the function checks if the expanded return tuple and another return tuple have common unique characteristics. If so, the two return tuples are expanded with each other. The functions \(SPLIT\_QUERY\), \(SUBSUME\), \(OVERLAP\), and \(COMPLEMENTARY\) operate as described in the corresponding inference rules, and we omit the presentations of their algorithms. The \(FIND\_UNION\) function checks if there are unions of query that satisfy the subsume relations with other queries. If so, the inference rules are applied to them.
16.6 EXPERIMENTAL RESULTS

We have developed a prototype of the inference detection system in about 4,000 lines of Perl code. We have implemented the split query, subsume, unique characteristic, overlapping (except OI1), and complementary inference rules. The system also handles applications of the inference rules on union queries. We run our experiments with randomly generated tables and user queries. Each table has $N_{\text{attr}}$ number of attributes, and $N_{\text{rec.num}}$ number of records. The primary key of the table is a single attribute. All attributes are of integer types. Attribute values in the table are uniformly distributed between 0 and $(N_{\text{data.dist}} \times N_{\text{rec.num}})$, where $0 < N_{\text{data.dist}} \leq 1$. We also randomly generate $N_{\text{query.num}}$ number of user queries. Each query projects $N_{\text{proj}}$ number of attributes from the table. The selection criterion of each query is a conjunction of $N_{\text{cond}}$ number of conjuncts. Each conjunct is of the form ‘$A_i \ op \ a_i$’, where $A_i$ is an attribute from the table, $\op$ is one of the comparison operators ($>$, $\geq$, $\leq$, $<$, and $=$), and $a_i$ is an attribute value. Each query has $N_{\text{ret.tuple}}$ number of return tuples. We approximate the evaluation of a logical implication $C_i \Rightarrow C_j$ by checking if the tuples selected by $C_i$ is also selected by $C_j$, and that the set of attributes appear in $C_j$ is a subset of those appear in $C_i$. We collect the following two data to measure the system performance: 1) average number of seconds used to process one query. 2) number of times the inference rules are applied.

We ran six experiments to determine how the characteristics of the database and the queries affect the system performance. For the database, we consider the following characteristics: 1) the number of tuples in the database; 2) the number of attributes in the database; and 3) the amount of duplication of the data values. For the queries, we consider the following characteristics: 1) the number of attributes projected by the queries; 2) the number of conjuncts in the selection criteria; 3) the number of queries being issued; and 4) the number of tuples returned by the queries. The experimental results of running the inference detection system on a Sun SPARC 20 workstation are shown in Figure 16.2–Figure 16.7.

Experiment 1 investigates the effect of the number of attributes and the amount of data duplication in the database on the system performance. In this experiment, we choose the following parameter values: $N_{\text{rec.num}} = 1000$, $N_{\text{ret.tuple}} = 50$, $N_{\text{proj}} = 4$, $N_{\text{cond}} = 3$, and $N_{\text{query.num}} = 500$. $N_{\text{attr}}$ is varied with the following values: 40, 60, 80, 100, 120, and 140. $N_{\text{data.dist}}$ is varied with the following values 25%, 50%, 75%, and 100%. Figure 16.2 shows the results in a graph plotted with the average query processing time (in seconds) against the number of attributes in the database. Consider each individual line in Figure 16.2. It shows that the system runs faster as $N_{\text{attr}}$ increases from 40 to 140. With a fixed type of queries, the larger the number of attributes in the table, the lesser the amount of overlapping among the return tuples of queries. This results in lesser subsume relations hold among queries, and hence the smaller the number of inferences.
Consider the four lines in Figure 16.2. They correspond to the cases where \(N_{data\_dist} = 25\%, 50\%, 75\%, \text{ and } 100\%.\) The lower the value of \(N_{data\_dist},\) the more duplication of the data in the database. Intuitively, the higher the duplication of the data, the lesser the number of distinguishable return tuples, and hence the smaller number of inferences. This is true in some cases. However, in general the results do not show a significant effect of data duplication on the system performance.

Experiment 2 investigates the effect of the number of return tuples of queries on the system performance. Figure 16.3 shows the results for \(N_{rec\_num} = 1000, \ N_{data\_dist} = 50\%, \ N_{proj} = 4, \ N_{cond} = 3, \text{ and } N_{query\_num} = 500.\) \(N_{ret\_tuple}\) takes the values of 50, 100, 150, 200, and 250, and \(N_{attr}\) takes the values of 80 and 120. The figure shows that the system runs slower as \(N_{ret\_tuple}\) increases. The larger the number of return tuples, the longer it takes for the system to process them. Also, the more the number of tuples returned by the queries, the more the number of occurrences of inferences, and also the more the number of inferred queries being generated.

Experiment 3 investigates the effect of the number of projected attributes in queries on the system performance. Figure 16.4 shows the results for \(N_{rec\_num} = 1000, \ N_{query\_num} = 500, \ N_{data\_dist} = 50\%, \ N_{attr} = 80, \text{ and } N_{ret\_tuple} = 50.\) \(N_{proj}\) takes the values of 4, 5, 6, 7, and 8. \(N_{cond}\) takes the values of 4, 5, 6, and 7. It shows that the system runs slower as \(N_{proj}\) increases. This is because the more the number of attributes projected by the queries, the more overlapping among the return tuples of queries, and hence the more number of inferences.

Experiment 4 investigates the effect of the number of conjunts in the selection criteria on the system performance. Figure 16.5 shows the results for \(N_{rec\_num} = 1000, \ N_{query\_num} = 500, \ N_{data\_dist} = 50\%, \ N_{attr} = 80, \text{ and } N_{ret\_tuple} = 50.\) \(N_{cond}\) takes the values of 3, 4, 5, 6, and 7. \(N_{proj}\) takes the values of 4, 5, 6, and 7. It shows that the system runs faster as \(N_{cond}\) increases. This is because the larger the number of conjunts in the selection criteria of the queries, the lesser the chance that the subsume relations hold among the queries, and hence the smaller number of occurrences of inferences. However, the effect is not significant when \(N_{cond} > 3.\)

Experiment 5 investigates the effect of the number of tuples in the database on the system performance. Figure 16.6 shows the result for \(N_{data\_dist} = 50\%, \ N_{attr} = 80, \ N_{ret\_tuple} = 50, \ N_{query\_num} = 500, \ N_{proj} = 4, \text{ and } N_{cond} = 3.\) \(N_{rec\_num}\) is varied with the following values: 1000, 2500, 5000, 7500, and 10000. It shows that the system runs faster as the number of tuples of the database increases. As the size of the database increases, the possible amount of overlapping among the queries decreases, and hence the lesser number of inferences. For \(N_{ret\_tuple} = 10000,\) the set of queries happen to generate more inferences than the case for \(N_{ret\_tuple} = 5000\) or 7500, and hence it has a longer running time.

Experiment 6 investigates the effect of the number of queries on the system performance. Figure 16.7 shows the results for \(N_{rec\_num} = 1000, \ N_{data\_dist} = 50\%, \ N_{attr} = 80, \ N_{ret\_tuple} = 30, \ N_{proj} = 4, \text{ and } N_{cond} = 3.\) \(N_{rec\_num}\) takes
the values of 200, 400, 600, 800, 1000, and 1200. It shows that the system runs
closer as the number of queries to be processed increases. This is because the
more the number of queries, the more the number of inferences. Also, as each
user query needs to be compared with previously issued queries for the subsume
relations, the more the number of queries, the longer it takes to determine all
possible subsume relations.

16.7 SUMMARY
In this paper, we describe our effort in developing a data level inference de­
tection system. We have identified six inference rules: split query, subsume,
unique characteristic, overlapping, complementary, and functional dependency
inference rules. We have also discussed the applications of the inference rules
on union queries. The existence of these inference rules shows that simply using
functional dependencies to detect inference is inadequate. We have developed
a prototype of the inference detection system using Perl on a Sun SPARC 20
workstation.

Although the data level inference detection approach is inevitably expensive,
there are cases where the uses of such approach is practical. As shown in our
experimental results, the system generally performs better with a larger size
of the database, and queries that return smaller number of tuples and project
smaller number of attributes. The system running time becomes high when
queries retrieve a large amount of data from the database, and there are large
amount of overlapping among query results. However, when a user issues such
type of queries, it is suspicious that the user is attempting to infer associations
among the data.

Acknowledgments
The research reported in this paper is supported by the National Security Agency
University Research Program under Contract DOD MDA904-96-1-0117, and by the
CIA Office of Research and Development under Contract 96F 154000-000.

References
IFIP WG11.3 Workshop on Database Security, pages 195–209, August
[2] LiWu Chang and Ira S. Moskowitz. Bayesian methods applied to the
database inference problem. In Proc. 12th IFIP WG11.3 Workshop on
analysis and detection system. IEEE Transactions on Knowledge and Data


Figure 16.2  Effect of the Number of Attributes in the Database.

Figure 16.3  Effect of the Number of Return Tuples of Queries.

Figure 16.4  Effect of the Number of Projected Attributes of Queries.

Figure 16.5  Effect of the Size of the Selection Criteria of Queries.

Figure 16.6  Effect of the Number of Tuples in the Database.

Figure 16.7  Effect of the Number of Queries Issued.
XI Panel
17 SECURITY AND PRIVACY ISSUES FOR THE WORLD WIDE WEB: PANEL DISCUSSION

Bhavani Thuraisingham, Sushil Jajodia, Pierangela Samarati*, John Dobson, and Martin Olivier

17.1 INTRODUCTION BY BHAVANI THURAISINGHAM

This is the second in a series of panels at the IFIP 11.3 Working Conference on Database and Application Security. While the first panel in 1997 focussed on data warehousing, data mining and security, the panel in 1998 focussed on web security with discussions on data warehousing and data mining.

A data warehouse integrates data from heterogeneous data sources possibly on the web into a single repository so that users can query to make decisions effectively. Data mining is the process of posing queries and extracting information previously unknown possibly form warehouses. The advent of the web together with data warehousing and mining tools is a serious threat to privacy and security of individuals.

The panel positions include discussions of data warehousing and data mining security aspects as well as legal and social aspects of web security. Appropriate privacy laws as well as policies are needed to protect the privacy of individuals. This is a major problem as web id international and different countries have

*Pierangela Samarati is on leave from Università di Milano. The work of Pierangela Samarati was supported in part by the National Science Foundation under grant ECS-94-22688 and by DARPA/Rome Laboratory under contract F30602-96-C-0337.
different privacy laws. These heterogeneous policies may have to be resolved possibly to have a uniform policy for the web. There was also interest with the IFIP group to keep in contact with the security group at the World Wide Web Consortium and hopefully to influence the developments with this consortium.

The panelists were: Sushil Jajodia from The MITRE Corporation, Pierangela Samarati from SRI, John Dobson from University of New Castle Upon Tyne and Martin Olivier from Rand Afrikaans University. The positions of the panelists are described below.

17.2 POSITION BY SUSHIL JAJOdia: ACCESS CONTROL IN DATA WAREHOUSES

Generally, the driving force behind the implementation of a data warehouse is the goal of providing a more complete picture of an organization’s operations to support management decisions. Although the security concerns for a data warehouse are the same as those for any other information systems (integrity, access control, authorization, privacy, and confidentiality), data warehouses present some unique and challenging issues.

Identifying and implementing an access control policy for a data warehouse involves a number of unique challenges. One is the dissonance between access control schemes for data models supported by operational DBMSs and those provided by data warehouse. For example, the relational model is the predominate data model in use today, while decision support systems tend to exploit analytical opportunities offered by non-traditional data models such as the star, temporal, snow flake, or multidimensional data models. The general lack of representation models for defining access controls further frustrates any process for deriving appropriate access controls at the data warehouse level from those used at the operational database level.

In practice, the specification of security policies at the DBMS level is very rudimentary, and organizations rarely document their information system security policies. Finally, users of the operational systems are not the same as the users of the data warehouse, so an access control policy used for an operational system may have little resemblance to one appropriate for the data warehouse level.

Below, we examine issues related to the access control in data warehouses. We have taken a pragmatic approach. By its very nature, a data warehouse creates a conflict between data availability and security [19]. On the one hand, the goal of every data warehouse is to make available to all concerned the information they need, and too much security may have the consequence that users do not have access to all the information that is necessary to do their job. On the other hand, an organization needs to ensure that this same valuable data is not exposed to unauthorized individuals or corrupted by hostile parties. Too little security may mean that users may access information through data warehouse that they cannot access to directly from the sources and, moreover, certain important data may not be made available to the data warehouse by the sources. Therefore, it is important that the correct balance between avail-
ability and security is maintained so that all users that could benefit from some information will have access to it.

Controlling access to a data warehouse is particularly important since the data warehouse encompasses data from many systems and contributes to decision-making across organizational boundaries. In fact, access controls to a data warehouse need to be considered at a number of levels [Ross96]:

- Who will have access to the processes that extract the operational data?
- Who has access to the extracted data and the processes that transform the extracted data into a format suitable for inclusion in the data warehouse?
- Who will access to the data in the data warehouse itself? The ease of access to large amounts of data raises concerns about attaching the appropriate level of security without inhibiting analysis.

In a data warehouse, there is a Warehouse Administrator who should be responsible for deciding which users can execute which processes to extract what operational objects. It should normally be the case that if a user has the privilege to extract data from some operational objects, then that user has the read (or select) privilege to the operational objects in the first place.

Who should control to the extracted data and to the processes that transform the extracted data into a format suitable for inclusion in the data warehouse? Once again, the warehouse administrator should be responsible for designating a small group of privileged users who have access to the extracted data and to the processes that transform the extracted data into a format suitable for inclusion in the data warehouse. The user who creates an object to be stored in the data warehouse becomes the owner of the object and is responsible for deciding which subjects are to have what privileges on the objects.

Who will access to the data in the data warehouse itself? In a relational on-line analytical processing (ROLAP), we see the creation of a star schema as being analogous to defining a view in a relational DBMS. The creator of the star schema becomes the owner of the star schema and can decide who is to have what types of accesses on his/her object. Just as different views can be defined for different users for security reason, different star schemas can be defined for security as well.

In a multidimensional on-line analytical processing (MOLAP), an entire cube is materialized, and the analysis tools are applied directly on the materialized cube. We view each materialized cube as an authorization object. The user creating the data cube must have the read privilege on the underlying detail data, and gets to decide who has subsequent access to the cube. No attempt is made to hide parts of a cube from the analyst.

17.3 POSITION BY PIERANGELA SAMARATI: THE PRIVACY PROBLEM AND THE WORLD WIDE WEB

The increased power and interconnectivity of computer systems available today provide the ability of storing and processing large amounts of data, resulting
in networked information accessible from anywhere at any time. It is becoming increasingly easier to collect, exchange, access, process, and link information. In this global picture, people lose control of what information others collect about them, how it is used, and how, and to whom it is disclosed. While before, when releasing some information (be it our health situation to a doctor or our credit card number to a restaurant waiter) we needed to trust a specific person or organization, we now have to worry about putting trust, or some control, over the entire network. It is therefore inevitable that we have an increasing degree of awareness with respect to privacy. Privacy issues have been the subject of public debates and discussions and many controversial proposals for the use of information have been debated openly. In the United States as well as in many European countries, privacy laws and regulations are being demanded, proposed and enforced, some still under study and the subject of debates.

A commonly accepted definition of privacy refers to privacy as the “right of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others.” As we try to spell out the privacy problem with respect to the World Wide Web we can distinguish different aspects, including the classical problem of protecting the confidentiality of information when transmitted over the network (for instance in electronic commerce when we communicate a credit card number over the web); the problem of protecting web surfers from “being observed” as they navigate through the network; the problem of controlling the use and dissemination of information collected or available through the web; and the problem of protecting against inference and linking (computer matching) attacks, which are becoming easier and easier because of the increased information availability and ease of access as well as the increased computational power provided by today’s technology. Although we recognize the importance of providing communication secrecy, we will not discuss this problem any further. We will focus instead on privacy issues concerning information gathering and dissemination.

Privacy issues in data collection and disclosure

Information about us is collected every day, as we join associations or groups, shop for groceries, or execute most of our common daily activities. It has been estimated that in the United States there are currently about five billion privately owned records that describe each citizen’s finances, interests, and demographics. Information bureaus such as TRW, Equifax, and Trans Union hold the largest and most detailed databases on American consumers. There are also the databases maintained by governmental and federal organizations, DMVs, HMOs, insurance companies, public offices, commercial organizations, and so on. Typical data contained in these databases may include names, Social Security numbers, birth dates, addresses, telephone numbers, family status, and employment and salary histories. These data often are distributed, or sold. This dissemination of information has been in some cases a matter of controversy (remember the open debates about the plan of America On Line to provide telephone numbers of its subscribers to “partner” telemarketing firms,
which resulted in AOL canceling the plan). In other cases, this dissemination of information is becoming common practice. In some states (Texas is an example) it is today possible to get access to both the driver’s license and license plate files for a $25 fee. Although one may claim that information these databases contain is officially public, the restricted access to it and expensive processing (in both time and resources) of it represented, in the past, a form of protection. This is less true today. Concerns are voiced by individuals who are annoyed by having their phone numbers and addresses distributed, resulting in the reception of junk mail and advertisement phone calls. Even more of a concern is that these data open up the possibility of linking attacks to infer sensitive information from data that are otherwise considered “sanitized” and are disclosed by other sources.

Even if only in statistical, aggregate, or anonymous form, released data too often open up privacy vulnerabilities through data mining techniques and computer matching (record linkage). Tabular and statistical data are vulnerable to inference attacks. By combining information available through different interrelated tabular data (e.g., Bureau of Census, Department of Commerce, Federal and Governmental organizations) and, possibly, publicly available data (e.g., voter registers) the data recipient may infer information on specific microdata that were not intended for disclosure. Anonymous data are microdata (i.e., data actually stored in the database and not an aggregation of them) where the identities of the individuals to whom the data refer have been removed, encrypted, or coded. Identity information removed or encoded to produce anonymous data includes names, telephone numbers, and Social Security numbers. Although apparently anonymous, however, the de-identified data may contain other identifying information that uniquely or almost uniquely distinguishes the individual. Examples of such identifying information, also called key variables, or quasi-identifiers, may be age, sex, and geographical location. By linking quasi-identifiers to publicly available databases associating them to the individual’s identity, the data recipients can determine to which individual each piece of released data belongs, or restrict their uncertainty to a specific subset of individuals [25].

The large amount of information easily accessible today and the increased computational power available to the attackers make inference and linking attacks one of the serious problems that should be addressed. The threat represented by inference and linking attacks is also of great concern because of the fact that statistical, aggregate, and anonymous data are often exempted from privacy protection regulations. More than others, these data may therefore open up the possibility of potential misuses.

2For simplicity, we refer to the entity to whom information refers as an individual. This entity could, however, be an organization, association, business establishment, and so on.
Anonymity issues when surfing the web

While in some cases we know that data about us are collected, but we may not have any control about their use and dissemination; in other cases we may not even be informed that data about us are being collected and distributed. Many people surf the web under the illusion that their actions are private and anonymous. On the contrary, every move they make throughout the net and every access they request are observed and possibly recorded. It is common practice for web servers to maintain a log file recording requests to URLs stored at the server. Each time we hit a web page, the web server records the following information: name and IP address of the computer that made the connection, username (if HTTP authentication was used), date and time of the request, name (URL) and size of the file that was requested and time employed for downloading it, status code or any errors that may have occurred, web browser used, and the previous web page that was downloaded by the web browser (refer link). The refer link tells the server the page at which we were looking prior to making the request (i.e., the page “we came from”). One of the reasons for justifying the passing and recording of such information is to allow servers to chart how customers move through a site, and to check the effectiveness of advertisements (as advertisers can control “from where” visitors to their pages arrive). The refer information itself can be seen as a violation of the surfer’s privacy, and some more serious concerns arise from information that can be inappropriately leaked through it. Web search engines, such as Lycos, encode the user’s search query inside the URL. This information is sent along and stored in the refer link. This means that the server not only knows where we came from, but also what we were looking for. More of a concern is the fact that the URLs fetched from one site using cryptographic protocols (e.g., SSL) may be sent to the next site contacted over an unencrypted link. Thus, for instance, our credit card number that we thought protected because it was communicated over an encrypted link may be communicated unencrypted to other sites. Another threat to surfers’ privacy is represented by cookies. A cookie is a block of ASCII text that a web server can pass into a user’s instance of a browser and that is then sent to the server (and back again to the browser) along with any subsequent request by the user. Cookies, while providing advantages such as the user’s customization, also allow the server to track down a user through multiple access requests to the server and possibly (if cookies are passed among servers) through the entire network. In this sense, cookies represent threats to surfers’ privacy.

Data recording information about users’ surfing activities over the network are called navigational or transactional data. Privacy regulations (such as the Electronic Communication Privacy Act) do not generally restrict the use of transactional data; they protect only its content but not its existence. This implies that a service provider can disclose transaction information without the subscriber’s consent.

Users concerned with privacy and wishing to anonymously surf the network can today do so by using anonymizing servers. Anonymizing servers act as
proxies for the user. Instead of connecting directly to the server they wish to access, users connect to the anonymizing server and pass it the desired URL. The anonymizing server removes a user's identifying information from the request and submits it. The reply also passes to the user through the anonymizing server. In this way the web server of the URL to be accessed receives the request as coming from the anonymizing server. It is worth noticing that in this case the anonymizing server has the ability to observe and record the user's requests. Users need therefore to trust the anonymizer to provide the desired anonymity.

In June 1997, the Electronic Privacy Information Center reviewed 100 of the most frequently visited web sites. The purpose of the review was to examine the collection of personal information and the application of privacy policies by web sites. In December 1997, Bill Helling performed the same survey on the same sites to see whether the situation had changed. Some interesting numbers were reported by EPIC [10] and by Helling [14] as a result of these reviews (numbers reported by the later survey appear in parentheses):

- 49 (57) sites collected personal information (such as name, address, e-mail address) through online registrations, mailing lists, surveys, user profiles, and so forth. The review could not determine whether the collected information was used for linking data with other databases. Such linking has been found to be performed in some cases (for instance, by America On Line).

- Only 17 (29) sites had explicit privacy policies. Among those, some had policies considered inadequate, some reasonably good. EPIC reports that only a few were easy to find and, although some were considered reasonably good, none of them was considered to meet the basic standards for privacy protection. Helling notes that the sites that later added a privacy policy seemed to make this policy easier for users to locate.

- Only 8 sites provided some ability to the users to limit secondary use of their personal information. This ability is limited to the possibility of specifying whether the collecting organization will be authorized to share (or sell) the information to a third party.

- No site allowed users to review information collected about them. As an exception the Firefly site allowed users to create, access, and update their own personal profile.

- 24 (30) sites enabled cookies. According to [14], 16 of the 30 sites collecting cookies passed the cookie on the home page, before the user could read or link to any explanation. Moreover, at least 7 of the cookies passed on the home page were third-party cookies.
Specifying privacy constraints

Privacy laws and regulations are currently being enforced, and new laws are still under study. They establish privacy policies to be followed that regulate the use and dissemination of private information. A basic requirement of a privacy policy is to establish both the responsibilities of the data holder with respect to data use and dissemination, and the rights of the individual to whom the information refers. In particular, individuals should be able to control further disclosure, view data collected about them and, possibly, make or require corrections. These last two aspects concerning the integrity of the individual's data are very often ignored in practice (as visible from the results of the EPIC survey).

The application of a privacy policy requires corresponding technology to express and enforce the required protection constraints, possibly in the form of rules that establish how, to/by whom, and under which conditions private information can be used or disclosed. With respect to the specification of use and release permissions, authorization models available today prove inadequate with respect to privacy protection and, in particular, to dissemination control or protection by inference. Features that should be provided in an authorization model addressing privacy issues should include

- **Explicit permission.** Private and sensitive data should be protected by default and released only by explicit consent of the information owner (or a party explicitly delegated by the owner to grant release permission).

- **Purpose specific permission.** The permission to release data should relate to the purpose for which data are being used or distributed. The model should prevent information collected for one purpose from being used for other purposes.

- **Dissemination control.** The information owner should be able to control further dissemination and use of the information.

- **Conditional permission.** Access and disclosure permissions should be limited to specific times and conditions.

- **Fine granularity.** The model should allow for permissions referred to fine-grained data. Today's permission forms for authorizing the release of private information are often of a whole/nothing kind, whereby the individual, with a single signature, grants the data holder permission to use or distribute all referred data maintained by the data holder.

- **Linking and inference protection requirements.** The model should allow the specification and enforcement of privacy requirements with respect to inference and linking attacks. Absolute protection from these attacks is often impossible, if not at the price of not disclosing any information at all. For instance, given some released anonymous microdata, the recipients will most certainly always be able, if not to determine exactly the
individual to whom some data refer, to reduce their uncertainty about it. Privacy requirements control what can be tolerated, for instance, with respect to the size of the set to whom this uncertainty can be reduced [25].

It is worth noticing that simple concepts, traditionally applied in authorization models, become more complicated in the framework of privacy. An example is the concept of information owner. The answer to this question is not easy and perhaps belongs more properly to the public policy domain. For instance, there have been open debates concerning whether a patient or the hospital owns the information in the patient’s medical records maintained by the hospital. Perhaps the notion of owner as traditionally thought does not fit in such context and instead should be revised or substituted by one or more other concepts expressing the different parties involved (data holder vs. individual). A good privacy model should allow the expression of these different parties and of their responsibilities. To the public policy domain will then belong the answer as to how to express such responsibilities (for instance, whether the specification of privacy constraints must remain with the data holder, the individual, or both).

Conclusions

The protection of privacy in today’s global infrastructure requires the combined application solution from technology (technical measures), legislation (law and public policy), and organizational and individual policies and practices. Ethics also will play a major role in this context. The privacy problem therefore covers different and various fields and issues on which much is to be said. These notes are far from being complete in that respect. As society discusses privacy concerns and needs, it is clear that research is needed to develop appropriate technology to allow enforcement of the protection requirements.

While stressing the importance of protecting privacy, it is also fair to mention that there are trade-offs to be considered. With respect to anonymity of web surfers, for example, complete and absolute privacy conflicts with the basic requirement of accountability, which demands that users be accountable for actions they execute. Just as we would like not to be consistently observed and recorded while we navigate through the network, it is also true that we would like to be able to determine who accessed our site if, for instance, some violations are being suspected. With respect to data dissemination control and protection from inference and linking attacks, cases may exist where privacy can be (partly) sacrificed in favor of data availability. Let us think for example about data disclosed for scientific research purposes, or about the desire of having globally accessible medical databases so that an individual’s medical history be available immediately in case of an emergency, wherever or whenever this might occur. A satisfactory response to these trade-offs may come from the development of new and better technologies. For instance, the application of new measures to protect against inference and linking attacks can allow the
satisfaction of data privacy requirements while at the same time maximizing data sharing and availability. Much research needs to be done in this field.

17.4 POSITION BY JOHN DOBSON: WHY IS INFORMATION PRIVACY SO HARD?

The best definition of privacy that I know is one due to Joachim Biskup that defines it in terms of role separation: an individual in society takes on a number of roles, and privacy is the right to expect society to respect the individual’s chosen separation between these roles.

One problem is that social roles are constantly renegotiated in conversations and changing relationships, or are subverted by legislation; therefore any view of privacy that assumes roles are fixed or immutable is likely to be inadequate. Unfortunately, many computer systems take this view.

A second reason is that the location of the public/private boundary is culturally determined, and therefore a system developed in one culture with one set of assumptions about the location of that boundary is likely to prove unfit for purpose in another culture with another set of assumptions.

A third reason is that we don’t know yet how to transfer our understanding of normal social relationships to computer-mediated social relationships. Every time you engage in a social relationship you take a privacy risk. In circumstances we understand, we can evaluate this risk (at least subjectively) in making the decision how much to commit to the relationship. In computer-mediated relationships we don’t yet have enough experience to know how to do this. Furthermore, part of this risk evaluation depends on the existence in the social world of recourse: we can employ the courts and other social sanctions and actions if we feel we have been betrayed. Again, in computer-mediated relationships we don’t know how to do this.

More generally, there is a distinction between security and privacy which throws an interesting light on the issue, and it is this: in security, role definitions have been institutionalised whereas in privacy, role definitions have not been (and probably cannot be) institutionalised. By this I mean that our understanding of security depends on our knowing who is supposed to know or have access to what, this knowledge being based on public knowledge of assignment of role; whereas our understanding of privacy depends on the individual’s choice of which roles to assume, this being an individual matter and not necessarily open to public knowledge. The importance of the kind of institutionalisation required for security is that mechanisms to support it can be made part of the social or technical infrastructure underlying the institution, whereas this is just not possible for something like privacy that almost by its very definition has to remain structural. In fact any information system has the effect of forcing the institutionalisation of role, and that is why they can be so dangerous to privacy.

A related way of putting this is that in terms of a formal logic, infrastructural definitions (e.g. of security) can be expressed in extension and can therefore be the subject of mechanical interpretation whereas structural definitions (e.g.
for privacy) can only be expressed intensionally, and must therefore remain in the domain of policy and legislation.

17.5 POSITION BY MARTIN OLIVIER: PERSONAL PRIVACY

A human being's personal privacy refers to his/her ability to limit collection and use of his/her personal information. Where such limitations cannot be determined by the individual, suitable mechanisms have to exist to ensure that collection and use will be adequately limited. While a significant amount of work has been done about what constitutes adequate limitations (see below), hardly any work has been done on suitable (technical) mechanisms to provide any guarantees about such limitations.

It is the contention of this author that such mechanisms are not only required, but that it is also technically feasible to develop them.

**General privacy principles**

For computing purposes personal privacy may be split into communications privacy and database privacy. **Communications privacy** has to ensure the privacy of personal information during communication. Privacy mechanisms may enable the individual to control what personal information is communicated, what information is collected by the party it communicates with and for what purpose such information is collected. In addition to these controls, communication privacy requires communication security mechanisms to ensure integrity of communicated information and confidentiality of such information against third parties.

**Database privacy** has to ensure proper use of personal information once it has been collected. Fundamental to such proper use is use of the collected information only for the purpose for which it has been collected. Three principles govern proper use of personal information: The **need to know privacy principle** is similar to the need to know security principle, but restricts a subject to access an individual’s personal information to *when* the subject needs the access to *that* individual’s information. The **acceptable use** privacy principle mandates that no-one should be able to use information for purposes other than for what it had been collected. In particular does this principle prohibit comparison or aggregation of or derivation of information from personal data collected for incompatible purposes. **Integrity of personal information** has to ensure that information is correct, timely and up to date.

Privacy protection is both a personal and fundamental right of all individuals. Individuals have a right to expect that organisations will keep personal information confidential. One way to ensure this is to require that organisations will collect, maintain, use, and disseminate identifiable personal information and data only as necessary to carry out their functions.
Privacy in practice

The principles given above form the basis of ethics viewpoints on privacy [22, 2, 6, 18, 20, 26]. Laws are also based on these principles. The US Privacy Act of 1974, for example, limits government (federal) use of personal data to relevant and necessary data to accomplish the purpose of the concerned federal agency [22].

Processing of personal information is allowed according to the European Data Protection Directive if the concerned subject has unambiguously consented to the processing for which notification has been given as to the purpose for which the information is sought [20]. Otherwise "the processing of personal data must ... be necessary with a view to the conclusion or performance to a contract binding on the data subject, or be required by law, by the performance of a task in the public interest or in the exercise of official authority, or by the interest of a natural or legal person provided that the interests or the rights and freedoms of the data subject are not overriding" [8]. The directive requires "appropriate safeguards" to ensure that personal information is not compromised, but does not specify the nature of such safeguards.

Correctness, and legitimate use of personal information is addressed by the right of any individual of "... access to data relating to him which are being processed, in order to verify the accuracy of the data and the lawfulness of the processing" [8].

Note that, once information has been collected for "specified, explicit and legitimate purposes" it may not be " further processed in a way incompatible with those purposes" [8, Article 6.1(b)].

As a national example, the Dutch Data Registration Act [27] also limits use of personal information to the purpose for which it has been collected (Article 6), information has to be obtained lawfully (Article 5) and individuals normally should be able to access their personal information (Article 29). Not only are individuals able to request the contents of such records, but also the origin. The law also requires the technical and organisational protection of such information against unauthorised access or modification. The law does, however, primarily expect the individual to monitor application of the law [26].

Some laws even go so far as to prohibit the collection of some personal information. The European Data Privacy Directive requires that "Member States shall prohibit the processing of personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, trade-union membership, and the processing of data concerning health or sex life" [8, Article 8.1] except in a small number of enumerated cases.

Laws also realise the power of computers — in particular the power to derive information that would not have been possible without their ability to aggregate and compare large amounts of data. The US Computer Matching and Privacy Protection Act of 1988 states that "agencies must follow specific procedures when engaging in the automated comparison of Privacy Act databases on the basis of certain data elements" [22].
Automated privacy mechanisms

Very little has been done to use technology to enhance privacy. Almost all such attempts focus on the individual's ability to limit the collection of private information. It has been argued that technology should be used to allow the individual to specify personal preferences as to what information may be collected [15, 5]. How a specific individual's personal information will be handled, will be negotiated before any interaction between an individual and an organisation occurs. If such contact is on-line (such on the World Wide Web), negotiation may even be automated. How can an organisation be trusted that it will honour its privacy undertakings? Here it has been suggested that organisations may be 'certified' by some mutually trusted body [9, 3]. Various possibilities then exist if the organisation does not adhere to its undertaking.

Technical work has also been done that enables an individual to withhold information from an organisation by interacting anonymously with the organisation. This ranges from 'anonymysers' on the World Wide Web, anonymous remailers on the Internet, electronic forms of cash and other secure payment protocols.

There are, however, many instances where collection of private information cannot be avoided. And, once personal information has been collected, it needs to be properly protected.

Traditional security mechanisms form the first defence against privacy violations. But traditional security is not enough to ensure privacy. The majority of the privacy violations mentioned in [21, 17] were committed by authorised users of the system. Anderson [2] rightly points out that the privacy problem is compounded if the value of information increases or the number of people that have access to it increases — the former because the incentives to misuse information are that much higher; the latter because the potential to find an unscrupulous individual is simply higher in a larger group of (often remote) users. Aggregating personal information in a centralised database increases both the value of the information and, usually, the number of people who have access to it. Someone who is allowed access to some individual's personal information should not necessarily be allowed access to the same information of other individuals.

Solutions

Data privacy needs mechanisms that will limit use of personal information according to some privacy policy. Many privacy policies already exist: a quick survey of many corporate (and other) Web-sites will show that many of them have a privacy policy in place. These policies often assume that individual employees of such organisations are trustworthy and will adhere to the accepted policy. The examples cited in [21, 17] show that this is not, in general, true for all employees. To address this problem, policies aimed at limiting the damage an authorised individual may do need to be considered. Note that perfect policies do not exist: consider the doctor who obtains personal information
shares this information with an individual not concerned with the case. This violates privacy, but neither the collection of the personal information, nor the sharing of the information can be controlled by technical means. We therefore need policies that technically limit the possibility of violations and then use professional, ethical and legal means to address the remaining problem.

The starting point is obviously traditional security mechanisms: by limiting access to individuals or roles who need to know the information, already avoids many potential violations.

Various possibilities limiting users’ abilities to browse personal information also exist: a tax inspector needs access to personal tax information; however, no need exists for such an inspector to be able to access all individuals’ tax records. This seems to require partitioning mechanisms based on users: with any particular user (or subject) only having access to a subset of individuals’ information, the potential to violate privacy is again reduced. Determining suitable subsets depends on the application: A doctor may, for example, only access medical information of patients treated by him/her. The subset of taxpayers whose information a tax inspector may see, may conceptually be determined randomly.

Possible ‘technical’ privacy policies to limit privacy violations further, may focus on the partitioning of information in databases based on the purpose for which such information has been collected. Limiting an individual’s access to a single partition (or a few partitions) may be the first step to limiting users’ abilities to acquire unneeded information. In particular does limiting computerised correlation and aggregation of information across partitions have potential.

Once such technical policies have been devised, it becomes possible to design mechanisms to support these policies.

Ensuring privacy of information usually does not hold immediate benefits for the concerned organisation, as secrecy of organisational confidential information does. The reasons why organisations may consider implementing privacy controls are twofold: Firstly, they may be forced by law to ensure privacy. Secondly, having such controls in place may be indirectly beneficial to the organisation, in the same way that steps taken to prevent pollution may be. In both cases it is necessary to audit the controls to ensure that they are adequate and effective.

References


XII Discussion Summary
The twelfth IFIP WG 11.3 Working Conference on Database Security was held at Porto Carras Complex, Chalkidiki, Greece, on July 15-17, 1998. Sushil Ja­jodia, the Program Chair, gave the opening remarks. The conference consisted of fourteen papers, two invited talks and a panel discussion.

The first session began with an invited talk "E-Commerce Security: No Silver Bullet," by Anup Ghosh, Reliable Software Technologies. Anup discussed several security issues in E-Commerce including security requirements in the E­Commerce environment, client and server vulnerabilities, security concerns of both merchants and consumers, various secure payment systems and protocols, and E-Commerce security today and tomorrow.

When Anup mentioned that most E-Commerce security protocols are based on encryption and are susceptible to a variety of common attacks, Sushil Ja­jodia questioned how realistic attacks such as man-in-the-middle are. Anup responded that the public might loose confidence if there is a single attack. When Anup mentioned that the system is only as secure as its weakest compo­nents, Sushil did not agree. Sushil suggested that there is a need for something like a reference monitor where one can have several untrusted components but the whole system can still be secure.

T.C. Ting questioned how secure is secure so that customers feel comfortable with electronic commerce just the same as customers trust a restaurant when giving a credit card. T.C. asked whether there is any notion of the degree of security that is acceptable to the public. Anup replied that there will be problems until businesses become aware of the problem and do something about it. Sushil commented that E-Commerce progress is slow because there is no customer confidence, and he suggested that there is a need for standards and
risk analysis of protocols. Anup replied that his company used risk analysis on its full system.

For Sushil's question on what certification of a software product means, Anup responded that a 3rd party certifier analyzes software for certain types of defects. T.C. asked about the technical basis for certifiers. Anup said that merchants have most of the risk and the most to lose. John Dobson commented that pipeline component testing will identify interesting research problems underneath.

For Thomas Keefe's questions on Trojan Horse detection and black box techniques, Anup responded that they look at the behavior of the product. For Sujeet Shenoi's question on how successful Trojan Horse detection is, Anup said that they have developed methodologies to detect Trojan Horses but they do not have any empirical results. For T.C.'s question of who are the potential customers, Anup replied that they are VISA and banks. For Sujeet's comment on the advertisement of secure E-Commerce servers by IBM, Anup commented that we trust them because they are from IBM.

During session 2, Anup continued his discussion and presented techniques to detect software vulnerabilities. For John's question on whether software flaws rather than configuration problems cause security violations, Anup responded that his company is now addressing configuration more from the software development point of view. Anup mentioned that current approaches first develop the software and then release patches to that software, which is not good. Sushil supported Anup by adding that this approach does not work because it may mean that the system has to be down for some time. This can be avoided by Anup's system since it analyzes software behavior and this enables software vendors to detect security flaws before they release the software. For T.C.'s question on what security value can be given to a system, Anup said that testing provides some level of confidence.

For John's question on whether there is any use of formal methods to decide if a flaw is a security flaw, Anup responded by saying that a flaw gives more privileges than it should. One does not need a well-defined security policy to observe this. There does not exist a formal method that is capable of doing this.

For Ehud Gudes' concern over the exponential number of states in a software system, Anup responded that they inject one fault at a time to identify the statement that causes a security violation. For Sushil's question on the size of code analyzed, Anup said that it is typically in the range of 10 - 20 thousand lines and it typically takes one person two weeks to develop a degree of confidence in the certification of a software component.

John questioned that since an analyst must write assertions about a security policy, how does the analyst know that all the required assertions are included, especially for non-intuitive things about causal relationships? There may be a mismatch in the level of expression of the security policy and a flaw. Anup replied that they map to a program, not to a flaw.
These two sessions on Security in Electronic Commerce generated interesting discussions among the conference participants. It was decided that the scope of future WG 11.3 conferences should be broadened to include such topics as security for E-Commerce, the WWW and Digital Libraries.

The third session was an invited talk by Joachim Biskup on "Technical Enforcement of Informational Assurances," and was chaired by David Spooner. Sushil asked the reason for using the word 'informational' rather than 'information'. Joachim said that it is a generalization of the word based on the German language. For John's question on whether a person can have multiple digital signatures under the German law, Joachim said yes. John was concerned about the presupposition of conflict resolution such that the intended purposes of a system are considered legitimate and consistent. When Joachim answered that one must relate it to prior law, John commented that this is very fluid as the law changes over time. Joachim said that periodic review of the security of a system is needed.

The last session on the first day of the conference was a panel discussion on "Privacy Issues in the WWW, Data Warehousing and Data Mining." Bhavani Thuraisingham chaired this session. The panelists, John Dobson, Sushil Jajodia, Martin Olivier and Pierangela Samarati, discussed issues such as anonymity versus accountability, data availability and access versus privacy, access control models for data warehouses, and conflicting and differing privacy policies in different countries. Ehud asked whether it is possible to have anonymous access to the web. Pierangela answered that anonymizers are available. Sujeet said Mike Reiter's work proves how anonymous anonymous is. Sushil asked whether we should trust the anonymizer. John responded that a decision has to be made about the risks involved and what information one is willing to share and not willing to share with others. T.C. asked what the borderline is between privacy and availability. Bhavani responded that the issues are more social than technical. Pierangela responded that availability is essential and more important than privacy in environments such as medical organizations.

For Anup's question on whether there are any technical solutions to protect user information collected by a web site, Sushil responded that clean polices and laws are needed and privacy issues need to be incorporated into access control. John responded by saying that technical solutions are not sufficient; the only solution is a method of recourse backed by law.

Bhavani concluded the panel by saying that the WWW Consortium is looking at security issues and asked whether anyone attending the conference knows what is going on there and how we can contribute. John responded that the current work of the consortium in the security area is flawed technology with no problem statement, and there is a need for a definition of security. The consensus was that the group should follow up with another panel session on WWW security at next year's conference in Seattle.

The second day of the conference started with a session on "Workflows," chaired by Tom Keefe. The first paper was "Analyzing the Safety of Workflow Authorization Models," by Huang, W.-K., and Atluri, V. This paper was
presented by Vijay Atluri. Vijay explained how the Workflow Authorization Model (WAM) can be extended to specify separation of duties constraints and showed how a safety analysis of WAM can be carried out using a Petri net based approach. Ehud asked how distributed authorizations can be checked. Vijay responded by saying that they are now using a centralized database and therefore everything can be determined from it. Tom added that a temporal part is not essential for enforcing separation of duties. Vijay said that they use a non-temporal projection of the authorization. The second paper in this session titled, "Rules and Patterns for Security in Workflow Systems," by Castano, S., and Fugini, M.G., was presented by Silvana Castano. She discussed how ECA rules are used to enforce authorization constraints in the Workflow Interactive Development Environment (WIDE) workflow management system.

T.C. Ting chaired the next session on "Policy Modeling." The first paper, "Security, Queries, and Autonomous Object Databases," by Gudes, E., and Olivier, M.S., was presented by Ehud Gudes. Ehud presented security policies to handle conflicts between local and global authorizations due to local autonomy. Sushil asked when data objects are replicated, how consistency of all the copies is ensured. Ehud said that they have not addressed this issue yet, but a version-based system may be appropriate. Karabulut asked how migration of object instances can be handled. Ehud said that global rules must be considered for moves. In addition, new local rules must be created. The original administrator loses control of the object instances.

The second paper titled, "Programmable Security for Object-Oriented Systems," by Hale, J., Papa, M., and Shenoi, S., was presented by Sujeet Shenoi. Sujeet showed how a primitive distributed object model can be used to support various types of decentralized authorization models by programming access control at the language level. Bhavani said that OMG has a standard object meta model and questioned the need for a new model. Sujeet said that it is part of a larger project that started before the OMG standard and that they can translate other meta models such as the OMG standard into their model. Ehud asked whether their model can handle a hierarchy of roles. Sujeet said that they have not looked at this yet, but are thinking that they should be able to do it. Bhavani suggested that they should look at the OMG standard and incorporate that in this work. Sujeet pointed out that it could be a problem because the OMG standard is a moving target. Tom asked what a token in the access control model is. Sujeet said that a token and a privilege together are needed to access an object and both are unforgeable. Karabulut asked about the relationship of their model to CORBA. Sujeet said that CORBA can be easily mapped into their model but they have not looked at this yet.

Sushil chaired the next session on "Mediation and Information Warfare Defense." The first paper, "Secure Mediation: Requirements and Design," by Biskup, J., Flegel, U., and Karabulut, Y. was presented by both Flegel and Karabulut. They first distinguished mediation from federation and presented approaches to secure interoperation. Bhavani asked if one needs to do mediation to do a federation. Flegel agreed that this is true but said that there are
differences between the two concepts that would become apparent. John asked whether they have considered the issue of requiring prior access to an information provider in order to establish a relationship with that provider, which also requires mediation. Karabulut said that they can extend their approach to handle this. John suggested that it would be useful to extend it into other types of relationships as well (anonymous, commercial, etc.).

The second paper, "Reconstructing the Database After Electronic Attacks," by Panda, B., and Giordano, J., was presented by Brijendra Panda. He presented algorithms to assess the damage after an attack and then to recover from the damage. Bhavani asked how the transaction that is causing the problem can be identified. Brijendra said that they assume an intrusion detection system that is capable of determining this. John asked how they can justify undoing operations based on false information resulting from previous intrusions. Brijendra said that they cannot handle situations where bad data is remembered outside the system and used in later transactions. Tom asked whether they can model data that leaves the system and notify users when a problem is discover with it. Brijendra responded that he believes this can be done, but added that it might require maintaining a lot of extra data. Sushil said that external effects can be fixed outside the system in some other way. Sushil commented that this can be called data corruption detection, which is similar to storage jamming that was described at a previous WG 11.3 conference. Storage jamming also tries to detect corruption. Sushil added that a new research direction would be to look at what to do to recover after the detection. Pierangela asked whether they keep track of the conditions that lead to the updates. Redoing old operations from the log is not sufficient since the conditions may change and different operations should be done. Sushil added that the log doesn’t make sense any more. Brijendra said that the entire transaction must be re-executed in the proper sequence. Ehud commented that it is not always possible to re-execute, e.g., ATM machine. Sushil said that there is a need to look deeper at the transaction before deciding what to do.

The last session on the second day was on "Multilevel Security," and was chaired by Bhavani. The first paper titled, "Version Management in the STAR MLS Database System," by Sripada, R., and Keefe, T., was presented by Tom Keefe. There were few questions regarding this paper.

The second paper, "SACADOS: a Support Tool to Manage Multilevel Documents," by Carrere, J., Cuppens, F., and Saurel, C., was presented by Jerome Carrere. Sushil asked when a new rule is applied, whether all the rules are recursively applied, and Jerome answered yes. Then Sushil questioned how conflicting rules are handled. Jerome replied that they use a resolution strategy that orders rules by priority. Sushil commented that all exceptions then need to be planned in advance to get the priorities correct. Joachim asked whether the previous work of their group that focused on complete and consistent assignment of labels is related to the present work. Jerome said no.

At the closing of this session, Bhavani asked whether anyone has comments on MLS. Sushil responded that we need to wait and see what happens with
existing commercial MLS products. He said that a new approach uses COTS and firewalls, but there are still interesting problems in MLS from a research point of view. A few people in the group said that they are still interested in working in this area. Bhavani said that no OODBMS vendor has picked up MLS ideas in a commercial product and noted that MLS aggregation and inference work transfers to data mining. Sushil said that only MLS systems are built with assurance in mind. Bhavani asked whether there are any agencies funding MLS research. Jerome said that in France there is support for MLS research and Sushil added that funding for MLS appears to be declining, but there is still some interest.

The first session on the third day was on "Role-based Access Controls and Mobile Databases" and was chaired by Silvana Castano. The first paper, "Using Role-Templates for Handling Recurring Role Structures" by Essmayr, W., Kapsammer, E., Wagner, R.R., and Tjoa, A.M. was presented by Wolfgang Essmayr. Bhavani asked whether the notion of 'agent' is similar to the notion of 'mediator'. Wolfgang said yes. Joachim commented that reuse of concepts is well known in programming languages and asked how this work is different. Wolfgang said that they mapped the concepts in programming languages to the concept of roles. Tom asked whether they need to modify applications to take advantage of role templates. Wolfgang said that they are not that far yet. Ehud asked whether they can specify different instances of roles to have different privileges. Wolfgang answered yes. Silvana asked how one can identify what templates are needed for roles. Wolfgang said that they can provide tools to do this analysis.

The second paper titled, "Security Capabilities and Potentials of Java" by Smarkusky, D.L., Demurjian, S.A.Sr., Bastarrica, M.C., and Ting, T.C., was presented by T.C. Ting. Ehud asked given roles, why are access control lists used. T.C. said that they provide additional functionality outside of roles and there is an access control list for each role that controls access to methods. Joachim asked whether translating roles to a class hierarchy using scope and visibility was enough to guarantee security. T.C. said that one can use public interfaces as in most OO systems and scope and visibility is one way to do it, but it may not be secure enough for all cases. Bhavani suggested that they need to work closely with the Java Security Group by making suggestions to them and learning from their work.

The third paper, "Security Issues in Mobile Database Access," by Lubinski, A. was presented by Astrid Lubinski. John asked what assumptions are made about the underlying network structure? Astrid said that she assumes a fixed network between base stations where only the first and last links can be mobile. Karabulut asked what minimal security means. Astrid said that this means the abstraction of the security rules to two or three main rules that must be satisfied and that often this takes the form of encryption. George Pangalos asked how the mobile component affects the security rules. Astrid said that it depends on the situation and there is no one answer. Bhavani asked whether there is any other work in this area. John responded that there has been a lot of classified
work in this area and the idea in this work is to separate the medium into a large number of segments so that one can have counter measures using different channels and introducing noise. T.C. asked whether packet switching can be a problem and added that time switching has more opportunities. Sushil added that this is the start of new work and more has to be done in this area.

The final session of the conference was on "Inference and Privacy," and was chaired by Ehud. The first paper, "Bayesian Methods Applied to the Database Inference Problem," by Chang, L., and Moskowitz, I. S., was presented by LiWu Chang. John asked whether they have any methods for estimating the confidence level. On the same note, Ehud asked what can be done as a defense for this approach. LiWu said that one might try spurious data and classifying more data. However, one does not want to undermine the usefulness of the database.

The second paper titled, "Inference Detection - A Data Level Approach," by Yip, R., and Levitt, K., was presented by Raymond Yip. Tom asked whether they used simulation or performance measurements in their study. Raymond said that they generated a database and queries and then measured the performance. Tom noted that one must consider the history, which gets larger and larger. Vijay added that changes to the database may invalidate the history. Raymond said that they consider changes as new information and can also search the history to identify problems. Sujit asked how they handle collusion. Raymond said that they combine the queries of users together. Sushil noted that one can perform the analysis in data dependent or data independent mode depending on the goals. Raymond said that there is not much overlap between queries in general, so the complexity is manageable.

The last paper, "An Information-Flow Model for Privacy (InfoPriv)," by Dreyer, L.C.J., and Olivier, M.S., was presented by L. C. J. Dryer. John asked how the assumptions about the environment are made. Tom said that one can trust certain users not to do certain things. John said that the only assumption one can make is that one cannot trust anyone, and added that serious breaches in the past have come from people who were trusted. He warned that the assumptions are too optimistic. Martin replied that their approach cannot catch all cases but will catch many. Sushil said that this is an important problem that we need to work on. John asked whether there is an assumption that people do not lie. Dreyer said that they assume that values are copied from entity to entity. But John said that in reality, people lie. Joachim noted that this is a very important problem, however, reducing privacy to information flow is like using agents to represent people and such systems have proven to be difficult to analyze. Ehud added that putting the user in the system allows one to include the user in the analysis. However, today firewalls control the flow between sites. Joachim said that firewalls do not consider the semantics of the information.
Index

Access control, 191–193, 195–196, 253
ActiveX controls, 7–11
ACTORS
   Loops, 120
Anonymity, 34, 128, 131
Argos, 121
Assigned methods, 211–212, 216
Asymmetric fingerprinting, 33
Attachments, 8
Authenticity, 128, 131
Authorization constraints, 64–65
Authorization patterns, 59–60, 68, 72–73
Authorization, 45, 47, 50
Authorization, 129–130
Bayesian, 237–250
Certificate, 32
CGI scripts, 14
Classification, 173–183, 185–187
Clustering index, 162
Communicating personal data agents, 30
Computer security, 3, 6
Confidentiality, 128, 131, 136
Conflict, 22
Cookies, 7
Cookies, 8
Cooperation under threats, 32
Credential, 30, 128, 131, 135
Cryptographic protocol, 32
Damage Assessment, 146–148, 154
Data Dependency, 146
Data-driven attacks, 13
Decision support tool, 173–174, 181, 184
Digital certificates, 7
Digital pseudonym, 34
Digital Signature Act, 26
Digital Signature Ordinance, 26
Digital signature, 26
Discretionary access control, 114
Discretionary access control, 210
Distributed Computing Environment, 121
Distributed object computing, 205–206, 210, 217
Distributed Trusted Operating System, 165
DoFlow, 88
Downgrading, 174–187
Dynamic aspects, 77, 85–89
E-commerce security, 5
E-commerce, 4
Electronic commerce, 3
European Union, 27
Evaluation criteria, 19
Executable content, 6, 8, 10
Fail-stop signature scheme, 33
Federated database system, 31
Federated database, 94, 127
Federated system structure, 30
Fingerprint, 137
Firewall, 12
Firewalls, 5
Firewalls, 13
Germany, 25
Health care, 24,
High assurance security
   Programmable security
      Object-oriented systems, 109
In-memory version directory, 159, 163
Inference, 237–239, 249
InfoPriv, 77–81, 83–90
Information can-flow graph, 78, 81
Information sanitiser, 77, 83–84, 88
Information society, 21
Information Warfare, 144
Informational assurances, 21
Informational self-determination, 18
Interest, 22
Internet, 4
Interoperability, 117
Interpreted languages, 15
Java applets, 7–8
Java security manager, 209
Java security, 9
Java, 120
JavaScript, 7–8
JavaScipts, 8
Javascripts, 8
Local autonomy, 94–95, 103
Local policies, 94, 98–101, 103
Local security autonomy, 30
Local view, 89
Mandatory access control, 115
Materialization, 137
MD5, 15
Mediated information system, 31
Mediation, 128–129, 133
Mediator, 129
Mediators, 121
Message filters
  Access control lists, 111
Meta-Object Model, 110
Method based access control, 120
Middleware, 14
Missing data, 237–238
Mix communication network, 34
MLS database, 237, 249
Mobility, 224
Modula-3
  Secure Network Objects, 120
Multilateral security, 28
Multimedia, 138
Multipolicy access control, 120
Multiversion databases, 160
Mumbo
  Object coordination language, 118
Negative information flow, 78, 82, 89
Non-observability, 34
Non-repudiation, 4
Nonuniform access, 162
On-page caching, 161
Optimistic contract signing scheme, 33
Optimistic exchange scheme, 33
ORION
  ITASCA, 120
Personal model of data, 30
Petri nets, 45, 50, 53, 56
Plug-ins, 8
Positive information flow, 89
Potential information flow, 78–81, 84, 88
Principle of Completeness, 78–79
Privacy, 5–7, 12, 18, 35, 77–83, 85–90
Prohibited methods, 211, 215–216
Prolog prototype, 77, 89
Public encryption, 135
Push technology, 8, 10–11
Query processing, 262
Reachability, 85
Role based access control, 115
Role, 128, 136, 191–202
Role-hierarchy, 194, 197
Sandbox, 9
Secure multiversion timestamp, 159
Secure timestamp generator, 164
Secure TransActional Resources-Database System, 159
Security agency, 19
Security models, 77–78, 80–81, 83
Security violation, 4
Security, 4–5, 191–196, 202
Security, 223, 225–228
Software agents, 214, 216
Star-DBS prototype, 165
Static aspects, 79, 85–86
Statistical, 238, 249
Tamper resistant hardware foundation, 34
Task based access control, 116
Technical enforcement, 22
Teleservice, 26
Teleservices Act, 26
Template, 191–193, 196–202
Threat, 22
Ticket based access control, 113
Tickets, 112
Transient versioning, 160
Trust, 22, 32, 77, 83–84, 88
User-role based security, 210
User-role definition hierarchy, 211
VBScript, 7
VBscripts, 8
Virtual machine, 117
Wallet with observer, 33
Workflow Management, 43
World Wide Web, 3
Wrappers, 118
Write-one policy, 162