Volume 26 Number 2 July 2002 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Special Issue:

Security and Protection

Guest Editors:

Tatjana Welzer, Leon Strous

The Slovene Society Informatika, Ljubljana, Slovenia

Informatica **26** (2002) Number 2, pp. 103–243

Informatica

An International Journal of Computing and Informatics

Archive of abstracts may be accessed at USA: http://, Europe: http://ai.ijs.si/informatica, Asia: http://www.comp.nus.edu.sg/ liuh/Informatica/index.html.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2002 (Volume 26) is
- USD 80 for institutions,
- USD 40 for individuals, and
- USD 20 for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

LaTeX Tech. Support: Borut Žnidar, Kranj, Slovenia.
Lectorship: Fergus F. Smith, AMIDAS d.o.o., Cankarjevo nabrežje 11, Ljubljana, Slovenia.
Printed by Biro M, d.o.o., Šmartinska 130, 1000 Ljubljana, Slovenia.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. R. Murn, Jožef Stefan Institute: Tel (+386) 1 4773 900, Fax (+386) 1 219 385, or send checks or VISA card number or use the bank account number 900–27620–5159/4 Nova Ljubljanska Banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, ACM Digital Library, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Computer Literature Index, Cur. Cont. & Comp. & Math. Sear., Current Mathematical Publications, Cybernetica Newsletter, DBLP Computer Science Bibliography, Engineering Index, INSPEC, Linguistics and Language Behaviour Abstracts, Mathematical Reviews, MathSci, Sociological Abstracts, Uncover, Zentralblatt für Mathematik

The issuing of the Informatica journal is financially supported by the Ministry for Science and Technology, Slovenska 50, 1000 Ljubljana, Slovenia.

Post tax payed at post 1102 Ljubljana. Slovenia taxe Percue.

INFORMATICA
AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS
INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of e-mails with the text in Informatica LaTeX format and figures in `.eps` format. The original figures can also be sent on separate sheets. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Rudi Murn, Informatica, Institut Jožef Stefan, Jamova 39, 1111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than five years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: ....

Title and Profession (optional): ....

....

Home Address and Telephone (optional): ....

....

Office Address and Telephone (optional): ....

....

E-mail Address (optional): ....

Signature and Date: ....

Informatica WWW:

http://ai.ijs.si/informatica/
http://orca.st.usm.edu/informatica/

Referees:

Special Issue of the ***Informatica*** - An International Journal of Computing and Informatics

Dedicated to

IFIP-TC 11

This special issue of Informatica includes a selection of high quality papers of IFIP-TC 11 Working Conferences and some additional papers. IFIP is an international non-profit umbrella organization of national societies working in the field of information processing. Since the inception of IFIP in 1960, the information technology (IT) has become a potent instrument, affecting people in many ways. It is a powerful tool in science and engineering, in commerce and industry, in education and administration and in entertainment. IFIP has twelve technical committees, each taking care of a given aspect of information processing.

The name of Technical Committee 11 is "Security and Protection in Information Processing Systems". The aim of TC 11 is to increase the reliability and general confidence in information processing as well as to act as a forum for security managers and others professionally active in the field of information processing security.

The TC 11's activities include the establishment of a common frame of reference for security in organizations, professions and the public domain, the exchange of practical experience in security work, the dissemination of information on and the evaluation of current and future protective techniques, and the promotion of security and protection as essential elements of information processing systems.

TC 11 has seven working groups: Security Management, Small Systems Security, Data and Application Security, Network Security, Systems Integrity and Control, Information Technology Mis-Use and the Law, and Security Education.

Each year several conferences are being organized by working groups where state-of-the-art scientific papers, research-in-progress and industry reports are presented. In order to give an overview of the broad field of information security and to give an impression of current topics, the chairs of the working groups have selected a number of high quality papers that were presented at the conferences. The authors of these papers have been asked to update them and present us their latest findings. In addition, the Guest Editors of this issue have invited several other scientists to submit their contributions.

Initially, 20 papers were received for the present special issue of Informatica, dedicated to Security. The review process and the final selection yielded twelve high-quality papers, which are covering the following research areas: Mobile computing, World Wide Web, Digital signature, Software development and other topics like Transaction processing, XML, Security architecture and Information integrity.

The paper "Employing an Extended Transaction Model in Multilevel Secure Transaction Processing", by V. Atluri and R.Mukkamala, deals with security issues in multi-level transaction processing. The authors argue that the proposed secure concurrency control protocols are not completely satisfactory from the performance point of view and propose a better approach that preserves the semantics of the transactions and guarantees serializability.

W. Brandi and M. Oliver cover the aspect of safety in mobile environments in the paper, "Maintaining Integrity within mobile Self Protecting Objects". The authors examine the integrity issues involved when a self-protecting object is moved between different sites which may become unavailable for some time. Their model guarantees that the security policies of participating databases in a federated database are respected regardless of the object's location.

The security and functionality weaknesses in both current e-commerce systems and emerging wireless-only systems is examined in the paper titled "Combining World Wide Web and wireless security" by J. Claessens, B. Preneel and J. Vandewalle. They argue that mobile devices are shown to be an alternative to costly special-purpose hardware and that the combination of both worlds has in many cases more interesting properties than when using mobile devices only.

The paper, "XML Access Control Systems: A Component-Based Approach" by E. Damiani, S. De Capitani, S. Paraboschi and P. Samarati, describes the design and implementation of an access control processor that permits the definition of authorization at a fine granularity for an access control model for XML information.

How aspect-oriented programming can help to overcome the problem of unmanageable code which is a consequence of implementing security mechanisms, which interferes with the core functionality of a system is described in the paper, "How aspect-oriented programming can help to build secure software" by B. de Win, B. Vanhaute and B. de Decker. The authors investigate how well the state of the art in aspect-oriented programming can deal with separating security concerns from an application.

The authors J. Janaček and R. Ostertág point out the security issues in practical use of electronic signatures in the paper, "Problems in Practical Use of Electronic Signatures". They propose using a dedicated system to create electronic signatures since state of the art systems are far from being perfect. They describe a cheap and reasonably secure solution.

"Securing Web-based Information Systems: A Model and Implementation Guidelines" by C. Margaritis, N. Kolokotronis, P. Papadopolou, P. Kanellis and D. Martakos outlines an integrated approach based on a rigorous multi-level and multi-dimensional model to design and implement information security, while keeping in focus overall business goals and objectives. The approach is demonstrated by an example.

V. Mandke and M. Nayar present an approach to cost-benefit analysis of information integrity in the paper entitled, "Implementing Information Integrity Technology – A Feedback Control System Approach". Further, the authors examine quantitive measures of integrity attributes, such as accuracy, consistency and reliability. Based on those, they discuss a choice of information model for integrity improvement. The framework of feedback control system approach enables presentation of information integrity implementation steps.

The authors of the paper, "Efficient methods for checking integrity: A structured Spreadsheet Engineering Methodology", K. Rajalingham, D. Chadwick and B. Knight, address the widespread problem of spreadsheet errors. They propose a structured spreadsheet engineering methodology for spreadsheet design and development, with focus on integrity control of spreadsheet models.

A. Spalka, A. Cremers and H. Langweg deal with attacks on software for electronic signatures in the paper, "Trojan Horse Attack on Software for Electronic Signatures".The underlying cryptographic methods for electronic signatures are sufficiently strong and thus non-brute force methods are being used, such as Trojan Horse attacks. In the contribution the authors propose a secure electronic paper as a counter-measure, which offers a high degree of protection of the system against untrustworthy programs.

In the paper, "Data Protection for Outsourced Data Mining" the authors B. Brumen, M. Družovec, I. Golob, T. Welzer, I. Rozman and H. Jaakkola research how data in a knowledge discovery process can be protected. Many organizations have a lack of expertise in data mining and require outside help, and this poses a possible threat. The authors present a formal framework for data and structure transformation by preserving the possibility to implement the modeling process.

The paper "A Security Architecture for Future Active IP Networks", by A. Savanović, D. Gabrijelčič and Borka Jerman-Blažič, describes a security framework to assure that infrastructure will behave as expected and will efficiently deal with malicious attacks, unauthorized attempts to execute active code etc. The authors present a security architecture that is designed within the FAIN project and aims at supporting multiple heterogeneous execution environments.

We kindly extend our sincerest thanks to all the authors of the papers in the present issue of Informatica for their update efforts and timely contribution. Our gratitude goes to all the reviewers for their superb and timely work. We hope the papers we selected will be as informative and useful for the readers as they are for us.

Finally, we would like to encourage the readers to participate in the interesting and rewarding work that many professionals are undertaking in the IFIP TC-11 community. More information and contact details can be found on the IFIP website (http://www.ifip.or.at) and on the website of TC-11 (http://www.ifip.tu-graz.ac.at/TC11). For a list of valuable publications we refer to the website of the IFIP publisher, Kluwer Academic Publishers where they have a special section for IFIP books (http://www.wkap.nl/prod/s/IFIP).

Guest Editors:
Tatjana Welzer
Leon Strous

Reviewers for this issue (in alphabetical order):

- *Bart de Decker, Belgium*
- *Marjan Družovec, Slovenia*
- *Simone Fischer-Huebner, Sweden*
- *Michael Gertz, USA*
- *Jozsef Györkös, Slovenia*
- *Jaak Henno, Estonia*
- *Sushil Jajodia, USA*
- *Les Labuschagne, South Africa*
- *Herbert Leitold, Austria*
- *Jari Multisilta, Finland*
- *Daniel Olejar, Slovakia*
- *Kai Rannenberg, UK*
- *Pierangela Samarati, Italy*
- *Leon Strous, The Netherlands*
- *Jan Verschuren, The Netherlands*
- *Jozef Vyskoc, Slovakia*
- *Matthew Warren, Australia*
- *Tatjana Welzer, Slovenia*

Employing an extended transaction model in multilevel secure transaction processing

Vijayalakshmi Atluri
MSIS Department and CIMIC
Rutgers University
Newark, NJ 07012
atluri@andromeda.rutgers.edu
AND
Ravi Mukkamala
Department of Computer Science
Old Dominion University
Norfolk, VA 23529
mukka@cs.odu.edu

Keywords: Multilevel Security, Transaction Processing, Advanced Transaction Model

Received: January 23, 2002

Multilevel secure transaction processing has been well explored in the past decade. Despite this research, the proposed secure concurrency control protocols are not completely satisfactory because of the stringent constraints imposed by multilevel security. In this paper, we argue that modeling a transaction as an extended transaction model could significantly reduce the performance penalty. We accomplish this by minimizing (1) the probability of restarting a high security level transaction; and (2) the portion of the transaction to be reexecuted, when a restart becomes inevitable. In particular, we exploit the non-flat nature of transactions by identifying dependencies among various components of a transaction and portraying a transaction as an advanced transaction model. We demonstrate, via formal proofs, that our approach preserves the semantics of the transaction, and our concurrency control algorithm guarantees serializability.

1 Introduction

Access control requirements on sensitive data can be broadly classified into two categories: (1) *discretionary access control* (DAC), in which users, at their discretion, are allowed to grant permissions on the data they own, and (2) *mandatory access control* (MAC), in which all data are labeled based on their level of sensitivity and all users are given clearances, where users are allowed to access data with a given label only if their level of clearance permits them. Although many commercial applications employ DAC, it is not adequate where more stringent security requirements are needed as they are vulnerable to sophisticated attacks, such as Trojan Horse Attacks. Such application domains call for multilevel secure (MLS) systems that enforce MAC.

The area of transaction processing in multilevel secure (MLS) database management systems (DBMSs) has substantially progressed in the past few years. The major emphasis of this progress can be seen in secure concurrency control, one of the essential components of secure transaction processing. Concurrency control aims at synchronizing the operations of concurrent transactions to ensure correct execution.

Despite the substantial research activity, the proposed secure concurrency control protocols are not completely satisfactory because of the stringent constraints imposed by multilevel security, such as elimination of all covert channels [15,12]. Researchers have proposed transaction processing protocols, in particular, concurrency control protocols using locking, timestamping, hybrid (combination of locking and timestamping) techniques. Other solutions try to tackle this difficult problem by maintaining multiple versions of data. Unfortunately, these solutions do not meet all the requirements of a secure concurrency control protocol; they trade either performance or recency for achieving security.

More specifically, the problem with the secure transaction processing is that if a data conflict between a low security level transaction and a high security level transaction should occur, for security reasons, the conflict must *always* be resolved in favor of the low security level transaction. In other words, the low level transaction has the right of way, whereas the high level transaction is hindered or restarted.

In the cases where a transaction has to be restarted, some solutions require that all of the transaction's effects must be undone and then the transaction is reexecuted from the beginning [1]. Other solutions reexecute starting from the first low read operation [8,5]. These approaches however, cause resource wastage and in some cases results in starvation of high security level transactions. Much of the prior research in this area has been limited to the traditional transaction concepts.

Motivated by the need for releasing the rigidity, traditional transaction models have been extended in various directions. The objective of this paper is to investigate how the properties of the extended transaction models can be utilized to yield better secure concurrency control solutions for MLS DBMSs.

We have recognized that modeling a transaction as an extended transaction model, in particular as a workflow model, could reap significant gains in performance. In our work, we explore mechanisms that are specifically tailored for reducing the performance penalty to be paid for restarts by designing protocols such that (1) the probability of restarting a high security level transaction is minimal; and (2) even when a restart becomes inevitable, the portion of the transaction to be reexecuted is minimal.

This paper is organized as follows. In section 2, we review the related work in this area. In section 3, we introduce the preliminaries of our extended transaction model and the security model. In section 4, we present an overview of our extended transaction model approach. In section 5, we present our secure concurrency control algorithm using the extended transaction model. We also provide proofs of correctness that demonstrate that our redesign algorithm does indeed generate extended transactions that are equivalent to the original transactions, and the concurrency control algorithm guarantees (one-copy) serializability. Finally, in section 7, we summarize the results and discuss future work.

2 Related Work

Secure concurrency control with traditional transaction models has been well understood [1,8,9,10,11]. In the following, we only review the solutions that are closely relevant to our work. The protocol proposed by Keefe and Tsai [9] uses a priority queue where transactions are placed in the queue based on the security level of the transaction. A transaction is assigned a timestamp that is smaller than all active transactions executing at lower security levels. As a result, a *high* transaction's read operation does not interfere with a *low* transaction's write operation since the *high* transaction is considered as an older transaction though it arrives later than the *low* transaction. Although this requires a trusted code for implementation, it ensures one-copy serializability (the usual notion of correctness when multiple versions are maintained in the database) without compromising security. Moreover, transactions are never subjected to starvation. However, high transactions are sometimes given stale versions of the low data.

The protocol proposed by Jajodia and Atluri [8] assigns timestamps to transactions as in a conventional timestamp ordering protocol. According to this protocol, transactions are sometimes made to wait for their commit. Whenever a *high* transaction reads *low* data, it is not allowed to commit until all *low* transactions with smaller timestamps than that of itself commit. This is because a *low* level transaction with a smaller timestamp can always invalidate the read operation of the *high* transaction by issuing a write. In such an event, the *high* transaction is reexecuted. This protocol guarantees one-copy serializability, is secure, and free of starvation. In addition, it can be implemented with completely untrusted code. Restarting a transaction may have significant impact on performance, especially if they are not short lived.

Researchers have proposed other solutions that compromise correctness for ensuring security. Jajodia and Atluri [8] have proposed three increasingly stricter notions of correctness for multiversion multilevel databases — *level-wise serializability, one-item read serializability* and *pairwise serializability* — that are weaker than one-copy serializability.

Bertino et al. [5] proposed a lock-based scheme for implementing transactions in a multilevel secure file storage system. Assuming the availability of a *trusted* lock manager and a *trusted* file manager, they implement a covert-channel-free MLS system. The main contribution of their work is the introduction of a "signal lock" along with the usual read lock and write lock in a 2PL scheme. They do assume a single version for each data item and use strict 2PL for concurrency control for transactions at the same security level. A high-level transaction intending to read a low-level data item obtains a signal-lock on the item. If the item is already write-locked by a low-level transaction, then the high-transaction has to wait until the write-lock is released. A low writer however does not need to wait when a data item is signal-locked by a high-level reader. In addition, they provide language primitives by which a high-level transaction can roll-back to a prespecified step when it receives a signal from the low-level indicating that the data item that it read has now been written. A transaction can then rollback to the state prior to the read statement. Similar strategy is also explained by Ray et al. in [13]. Both the approaches in [5,13] incur the same penalty in performance as that in [8].

Our work deviates from prior work significantly. The novel part of our work is that we exploit the non-flat nature of transactions by identifying the dependencies among various components of a transaction. In other words, we redesign a transaction and model it as an advanced transaction model. As a result of this redesign, we can minimize the number of restarts or even unnecessary rollbacks to a prior save-state.

3 The Model

In this section, we present our extended transaction model, and review the multilevel security model. Since our approach assumes multiple versions of data, we review the multiversion serializability theory in Appendix A.

3.1 The Extended Transaction Model

Driven by the needs of advanced application domains such as design, engineering, manufacturing and commerce, several extended transaction models have been proposed [7]. Many of these models recognize the fact that the transactions are not always flat. Some such extended models include nested transactions and workflow transactions. There has been an established model of nested transaction processing [3]. The model describes a way to ensure the correctness under the scenario where a transaction is split-up into various subtransactions. Splitting of a transaction into subtransactions based on semantic information is described in [2,14]. Hence, a transaction can be subdivided into meaningful subtransactions with dependencies among them. In the following, we develop the necessary formalism of the extended transaction model employed in this paper.

Let D be the set of all data objects, and $\mathcal{DO}$ be the set of all data operations. That is, $\mathcal{DO} = \{r[x], w[x] | x \in D\}$, where $r[x]$ and $w[x]$ denote the read and write operations on x. Let T_i be a transaction. A transaction in its simplest form consists of a set of data operations (DO) and task primitives {begin, abort, commit} (PO). We use $r_i[x]$ and $w_i[x]$ to denote the read and write operations issued by a transaction T_i on a data item x, b_i, a_i and c_i to denote the begin, abort and commit primitives of T_i. Formally,

Definition 1 A transaction T_i is a partial order with an ordering relation $<_i$ where

1. $T_i = DO_i \cup PO_i$ where $DO_i \subseteq \mathcal{DO}$ and $PO_i \subset \{b_i, a_i, c_i\}$;
2. $c_i \in T_i$ iff $b_i \in T_i \wedge a_i \notin T_i$, and $a_i \in T_i$ iff $b_i \in T_i \wedge c_i \notin T_i$;
3. for any $o_i \in DO_i$, $b_i <_i o_i <_i$ either c_i or a_i (whichever is in T_i); and
4. if $r_i[x], w_i[x] \in T_i$, then either $r_i[x] <_i w_i[x]$ or $w_i[x] <_i r_i[x]$.

The above traditional definition of the transaction includes only the database operations. However, a transaction, in general, is a procedure, which may include assignment statements that may involve computations, as well as conditional statements. Our intention precisely is to represent these formally into our extended transaction model.

In this paper, we recognize two types of dependencies among subtransactions within a transaction: data dependency and value dependency. These two dependencies, defined below, are to capture these additional statements of the transaction. In addition, we recognize another type of dependency, which introduces a dependency between two different transactions. We use T_{ij} to represent the jth subtransaction of T_i.

Definition 2 Assume T_{ik} consists of an assignment statement $x = f(d_1, \ldots d_n)$, where $x, d_1, \ldots d_n \in D$, and f is a mapping. We say that T_{ik} *reads-from* T_{ij}, if there exits a write operation $w[x]$ in T_{ik} and a read operation $r[y]$ in T_{ij} such that $y \in \{d_1, \ldots d_n\}$.

Definition 3 We say that a data dependency $T_{ij} \rightarrow T_{ik}$ exists, if T_{ik} *reads-from* T_{ij}.

Definition 4 Let exp be an expression over the data items $d_1, \ldots d_n \in D$. If T_{ij} comprises of a data operation $o[x]$ such that $x \in \{d_1, \ldots d_n\}$, and a conditional statement specifying that T_{ik} has to be executed only if exp is true, then we say that there exists a value dependency $T_{ij} \xrightarrow{exp} T_{ik}$.

Example 1 In the following, we present an example of a real-world database scenario where the above two dependencies exists. Consider a corporate database that contains several types of information about its employees: social security number, name, address, rank, years of service, weekly pay (p), number of hours worked per week (h), number of overtime hours worked per week (o), hourly rate (r), additional compensation (c), yearly bonus, etc. The information such as weekly pay, hourly rate and additional compensation are considered sensitive, and are labeled *high*, while the rest of the information is classified *low*.

Thus, $L(h) = L(o) = low$, and $L(p) = L(c) = L(r) = high$.

At the end of each week, the number of hours worked by an employee as well as the weekly pay of each employee is computed. The weekly pay of each employee is computed as follows:

Pay = ((number of regular hours + number of overtime hours)*hourly rate) + c, if overtime hours ≥ 20
= ((number of regular hours + number of overtime hours)*hourly rate), otherwise.

That is there will not be any additional compensation if an employee works less than 20 overtime hours per week. Since h and o are *low* data items, a *low* transaction (T_1) is initiated to update h and o. A *high* transaction (T_2) is initiated to compute the weekly pay. They are as follows:

$$\begin{array}{ll} T_1 = & w[h]\, w[o] \\ T_2 = & r[h]\, r[o]\, r[r] \\ & r[c]\ [p = ((h + o) * r) + c]\ w[p], \text{ if } o \geq 20 \\ & [p = (h + o) * r]\ w[p], \text{ otherwise} \end{array}$$

Based on the semantics of the transaction, the dependencies as shown in Figure 1 can be realized among the operations within a transaction. Note that there exist both data dependencies as well as value dependencies.

The third type of dependencies that we present below are specified between either subtransactions of two *different* transactions, or between two transactions. We assume that every (sub)transaction is associated with a set of transaction management primitives that move the (sub)transaction from one state to the other.

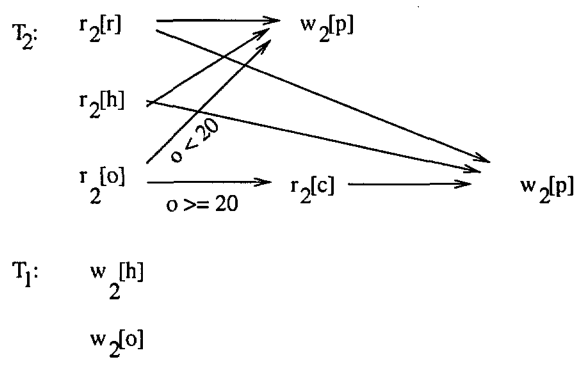


Figure 1: An extended transaction model for T_1 and T_2

For each transaction and subtransaction, we assume the following set of primitives and states:

	primitives	states
transaction	{begin, precommit, commit, abort}	{initial, done, commit, abort}
subtransaction	{begin}	{initial, done}

Definition 5 A control-flow dependency $T_{ij} \xrightarrow{s_{ij},p_{kl}} T_{kl}$ states that primitive p_{kl} of T_{kl} is invoked only if T_{ij} enters state s_{ij}.

Although several types of control-flow dependencies may be defined based on the above states and primitives [6], in this paper, we use the following two.

1. $T_{ij} \xrightarrow{bd} T_{kl}$: This states that T_{kl} "begins" only if T_{ij} enters a "done" state.

2. $T_i \xrightarrow{t} T_j$: This states that T_j "terminates" (either commit or abort) only if T_i "terminates."

3.2 The Multilevel Security Model

We assume the security structure to be a partially ordered set S of security levels with ordering relation $\leq$. A class $s_i \in S$ is said to be *dominated* by another class $s_j \in S$ if $s_i \leq s_j$.

Let D be the set of all data objects. Each data object $d \in D$ is associated with a security level. Every transaction T_i is associated with a security level. We assume that there is a function L that maps all data objects and tasks to security levels. That is, for every $d \in D$, $L(d) \in S$, and for every transaction $T_i \in W$, $L(T_i) \in S$. We require every transaction to obey the following two security properties — the simple security and the restricted $\star$-property.

1. A transaction T_i is allowed to read a data object d only if $L(d) \leq L(T_i)$

2. A transaction T_i is allowed to write to a data object d only if $L(d) = L(T_i)$.

In addition to these two restrictions, a *secure* system must prevent illegal information flows via covert channels.

4 Overview of the Extended Transaction Model Approach

Our approach to achieving efficient execution of multilevel transactions relies on first recognizing the data dependencies among the various operations (sets of operations), and the non-flat nature of a transaction.

As mentioned in section 1, the goal of our approach is to reduce the amount of reexecution of a transaction when restart becomes inevitable. In the following, we describe three cases where such optimizations are possible. In the first case, a restart is necessary, however, the transaction need not have to be restarted in full. In the second case, a particular data modification by a low security level transaction does not necessitate any form of corrective action by a high security level transaction. In the third case, a data modification may require some corrective action, but the compensatory action is pre-specified and can be efficiently accomplished rather than restarting the transaction. All these three techniques can also be used in conventional transaction processing systems. However, we expect that using them in the secure transaction processing scenario will significantly enhance the performance.

4.1 Exploiting the non-flat nature of a transaction

Let us consider a transaction T_1 with subtransactions and dependencies as shown in Figure 1.

As can be seen from this example, the subtransaction T_{17} comprises of $r[x]$ where $L(x) < L(T_1)$. Suppose T_1 reads x. But T_1 may have to be reexecuted due to a conflicting operation by a lower level transaction to ensure serializable execution. In such a scenario, not all parts of the transaction need to be reexecuted.

Our approach exploits the dependencies among subtransactions so that only those subtransactions which are effected by the modified low data item are reexecuted. For example, in Figure 1, subtransactions T_{11}, T_{12}, T_{16}, T_{18}

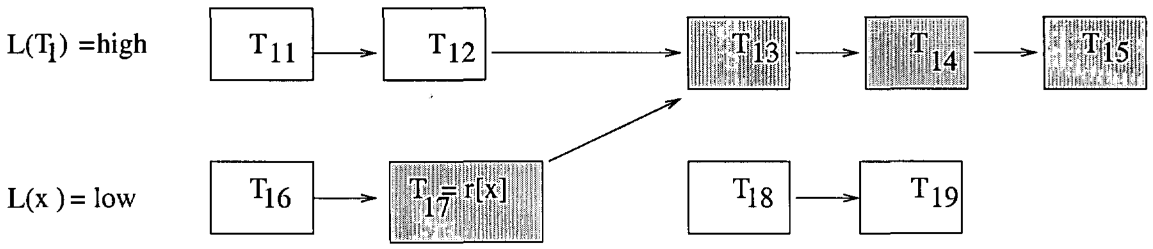


Figure 2: An example of partial reexecution

and T_{19} are not required to be reexecuted for ensuring correctness since the change in the value of x does not affect those subtransactions.

4.2 Exploiting the value dependencies among subtransactions

Consider the scenario featured in Figure 2. Depending on the value of x, the transaction could follow either the execution sequence A (T_{18}, T_{19}) or the sequence B (T_{13}, T_{14}, T_{15}). Now assume that the value of x is 5 when the high level transaction begins its execution. Since the condition $x < 20$ is satisfied, the transaction follows the sequence A. Now if the value of x was changed to 6 at a later point of time due to a lower security level transaction, the transaction need not roll back since the condition $x < 20$ still holds. Therefore, by modeling this as a value dependency, one may evaluate the condition and thereby avoid complete transaction reexecution. Only the necessary operations could be reexecuted when necessary.

4.3 Employing compensating subtransactions

Now consider the case in Figure 3. In this case, the value of x changes from 6 to 10, while the high-level transaction is executing. In a conventional scenario, this requires a transaction restart. However, the restart could be prevented here by executing a compensating module.* We realize that devising a compensating action is much more difficult, and moreover, some subtransactions may not be compensatable. In this paper, we do not address this issue.

5 Proposed Concurrency Control Algorithm

In this section, we describe the concurrency control that we propose. It is described in terms of a scheduling algorithm (Algorithm 1) and a restructuring algorithm (Algorithm 2).

Algorithm 1 [The Scheduler]

1. There is a separate scheduler for each security level s.

2. Whenever a new transaction T_i arrives, the scheduler performs the following steps:

   (a) Assign a unique timestamp $ts(T_i)$. We assume that there is a shared hardware clock at system low which is used by each scheduler at level s to assign a unique timestamp $ts(T_i)$ to every transaction T_i at its level.

   (b) Each transaction T_i declares its write- and read-sets. The scheduler at level s maintains $W(T_i)$ and $R(T_i)$.

   (c) If there exists a $r_i[x]$ such that $L(x) = s'$ and $s' < L(T_i)$,

      i. find every T_j such that $L(T_j) = s'$, $ts(T_j) < ts(T_i)$ and, $w_j[x] \in W(T_j)$. Add dependencies $w_j[x] \xrightarrow{bd} r_i[x]$ and $c_j \xrightarrow{t} c_i$.

      ii. Restructure T_i according to the restructuring algorithm (Algorithm 2).

3. Each version x_j of a data item x has a read timestamp $rts(x_j)$ and a write timestamp $wts(x_j)$ associated with it. We assume there is an initial transaction T_0 that writes into the database with $rts(x_0) = wts(x_0) = ts(T_0)$.

4. When a transaction T_i wants to read a data item x and if $L(T_i) = L(x)$, the scheduler selects a version x_k with the largest $wts(x_k)$ such that $wts(x_k) < ts(T_i)$, processes $r_i[x_k]$, and modifies $rts(x_k)$ as $rts(x_k) = \text{maximum}\{ts(T_i), rts(x_k)\}$.

5. When a transaction T_i wants to write a data item x, scheduler selects a version x_k with the largest $wts(x_k)$ such that $wts(x_k) < ts(T_i)$. It rejects $w_i[x]$ if $rts(x_k) > ts(T_i)$; otherwise, it processes $w_i[x_i]$ and modifies the timestamps of the new version x_i as $rts(x_i) = wts(x_i) = ts(T_i)$.

6. Suppose a transaction T_i wants to read a data item x. If $L(T_i) > L(x)$, the scheduler at level $L(T_i)$ selects a version x_k with the largest $wts(x_k)$ such that $wts(x_k) < ts(T_i)$. Although the operation $r_i[x_k]$ is processed, the read timestamp $rts(x_k)$ is not modified for security reasons.

Algorithm 2 [The Restructuring Algorithm]

*Note that this compensatory action is different from that in recovery where only the correctness of the database is the issue and not the minimization of transaction restarts.

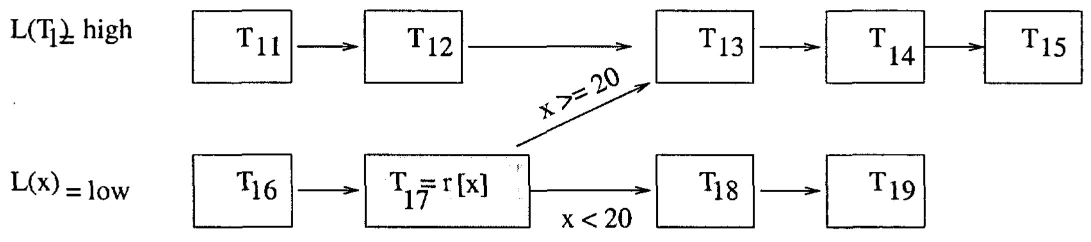


Figure 3: An example of value dependency approach

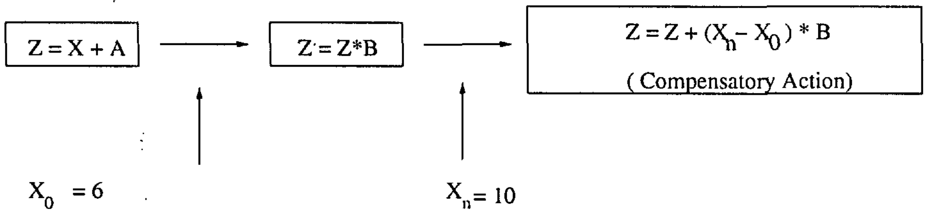


Figure 4: An example of compensation

1. Add data and value dependencies (Definitions 3 and 4) between the operations of each transaction based on their semantics, as identified in definitions 3 and 4.

2. If there exists a value dependency $d_k : o_i[x] \xrightarrow{exp} o_i[y]$, replace d_k with the following: $o_i[x] \xrightarrow{exp} inc[counter_k] \xrightarrow{counter_k<2} o_i[y]$.

5.1 Discussion

According to Step 2(c) of algorithm 1, the restructuring of a transaction is done only if a transaction reads from low. Otherwise, it is executed as a flat transaction. If it reads from low, the first step of algorithm 2 is to exploit the partial order of operations within the transaction and the data dependencies among them. In the second step of algorithm 2, we add counters to make sure that the operations are redone only if the condition has changed since the prior read. Let us consider various cases:

1. Suppose the schedule (only with the necessary operations) in the above example is

   $T_2 :$ $\qquad\qquad r_2[h]\ r_2[o]$
   $T_1 : w_1[h]$ $\qquad\qquad\qquad\qquad w_1[o]$

   Then the $r_2[o]$ operation is redone after $w_1[o]$ because of the "bd" dependency, as shown in Figure 5. Therefore this will result in a serializable schedule.

2. Suppose the schedule is

   $T_2 :$ $\qquad\qquad r_2[h]$ $\qquad\qquad r_2[o]$
   $T_1 : w_1[h]$ $\qquad\qquad w_1[o]$

   The above $r_2[o]$ is in fact due to the "bd" dependency, as shown in Figure 5. Therefore this is also correct.

So by adding these two dependencies, it makes sure that the write of low precedes the read of high if it hasn't already occurred in that order. That is, only the out-of-order read operation is redone but not all. This way, we are saving on the number of redones. The dotted arrows indicate the added dependencies between transactions according to step 2(c)(i) of algorithm 1. Moreover, according to step 1 of the restructuring algorithm, only parts of the transaction are to be redone, but not all.

Now the idea of the second step of the restructure algorithm is to provide further optimization. The objective is to exploit the nature of the value dependency to avoid the redoing certain parts of the transaction by examining the value. For example, during the high transaction execution, the low value of "o" has changed. This does not have to necessarily trigger a redo. Consider the value read by $r[o]$ is 10. Now it has changed to 15. This should not have any effect on the redoing part, and therefore, the $w[p]$ operation (in Figure 6) does not have to redone. This is accomplished by incorporating counters in that path and incremented whenever transaction executes that part of the transaction.

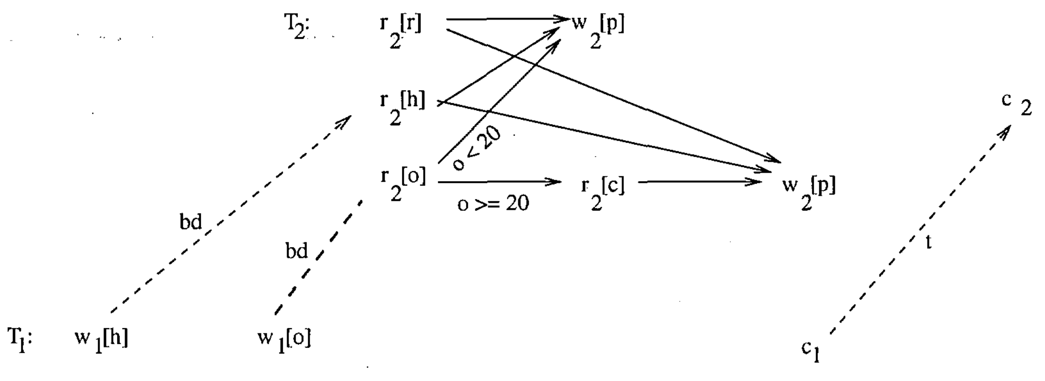


Figure 5: Dependencies added between T_1 and T_2 as in step 1 of algorithm 1

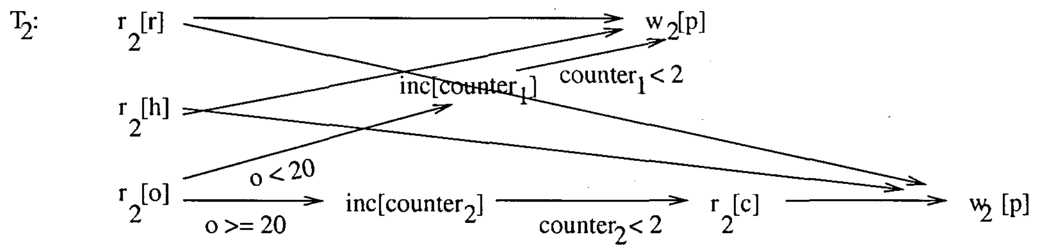


Figure 6: Exploiting the value dependencies in T_2

5.2 Proof of Equivalence

Definition 6 We say that a transaction T_i is *semantically equivalent* to another transaction T_j, (i) if the set of data operations of both T_i and T_j are equal, and (ii) if the values read by each read operation in both T_i and T_j are the same, then the values written by each write operation in T_i and T_j are the same.

Theorem 1 Let T_i be a transaction. Let T_j be the restructured transaction using algorithm 2. Then T_j is semantically equivalent to T_i.

Proof: We prove this in two parts, related to the two steps of algorithm 2.

First, we prove that T_j, generated after adding data dependencies, is semantically equivalent to T_i. By adding dependencies, neither new data operations are added nor the existing ones are deleted. Therefore, $DO_i \equiv DO_j$. Hence, the first condition of definition 6 is trivially satisfied. If there is a $reads - from$ relation between two subtransactions, then there will be a data dependency. Since the data dependencies capture all the $reads - from$ relationships within a transaction, if all the read operations in T_j read the same values as the read operations in T_i, the write operations in T_j will write the same values as those in T_i. Hence, the second condition of definition 6 is true.

Second, we prove that the T_j generated by adding additional dependencies and increment operations as in step 2, is semantically equivalent to T_i. The addition of increment operations does not change the original data operations of T. Therefore, $DO_i \equiv DO_j$. Hence, condition 1 of definition 6 is true. The effect of the replacement of $o_i[x] \xrightarrow{exp} o_i[y]$ with $o_i[x] \xrightarrow{x} inc[counter_k] \xrightarrow{counter_k<2} o_i[y]$ is as follows: initially, before the execution of the transaction, the value of $counter_k$ is zero. When exp is true, then $counter_k$ will be incremented to 1. So the condition $counter_k < 2$ is satisfied. Hence, $o_i[y]$ is executed. However, if $o_i[x]$ is performed again, and if x is again true, then $counter_k$ will be incremented to 2. In this case, the condition $counter_k < 2$ is not satisfied. Hence, $o_i[y]$ is not executed. Since Thus, addition of these dependencies does not effect the operation $o_i[y]$. These additional dependencies simply avoid redoing of $o_i[y]$ and its subsequent operations. In other words, the $reads - from$ relations are kept in tact in T_j and do not get affected. Therefore, if the read operations in T_j read the same values as those in T_i, the write operations in T_j write the same as those in T_i. Hence the second condition of definition 6 is true. □

5.3 Proof of Correctness

Theorem 2 Let H be a multiversion history produced by algorithm 1. Then H is one-copy serializable. In fact, H is equivalent to a one-serial history in which transactions are placed in a timestamp order.

Lemma 1 If there is an edge $T_i \rightarrow T_j$ in $MVSG(H)$ (multiversion serialization graph described in the appendix) such that $L(T_i) < L(T_j)$, then $ts(T_i) < ts(T_j)$.

Proof: If there is an edge $T_i \rightarrow T_j$ in $MVSG(H)$ such that $L(T_i) < L(T_j)$, it follows that there must exist a data item x such that $L(x) = L(T_j)$ and $r_j[x_i] \in H$. Form step 7 of algorithm 1, it follows that $wts(x_i) < ts(T_j)$.. From step 4 of this algorithm, $wts(x_i) = ts(T_i)$. The above two expressions can be combined and thus, $ts(T_i) < ts(T_j)$. □

Lemma 2 If there is an edge $T_i \rightarrow T_j$ in $MVSG(H)$ such that $L(T_i) = L(T_j)$, then $ts(T_i) < ts(T_j)$.

Proof: Since we are using multiversion timestamp algorithm, (refer theorem 5.5 in [4]) all edges follow the timestamp order. Therefore, if there is an edge $T_i \rightarrow T_j$ such that $L(T_i) = L(T_j)$, then $ts(T_i) < ts(T_j)$. □

Lemma 3 If there is an edge $T_i \rightarrow T_j$ in $MVSG(H)$ such that $L(T_i) > L(T_j)$, then $ts(T_i) < ts(T_j)$.

Proof: Let $L(T_j) = s$. If there is an edge $T_i \rightarrow T_j$ in $MVSG(H)$ such that $L(T_i) > L(T_j)$, this edge implies that there must exist a data item x such that $L(x) = s$ and $r_i[x_n], w_j[x_j] \in H$ with $x_n \ll_s x_j$. Now, the following two cases must be considered.
Case 1: Suppose $ts(T_i) < ts(T_j)$. This does not require any proof.
Case 2: Suppose $ts(T_i) > ts(T_j)$. In such a case, as per step 7 of algorithm 1, T_i is redesigned. That is, there exist $r_i[x_n] < w_j[x_j]$. In this case, when $w_j[x_j]$ is executed, the dependency $w_j[x] \xrightarrow{bd} r_i[x]$ is triggered. Due to this triggering, $r_i[x]$ is performed again. Since $wts(x_j) < ts(T_i)$, T_i reads the version written by T_j. Therefore, T_i reads a version x_n of x such that either $x_n = x_j$ or $x_n \gg_s x_j$. Hence, $T_i \rightarrow T_j$ disappears. Note that this is not same as the repeated reads because all the operations that depend on $r_i[x]$ are also redone. So the previous $r_i[x]$ and all the subtransactions that follow $r_i[x]$ are removed from H. Instead of $T_i \rightarrow T_j$, we now have $T_j \rightarrow T_i$. □

Proof of Theorem 2: From lemmas 1, 2 and 3, it follows that all edges in $MVSG(H)$ follow the timestamp order. Therefore, there cannot be a cycle. Hence, from theorem 3, H is one-copy serializable. □

6 The System Architecture and Implementation

The proposed system can be implemented in several ways. Here, we describe one such implementation. The system architecture used for the implementation is shown in figure 7. For the sake of simplicity, we use three hierarchical security levels. We assume that our system is implemented using the kernelized architecture. Thus, both the transaction manager (TM) and the scheduler are untrusted. (We assume the functionalities of algorithms 1 and 2 to be implemented in the TM and the scheduler.) The TM at any security level can view all the transactions at levels dominated by its level. When a new transaction is received by the appropriate level TM, it assigns the transaction a unique timestamp by reading from a clock at system low. We assume that the granularity of the clock is high enough so that the timestamps are unique among the transactions at all the security levels.

A TM, upon receiving a transaction, invokes its redesign module. This module first identifies the low-level reads, and their dependencies with write operations at its own level. It builds both the value and data dependencies among the operations, as per algorithm 2. We refer to this as the dependency graph for a transaction. At this stage, however, there is no knowledge of the actual low-level transactions that the high-level transaction under consideration may be in conflict with.

The TM, now, sends low-level read requests to the scheduler at its level. Since we assume a timestamp-based concurrency control, each read request is associated with a timestamp of the transaction that originated it. Accordingly, the scheduler selects the value of the latest version whose write timestamp is smaller than that of the transaction issuing the read request. Since we employ a kernelized architecture, these read requests are sent to the trusted OS to retrieve the data from the appropriate DB. In addition to the data, the request also reads the active transaction log maintained by the scheduler, and determines the active transactions and their timestamps that conflict with its own read. (An active transaction is one that has started but not yet terminated.) Thus, the low-level read request returns the data value, its timestamp, and a set (possibly empty) of active low-level transactions that conflict with the high-level read.

The high-level TM, after receiving the transaction sets from the low-level read operations, saves them for use at the time of transaction commit. Actually, it is only concerned with low-level transactions with timestamps smaller than the high-level transaction and with read-write conflicts. Let us call this set the active-transaction set (ATS).

At the time of transaction commit, in order to enforce the termination dependencies (as shown in figure 5), the TM needs to determine the state of the transactions in the ATS. Accordingly, it sends "read-log" request to the low-level DBMS through the TCB. In return, it is presented with the current list of low-level active transactions. If none of the transactions in its ATS are still active, then it proceeds with the next step of examining the dependency graph. Otherwise, it waits and repeats the above two steps periodically. This is how the "bd" dependencies, as shown in figure 5 between the subtransactions of two transactions are implemented.

When all transactions in the active-set have terminated, the high-level TM resends all low-level read operations to the low-level scheduler. When the new values are returned, it checks with the dependency graph to determine the op-

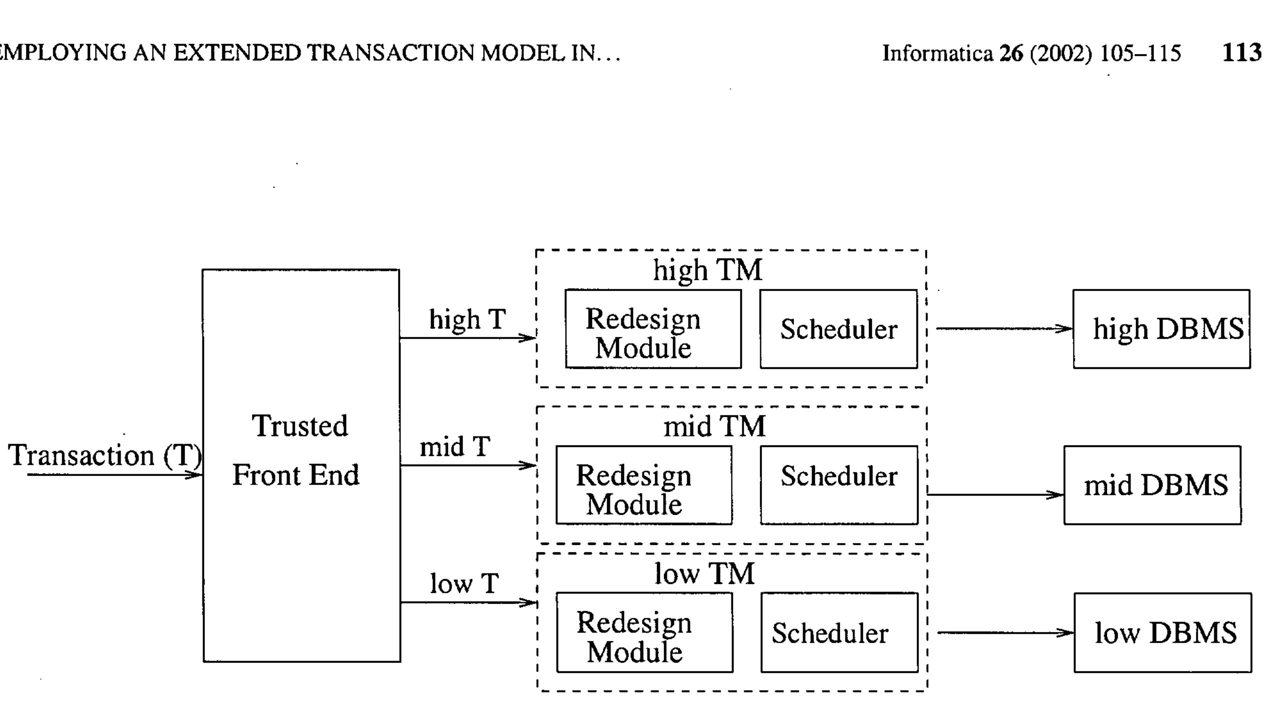


Figure 7: The System Architecture

erations that need to be reexecuted. The reread values are used for both data and value dependencies.

In summary, our implementation uses untrusted TMs and schedulers at each security level, logs maintained by the schedulers indicating the current active transactions and their read- and write-sets, and multiple versions of data items.

7 Summary and Future Work

In this paper, we described an extended model for multilevel secure transaction processing. In a traditional MLS model, a high-level transaction is aborted and restarted whenever its low-level reads conflict with low-level transactions' writes. This is often a concern for high-level users in real systems. Here, we suggest a way to extend the rigid transaction model by using some of the workflow transaction redesign techniques. The proposed scheme uses timestamping as a means of concurrency control. In addition, it employs multiversion data. It employs value and data dependencies to identify the relationship between low-level read data and high-level writes within a transaction. This information is then used at the time of transaction commit. Depending on the conflicts during execution between a high-level transaction and low-level transactions, only some parts of a high-level transaction may need to be reexecuted. But what parts, if any, are to be reexecuted is determined by the data and value dependencies identified earlier and the actual conflicts and data values. We show the correctness of the proposed scheme by proving the serializability of the resulting transaction histories.

In the future, we plan to evaluate the performance of the proposed method in terms of the throughput offered at higher security levels as well as the response time. We also plan to measure the additional protocol overhead imposed by the proposed system, especially the cost of maintaining additional versions of data, the list of active transactions, and the cost of active transaction enquiry and reread of low-level data. The costs and benefits will be compared with those of a traditional scheme where a high-level transaction is simply aborted when its low-read operation conflicts with a low-level write operation. In addition, we plan to compare the performance of our scheme with other schemes [13,5] that have similar objectives.

Acknowledgments

The work of V. Atluri was supported in part by the National Science Foundation under grant IRI-9624222. The work of R. Mukkamala was supported in part by the Commonwealth Information Security Center (CISC) under grant V01-198-02. We would like to acknowledge Binto George for taking part in the initial discussions on this work, while he was at Rutgers.

References

[1] Paul Ammann and Sushil Jajodia. A timestamp ordering algorithm for secure, single-version, multi-level databases. In Carl Landwehr and Sushil Jajodia, editors, *Database Security, II: Status and Prospects*, pages 23–25. North Holland, 1992.

[2] Paul Ammann, Sushil Jajodia, and Indrakshi Ray. A Semantic-Based Transaction Processing Model for Multilevel Transactions. *Journal of Computer Security*, 6(3), 1998.

[3] C. Beeri, P. Bernstein, and N. Goodman. A Model for Concurrency in Nested Transactions Systems. *Journal of ACM*, 36(2), 1989.

[4] Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, Reading, MA, 1987.

[5] Elisa Bertino, Sushil Jajodia, Luigi Mancini, and Indrajit Ray. Advanced transaction processing in multilevel secure file stores. *IEEE Transactions on Knowledge and Data Engineering*, 10(1):120–135, 1998.

[6] Panos K. Chrysanthis and Krithi Ramamritham. ACTA, A framework for specifying and reasoning about transaction structure and behavior. In *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, pages 194–203, June 1990.

[7] Ahmed K. Elmagarmid. *Database Transaction Models for Advanced Applications.* Morgan Kaufmann, San Mateo, California, 1992.

[8] Sushil Jajodia and Vijayalakshmi Atluri. Alternative Correctness Criteria for Concurrent Execution of Transactions in Multilevel Secure Databases. In *Proc. IEEE Symposium on Security and Privacy*, pages 216–224, Oakland, California, May 1992.

[9] T. F. Keefe and W. T. Tsai. Multiversion Concurrency Control for Multilevel Secure Database Systems. In *Proc. IEEE Symposium on Security and Privacy*, pages 369–383, Oakland, California, May 1990.

[10] William T. Maimone and Ira B. Greenberg. Single-level Multiversion Schedulers for Multilevel Secure Database Systems. In *Proc. 6th Annual Computer Security Applications Conf.*, pages 137–147, Tucson, Arizona, December 1990.

[11] John McDermott and Sushil Jajodia. Orange Locking: Channel-free Database Concurrency Control via Locking. In *Proc. of the 6th IFIP WG 11.3 Workshop on Database Security*, Vancouver, BC, August 1992.

[12] I.S. Moskowitz and M.H. kang. Covert channels—Here to stay? In *Proc. ninth Ann. Conf. Safety, Reliability, Fault Tolerance, Concurrency, and Real Time Security (COMPASS'94)*, pages 235–243, Gaithersburg, MD, IEEE Press, IEEE Cat. 94CH3415-7, ISBN 0-7803-1855-2, June 1994.

[13] Indrajit Ray, Luigi V. Mancini, Sushil Jajodia, and Elisa Bertino. Asep: A secure and flexible commit protocol for mls distributed database systems. *IEEE Transactions on Knowledge and Data Engineering*, 12(6):880–899, 2000.

[14] Dennis Shasha, Eric Simon, and Patrick Valduriez. Simple Relational Guidance for Chopping Up Transactions. In *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, pages 298–307, San Diego, CA, June 1992.

[15] J. Wray. An analysis of covert timing channels. In *Proc. IEEE Symposium on Security and Privacy*, pages 2–7, Oakland, CA, 1991.

A The Multiversion Serializability Theory

We present the concepts of the multiversion serializability [4] in this section. We use the same notation as in [8]. In a multiversion system, each data item x has a sequence of *versions*. Whenever a transaction T_i writes a data item x, it creates a new version x_i. Each read operation issued by T_i is translated into a read operation on a specific version of that element. Formally, there is a *translation* function h such that $h(w_i[x]) = w_i[x_i]$ and $h(r_i[x]) = r_i[x_j]$ for some j.

Definition 7 Two operations $o_i[x_k]$ and $o_j[x_k]$ *conflict* with each other if they operate on the same version x_k of a data item and at least one of them is a write. □

Definition 8 Two transactions T_i and T_j *conflict* if they contain two conflicting operations $o_i[x]$ and $o_j[x]$, respectively. □

Definition 9 A *multiversion (transaction) history* H over $T = \{T_0, \ldots, T_n\}$ is a partial order with ordering relation $<$ where

1. $H = h(\cup_{i=0}^{n} T_i)$ for some translation function h,
2. For each T_i and all operations $p_i, q_i \in T_i$, if $p_i <_i q_i$, then $h(p_i) < h(q_i)$, and
3. If $h(r_j[x]) = r_j[x_i]$, then $w_i[x_i] < r_j[x_i]$. □

Definition 10 A multiversion history H over T is *serial* if H is a totally ordered multiversion history such that for any pair of transactions T_i and T_j in T, either all of T_i's operations precede all of T_j's or vice versa. □

Definition 11 Given a multiversion history H over T, we say the transaction $T_i \in T$ *reads-x-from* the transaction $T_j \in T$ if $r_i[x_j] \in H$. □

Definition 12 A serial multiversion history H over T is *one-copy serial* if whenever T_i reads-x-from T_j, either $i = j$ or T_j is the last transaction preceding T_i that writes into any version of x. □

Definition 13 A multiversion history H over T is *one-copy serializable* if there exists a one-copy serial multiversion history H' over T such that H and H' are equivalent (i.e., both H and H' contain same operations). □

To test for one-copy serializability of a multiversion history, we will make use of a *multiversion serialization graph*, defined as follows.

Definition 14 Given a multiversion history H over T and a data item x, a *version order* for x is a total order for all versions of x in H. We denote the version order by $\ll$ and write $x_i \ll x_j$ if the version x_i precedes x_j in the version order. □

Definition 15 Suppose that H is a multiversion history over T and that there is some version order $\ll$ for each item x. A *multiversion serialization graph* $MVSG(H)$ for a multiversion history H over T is a directed graph such that

- Nodes of $MVSG(H)$ are transactions in T.
- There is a (directed) edge $T_i \rightarrow T_j, i \neq j$, in $MVSG(H)$ whenever $r_j[x_i] \in H$.
- $MVSG(H)$ also contains *version order edges*: For each $r_k[x_j]$ and $w_i[x_i]$ in H, there is an edge $T_i \rightarrow T_j$ if $x_i \ll x_j$; otherwise, there is an edge $T_k \rightarrow T_i$. □

Theorem 3 [4] Let H be a multiversion history over T. If H has an acyclic multiversion serialization graph $MVSG(H)$ (with respect to some version order $\ll$ for each x), then H is one-copy serializable. □

to this, code can be appended to objects in the form of a Trusted Extension (TE). This module is a form of a portable security policy specific to the object it is embedded in. In order for the TE of an object to function correctly, a Trusted Local Extension (TLE) is included at each site in the federated database, it provides any support required by the TE.

C MSPO Architecture

We define an MSPO as an SPO that has been transferred into a mobile environment and subsequently removed from the federated database. Before an SPO can be made mobile, the site requesting the transfer must be authorised to do so by the TLE of the site where the SPO originated from. The TLE of the original site will receive a request to make the SPO mobile. The request is much like that of a normal transfer request, in that an SPO is being moved from site A to site C. It differs only in that site C belongs to its parent site B, and site C will eventually disconnect itself from its parent site and therefore from the federated database. Site C is therefore a mobile site within the member site B.

C.1 Mobile Sites

Olivier [5] proposes a model for mobiles nodes (sites) in a federated database. This model has each site hosting a mobile site that implements a modified TCC, referred to as an HTCC (Host TCC). Each mobile site then implements a Mobile Trusted Core, which essentially mirrors the HTCC. SPOs moved from a federated site to a mobile site are modified prior to their relocation. In being modified, the object will not include any methods or variables that the mobile site will not be authorised to use. This implies that once an SPO has been relocated to the mobile site, no authorisation will be necessary to use it.

We propose an alternative approach to the mobile architecture; we assume that an SPO will not be used at all if it is not accessed through the TCC. One can argue that modifying an SPO does not necessarily address the problem of security in our model, since although an SPO will be secure once modified and copied to a mobile site. This is not the case when relocating an SPO to a normal site in the federated database, since SPOs relocated to these sites will not be modified.

A mobile site in our proposed model is much like that of a site which is a member of a federated database. The mobile site is slightly different in that it is a child site to a parent site which in turn is an official member site of the federated database. Some degree of authentication must exist between the mobile child site and its parent site, thus minimising the risk of an intruder masquerading as the mobile child site.

A mobile site must implement and trust the TCC, therefore as is the case with member sites of the federated database, a mobile site can only use an SPO through the TCC. This level of trust is essential. An SPO can not be

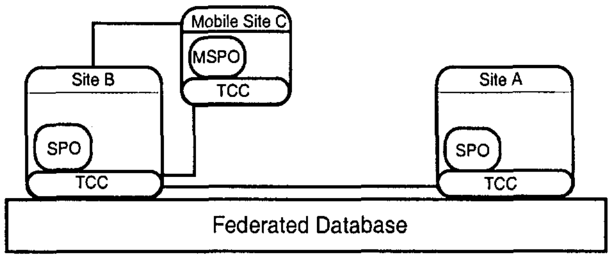


Figure 1: The mobile site model

moved to a site where the TCC is not trusted or is not implemented completely. In our proposed model, a mobile site can only have one parent site and can only connect to the federated database via this parent site. Figure 1 illustrates the mobile site architecture.

C.2 Relocation

Relocating an SPO to a mobile site is similar to the transfer process of an SPO. Since a mobile site is not a complete member of the federated database, it will have its parent site make the request on it's behalf. As is the case with transfer requests of an SPO in a federated database, the request to move an SPO to a mobile site must be authorised. When moving an SPO to a mobile site, authorisation must be gained from the SPO's originating site.

A request to move an SPO from site A to site C, assuming site C is a mobile site belonging to site B, would have the TCC include this information in its request to the TLE of the SPO's home site. Upon authorisation, the TLE will then perform several tasks which are essential to maintaining the integrity of the soon to be MSPO:

- Keep record that the SPO has been moved to the mobile site C, a child site of the member site B.
- Create a duplicate of the SPO and save it locally.
- Create and store hashes of all the values in the SPO that require authorisation to be modified.
- Create and store a hash of the entire SPO.

If the mobile site that an SPO has been relocated to is still active, in that it has not yet been disconnected from the federated database, the SPO can still be used normally, since the TCC of the mobile site is similar to a TCC of a participating site in the federated database.

C.3 Removal from the federated database

The fact that a site is mobile means that it will eventually become unavailable, in that it will disconnect itself from the federated database. Before a mobile site can disconnect from the federated database (its parent site), it must request authorisation from its parent site. If an MSPO that the mobile site is hosting is currently in use, the request to disconnect the site will be denied. Once the request has been authorised, the mobile site is then free to disconnect. The parent site will then inform the originating sites of the MSPOs hosted by the mobile site that the MSPOs have been disconnected from the federated database.

Having been disconnected from the database, an SPO is no longer available for modification in the federated database until it has been reintroduced into the system. However, if necessary, it is possible for sites to retrieve data from the duplicated SPOs as long as the sites retrieving the data understand that the SPOs are currently active elsewhere, hence, the data may not be valid. We do not analyse the concurrency issues involved when an SPO becomes mobile and is disconnected from the federated database; this is not within the scope of this paper and will be covered in future research.

Upon disconnection of a mobile site from the federated database, a timer on the parent site will be associated with the length of time of that the mobile site has been disconnected. If the timer exceeds a predefined threshold, the mobile site and the MSPOs it was hosting will be considered lost and the duplicate SPOs will be introduced into the federated database from the SPOs respective original sites.

This rollback policy will ensure that any MSPO that has been lost while disconnected from the federated database will be reintroduced into the database in its most recent consistent state.

Since a mobile site may disconnect without prior warning, and therefore without authorisation, a site hosting the mobile site must be able to handle such situations effectively. The primary reason for a mobile site to obtain authorisation before disconnecting is to accommodate any transactions that may be running on any of the MSPOs it is currently hosting. The worst case scenario in such a situation may be that a transaction does not run to completion. This transaction would have to be aborted, the parent site would then begin the process of handling the mobile site disconnection, as described earlier.

D Maintaining Integrity of an MSPO

The proposed MSPO architecture of the previous section gives us a framework in which to now examine and discuss methods as to how the integrity of an MSPO can be maintained. We believe that there are three major issues, that when addressed and implemented in a proper manner, will ensure integrity within the MSPO architecture. The issues we have identified are the following:

- The proper management of transactions in a mobile environment that require authorisation.
- An orderly and secure manner in which to update MSPOs.
- A process of re-entry for an MSPO into the federated database.

We begin this discussion by looking at general functionality of an MSPO and move onto an analysis of how transactions which require authorisation are dealt with whilst the mobile site is disconnected. We will then discuss how an MSPO can be updated and examine the process involved when an MSPO is to re-enter the federated database.

D.1 MSPO functionality outside of the federated database

Since the site an MSPO is hosted on is mobile, upon disconnection the TCC associated with that site may have no means of making contact with any other sites of the federated database. This has an impact particularly on transactions which require some kind of interaction with the TLE of an MSPO's originating site. Since the TCC may have no way of establishing contact with any member sites of the federated database, transactions which require authorisation or some kind of interaction with the MSPO's originating site may be denied execution by the TCC.

In light of this problem, before a transaction will be allowed to execute within a mobile site, the TCC will first analyse the nature of the transaction. From this, the TCC will determine what (if any) authorisation requirements there are for the transaction and whether or not the authorisation may be attained.

It could be the case that a mobile site may be able to establish contact with its parent site via a cellular link or remote network, thereby allowing the TCC to communicate with other sites of the federated database and attain proper authorisation for transactions. On the other hand, it may be the case that the mobile site may not be able to establish contact with its parent site, any transactions requiring authorisation will therefore be denied permission to execute.

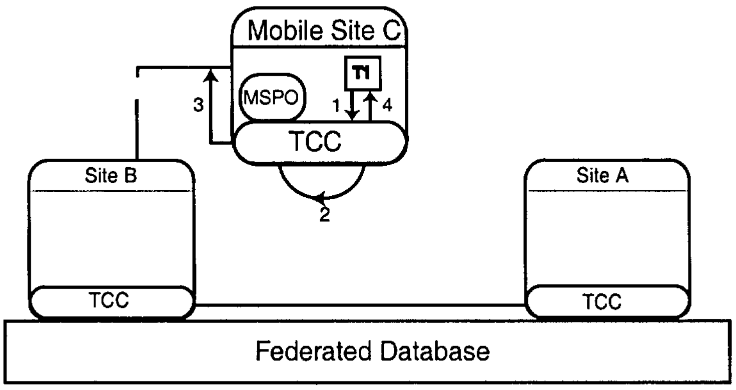
Figure 2 illustrates the flow of operations as a transaction which requires authorisation from a site on the federated database attempts to execute.

Since an MSPO can only be used within the framework of a trusted TCC on a mobile site, and since transactions which require authorisation will only be executed provided that the TCC of the mobile site can establish contact with its parent site, we can be assured that the integrity of the MSPO will not be maliciously compromised whilst it is disconnected from the federated database.

The integrity of each MSPO is still subject to several forms of compromise the likes of errors, viruses or failures in the system, see [2]. Section D.4 discusses how such compromises are detected and what action is taken upon detection.

D.2 Executing transactions which require authorisation

If the TCC of a mobile site is able to establish contact with its parent site, it can begin the process of obtaining authorisation for the transaction to be executed within the mobile



1. Request to execute transaction 'T1'.
2. The TCC analyses the nature of transaction T1 and determines authorisation from site A is required.
3. The TCC attempts to establish a connection to its parent, site B.
4. Since the TCC can not establish a connection to site B, transaction T1 is denied permission to execute.

Figure 2: An illustration of the steps taken by a mobile site to establish a connection for a transaction requiring authorisation.

site on an MSPO. Having obtained authorisation the transaction will execute and run to completion. Upon completion of the transaction, steps similar to those performed on an SPO will be taken in order to maintain the integrity of the MSPO: A duplicate of the MSPO will be made and sent to its originating site where a hash of the MSPO as well of each value that requires authorisation to be modified will be created and stored.

As an example of executing transactions on mobile sites which require authorisation, consider a car salesman who has moved an SPO of type Car onto his laptop. He is currently at a client's premises and the client has requested details on the lifetime of the gear box. The salesman queries the gear box lifetime property of the Car MSPO and is subsequently requested to connect to the federated database. The gear box lifetime property is a property which requires authorisation in order to be read. The TCC on the salesman's laptop observed the nature of the salesman's query and identified the need for authorisation, hence, the salesman has been prompted to connect to the federated database.

Should the salesman not connect to the database, the TCC will be unable to obtain authorisation and the query to view the gear box lifetime property will be denied. If the salesman does connect the laptop to the federated database, he may be prompted for a username and password, upon successful authorisation the TCC will grant permission to run the query and hence view the gear box lifetime property.

We have so far assumed that should a mobile site be unable to establish a connection to its parent site, any transactions that require authorisation will simply be denied permission to execute and subsequently deleted. In the following section we propose an alternative to handling transactions so as to acommodate the execution of transactions which require authorisation even though the mobile site may not be able to connect to the federated database.

D.3 An alternative to executing transactions which require authorisation

There are several problems in allowing an MSPO to be updated whilst disconnected from the federated database. We are concerned with the problem of integrity. Since an MSPO is not connected to the federated database, how can we be certain that an MSPO which has been updated whilst disconnected from the federated database has undergone a series of authorised modifications? One could argue that we can be sure the MSPO has maintained its integrity since it will only be used via the TCC.

Within the framework we have proposed so far, this is indeed the case, but usage of an MSPO limited. There will be properties of an MSPO which require authorisation in order to be modified. Since the MSPO is on a mobile site, connecting to the federated database to obtain authorisation may not always be feasible. Does this mean that one is denied the ability to update an MSPO property which requires authorisation, unless the mobile site can establish contact with the federated database? This is indeed the case if we are to be certain that integrity will be maintained.

An alternative to this approach may be to make use of a slightly modified TCC on the mobile site. Modifications can be made to the way a TCC handles transactions in general. The TCC could allow transactions which require authorisation to simply be executed, whilst maintaining a log of all the activities of each transaction. Upon re-entry into the federated database (discussed in the following section), the transaction log for the MSPO would be analysed and executed on the duplicate copy made of the MSPO before becoming mobile. If unauthorised modifications are identified, these transactions can simply be ignored. The duplicate copy of the MSPO, having undergone all the authorised transactions of the transaction log would then be permitted to re-enter the federated database. Figure 3 illustrates the proposed alternative to handling unauthorised transactions.

This approach to dealing with authorised transactions opens areas of future research regarding prioritising SPOs or SPO classification. If we once again consider a salesman who has moved a *Car* SPO to his laptop. The salesman visits a client who is going to purchase a car based on the retail value of the car. The salesman does not have access to modify the retail price property of the *Car* MSPO but since his mobile site is using the approach described above, he is able to reduce the cost of the car considerably. He visits the client and the client is so impressed with the car's retail price that he decides to purchase the car.

The salesman now updates the *Car* MSPO as being sold. Usage of the MSPO in this manner is authorised. The salesman returns from the client and the laptop is reconnected to the federated database. Upon parsing the transaction log, the TCC handles the two transactions above in the following manner:

- Salesman reduces *Car.retail_price* by 10,000: Unauthorised Transaction - **NOT EXECUTED**

- Salesman sells *Car*: Authorised Transaction - **EXECUTED**

Although the retail price of the car was not reduced, the sale is still made since it was an authorised transaction. The implications of this kind of transaction management are obvious. Future research could therefore entail classifying SPOs into categories which are too sensitive to be relocated onto sites that employ this type of transaction management.

D.4 Re-entry into the federated database

Before an MSPO can be allowed back into the federated database, one has to be certain it is an MSPO which has undergone a series of authorised transactions and has, essentially, maintained its integrity.

Assuming the time that the mobile site has been disconnected has not exceeded the maximum time that it is allowed to be disconnected, each MSPO that is hosted by the mobile site will in turn undergo a re-entry process which will determine whether or not the MSPO will be allowed to re-enter the federated database.

The primary goal of the re-entry process is to ensure that the MSPO being re-introduced into the database has not had its integrity compromised and at the very least, is the MSPO that it claims to be. As the mobile site reconnects to its parent site, the MSPO will be moved from the mobile site to its originating site. The Unique Identifier (UI) of the SPO will then be extracted and a hash made of all the values that require authorisation in order to me modified. The UI and the hash will then be compared to the duplicate UI and hash stored at the SPO's originating site before the SPO became mobile or after any authorised modifications made whilst the MSPO was mobile. If they are alike, the SPO will then undergo the second and final phase of the re-entry process.

Should the SPO fail the first phase of the re-entry process, the SPO will be denied re-entry into the database. Having failed the first phase, one can with certainty deduce that the integrity of the SPO has indeed been compromised. The SPO will be deleted and the duplicate copy made at its originating site will be used to re-introduce the SPO into the federated database, in its last uncompromised state.

The second phase of re-entry has the SPO undergo an order of operations which will ensure that any valid changes made to the SPO whilst mobile will now be saved and the SPO will be declared as being in a valid, uncompromised state. The new duplicate copy and hashes will simply replace the previous duplicate copy and hashes stored on the SPO's originating site.

E Conclusion

In this paper we have introduced the concept of Mobile Self Protecting Objects and have proposed an architecture within which it can be implemented. We have discussed how MSPOs can be used on mobile sites and how transactions requiring authorisation can be handled with the ultimate goal of ensuring that the integrity of the MSPO will always be maintained.

We have assumed throughout this paper that mobile sites will implement and trust the TCC. In doing so, we can be certain that the integrity of an MSPO will be maintained regardless of malicious intent or error.

References

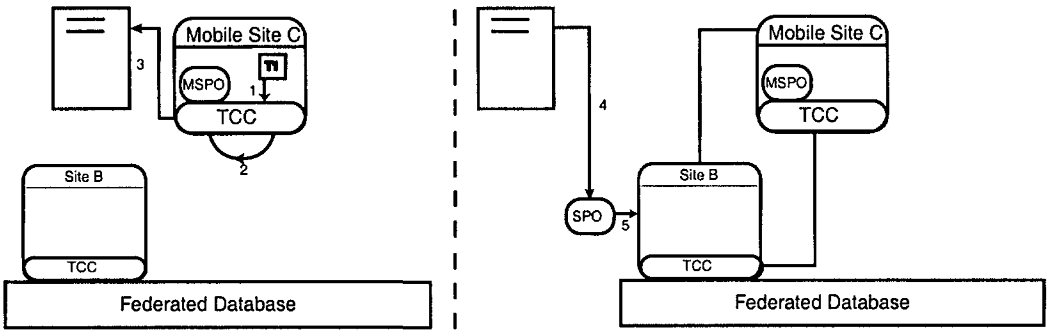
[1] B. R. Badrinath and Shirish Hemant Phatak. An Architecture for Mobile Databases. Rutgers University, Department of Computer Science, Technical report #DCS-TR-351, 1995

[2] S. Castano, M. Fugini, G. Martella and P. Samarati. Database Security. Addison-Wesley & ACM Press, 1995

[3] S. Ceri and G. Pelagatti. Distributed databases — Principles and Systems. McGraw-Hill, 1985

[4] M. S. Olivier. Self-Protecting Objects in a Secure Federated Database. In Spooner, SA Demurjian and JE Dobson (eds), *Database Security IX: Status and Prospects*, 27–42, Chapman & Hall, 1996

[5] M. S. Olivier, Secure Mobile Nodes in Federated Databasesbase, South African Computer Journal, 20, 11–17, 1997



1. Request to execute transaction 'T1'.
2. Authorisation to execute the transaction is automatically granted.
3. The TCC logs the transaction.
4. Upon reconnection, the logged transactions are executed on the duplicate SPO. All unauthorised transactions are ignored.
5. If this task is completed successfully, the SPO is granted access into the federated database.

Figure 3: An alternative to handling unauthorised transactions.

[6] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, **22**, 3, 183–236, 1990

[7] K. Tipton. Fortune 1000 Executives Forecast Dramatic Increase in Need for Mobile Communications. AVT Corporation News Release, 2000, Available online at http://www.ati-cti.com/ATIArticles/AVTArticle1.html

Combining World Wide Web and wireless security

Joris Claessens, Bart Preneel and Joos Vandewalle
COmputer Security and Industrial Cryptography (COSIC)
Dept. of Electrical Engineering – ESAT, Katholieke Universiteit Leuven, Belgium
http://www.esat.kuleuven.ac.be/cosic/
joris.claessens@esat.kuleuven.ac.be

Keywords: WWW security, wireless security, m-commerce

Received: February 3, 2002

In current electronic commerce systems, customers have an on-line interaction with merchants via a browser on their personal computer. Also payment is done electronically via the Internet, mostly with a credit card. In parallel to this, e-services via wireless-only systems are emerging. This paper identifies security and functionality weaknesses in both of these current approaches. The paper discusses why and how general-purpose mobile devices could be used as an extension to PC based systems, to provide more security and functionality. General-purpose mobile devices are shown to be an alternative to costly special-purpose hardware. This combined approach has in many cases more interesting properties than when using mobile devices only. As an example of the combined approach, a GSM based electronic payment system is proposed and investigated. The system enables users to order goods through the World Wide Web and pay by using their mobile phone.

A Introduction

In current electronic commerce systems, customers have an on-line interaction with merchants via a browser on their personal computer. Also payment is done electronically via the Internet, mostly by sending a credit card number to the merchant. This basic system is in widespread use today, and most people are familiar with buying books and music, booking flights, ordering PCs, etc. There are however some important security problems. For example, credit card numbers are often stolen by hackers from merchants' computers, orders and confirmations are usually not digitally signed and can be repudiated afterwards. In parallel to the fixed PC based systems, e-services are also emerging in the wireless world. Current mobile devices have however rather limited functionality, and in many applications, they are not suited to be used on their own.

This paper suggests a combined approach in which mobile devices are used as an extension to the World Wide Web environment. The paper starts with a description of the security properties of the World Wide Web in Sect. 2, and the security features in some wireless systems, i.e., GSM and WAP, in Sect. 3. Section 4 discusses security and functionality weaknesses in both worlds, and suggests a combined approach. An example of this approach is given in Sect. 5: a GSM based electronic payment system for the WWW is proposed and investigated. Further analysis of this system is presented in Sect. 6.

B World Wide Web security

There are many security issues related to the WWW. Within the scope of this paper, we will only discuss the communications security aspect, both at the network and the application level, and the payment security aspect.

B.1 Communications security

The communication between a web browser and a web server is secured by the SSL/TLS protocol. Historically, Secure Sockets Layer (SSL) was an initiative of Netscape Communications. SSL 2.0 contains a number of security flaws which are solved in SSL 3.0. SSL 3.0 was adopted by the IETF Transport Layer Security (TLS) working group, which made some small improvements and published the TLS 1.0 [9] standard. "SSL/TLS" is used in this paper, as "SSL" is an acronym everyone is quite familiar with; however, the use of TLS in applications is certainly preferred to the use of the SSL protocols.

Within the protocol stack, SSL/TLS is situated underneath the application layer. It can in principle be used to secure the communication of any application, and not only between a web browser and server. SSL/TLS provides entity authentication, data authentication, and data confidentiality. In short, SSL/TLS works as follows: public-key cryptography is used to authenticate the participating entities, and to establish cryptographic keys; symmetric key cryptography is used for encrypting the communication and adding Message Authentication Codes (MACs), to provide data confidentiality and data authentication respectively. Thus, SSL/TLS depends on a Public Key Infrastructure. Participating entities (usually only the server) should have a public/private key pair and a certificate. Root certificates (the certification authorities' certificates that are needed to verify the entities' certificates) should be securely distributed in advance (e.g., they are shipped with the browsers). Private keys should be properly protected. Note that these two elements, i.e., distribution of root certificates in browsers and the protection of private keys, is actually one of the weak and exploited points with respect to WWW security (see 4.1).

More detailed information on SSL/TLS, the security flaws in SSL 2.0, and the differences between SSL 3.0 and TLS 1.0, can be found in Rescorla [30].

B.2 Application security

SSL/TLS only protects data while it is in transit. Moreover, exchanged messages are not digitally signed. Therefore it does not provide non-repudiation. Both customers and merchants can always deny later on having sent or received requests or confirmations from each other.

In addition to SSL/TLS, critical messages should thus be digitally signed before they are sent through the secure channel. The concept of digitally signing messages is not really integrated yet in today's web browsers. Netscape though allows the content of forms to be digitally signed using the Javascript `signText()` function. XML will be more and more used on the WWW to represent content instead of the basic HTML. In the future, browsers are therefore expected to implement Signed XML [11], which specifies how XML documents should be digitally signed.

Note that an alternative protocol to secure the communication on the WWW has been proposed in the past: S-HTTP [31]. This protocol is situated at the application layer, and is specifically intended for HTTP. It secures HTTP messages in a very similar way to the protocols for secure email, and provides non-repudiation. SSL/TLS has however become the de-facto standard on the web, and S-HTTP was not a success.

B.3 Payment security

Although numerous different electronic payment systems have been proposed that can be or are used on the WWW, including micro-payment systems and cash-like systems, most transactions on the web are paid using credit cards. Mostly, customers just have to send their credit card number to the merchant's web server. This is normally done 'securely' over SSL/TLS, but some serious problems can still be identified. Users have to disclose their credit card number to each merchant. This is quite contradictory to the fact that the credit card number is actually the secret on which the whole payment system is based (note that there is no electronic equivalent of the additional security mechanisms present in real world credit card transactions, such as face-to-face interaction, physical cards and handwritten signatures). Even if the merchant is trusted and honest this is risky, as one can obtain huge lists of credit card numbers by hacking into (trustworthy, but less protected) merchants' web servers. Moreover, it is possible to generate fake but valid credit card numbers, which is of great concern for the on-line merchants. Thus, merchants bear risk in card-not-present transactions.

Secure Electronic Transaction, SET [33], is a more advanced standard for credit card based payments. One of its core features is that merchants only see encrypted credit card numbers, which can only be decrypted by the issuers. Moreover, the number is cryptographically bound to the transaction by a digital signature. This system is conceptually much better, but until now it has not become popular due to its complexity.

Recently, Visa published the specifications of its 3-D Secure Authenticated Payment Program [37]. This system is mainly based on SSL/TLS. Its purpose is to authenticate cardholders in order to reduce the number of disputed online transactions.

American Express offers a 'one-time credit card' solution [1] with which customers can protect their privacy, but which also solves some of the above mentioned problems. Alternatively, several similar systems exist (e.g., InternetCash [18]) in which customers can obtain some pre-paid value identified and protected with a number and PIN, and use it on-line in cooperation with a central server. Finally, real-life electronic payment means (e.g., Proton [29] and debit cards) are also starting to be deployed on the WWW (e.g., [2]).

C Wireless security

GSM and WAP are currently probably the two most popular and widely used wireless technologies. They are briefly presented in the following paragraphs. Thereafter, some other systems and initiatives in the wireless world are discussed.

C.1 GSM

GSM, Global System for Mobile communications, is the currently very popular digital cellular telecommunications system specified by the European Telecommunications Standards Institute (ETSI). In short, GSM intends to provide three security services [36]: temporary identities, for the confidentiality of the user identity; entity authentication, that is, to verify the identity of the user; and encryption, for the confidentiality of user-related data (note that data can be contained in a traffic channel, e.g., voice, or signaling channel, e.g., SMS messages).

The Subscriber Identity Module (SIM) is a security device, a smart card which contains all the necessary information and algorithms to authenticate the subscriber to the network. It is a removable module and may be used in any mobile equipment [36]. Note that the encryption algorithms are integrated into the mobile equipment as dedicated hardware. GSM does not use public-key cryptography. Symmetric keys are derived from user related data using an algorithm under the control of a master key.

The electronic payment system described in the example later in this paper, requires the SIM to contain a small payment application, based on the SIM Application Toolkit. The *SIM Application Toolkit* [14] provides mechanisms which allow applications, existing in the SIM, to interact and operate with any compliant mobile equipment. These mechanisms include displaying text from the SIM to the mobile phone, sending and receiving SMS messages, and initiating a dialogue with the user. In addition to the GSM security mechanisms, special SIM Application Toolkit security features have been defined [12, 13]. The security requirements that have been considered are: (entity) authentication, message integrity, replay detection and sequence integrity, proof of receipt and proof of execution, message confidentiality, and indication of the security mechanisms used. According to the standard, digital signatures can be used to implement some of these requirements.

Note that the same distinction between communications security and application security as made in the WWW security context, can be made here: standard GSM security at the communications level, and SIM Application Toolkit security at the application level.

C.2 WAP

The Wireless Application Protocol (WAP) is a protocol stack for wireless communication networks. WAP is bearer independent; the most common bearer is currently GSM.

Similar to SSL/TLS for the Internet, WTLS [44] is WAP's communications security solution. It also relies on a Public Key Infrastructure [40, 39]. The main differences are that WTLS supports by default algorithms based on elliptic-curve cryptography, is adapted for datagram communication (instead of connection), and supports its own certificate format, besides X.509v3, optimized for size. TLS was as such modified to make it more suitable in an environment where there are bandwidth, memory, and processing limitations.

At the application layer, WAP provides digital signature functionality through the WMLScript Crypto Library [45], which is similar to Netscape's Javascript signing. Comparable to the GSM's SIM, WAP devices will use a Wireless Identity Module (WIM) [43] which can contain the necessary private and public keys to perform digital signatures and certificate verification respectively.

C.3 Other systems and inititiatives

GSM is a second-generation system (2G). UMTS, Universal Mobile Telecommunications System [35], is part of a global family of third-generation (3G) mobile communications systems. These systems provide high-capacity and more secure [38] communication. A competitor of WAP is NTT DoCoMo's i-mode [27]. Bluetooth [5] is a wireless protocol for communication between devices that are in close proximity. The Internet itself is also expanding to the wireless world. The IETF is currently defining standards for Mobile IP [17], and is working on extensions (including wireless) for TLS [4].

The Mobile Electronic Signature Consortium has defined mSign [24], which should provide a standardized interface between Primary Service Providers (e.g., merchants) and Mobile Operators. It allows Primary Service Providers to request signatures from end-users through the Mobile Operators. The Mobile electronic Transactions initiative – MeT [25] – intends to establish a consistent and coherent framework for secure mobile transactions, based on existing standards and specifications; where needed, new functionality will be submitted to relevant standardization and specification organizations. There are numerous other fora concerned with mobile secure payments, see [7] for a description and comparison of these.

D Combining WWW and wireless

Both the World Wide Web and the wireless world on their own have security and/or functionality problems. These shortcomings are explained in the following paragraphs. An approach in which the two worlds and their advantages are combined, is then motivated.

D.1 WWW: problems

It is very common that only web servers have certificates with which they are authenticated. In case user authentication is needed, it is almost never done via SSL/TLS client authentication. Users are often authenticated via their IP address, which is vulnerable to IP spoofing [3], which certainly does not provide mobility, and which is just not usable in an open system. Fixed passwords are frequently used, which provide mobility, but which are vulnerable to guessing, dictionary attacks and social engineering [26]. Passwords that are only used once [21] are not frequently used. They would be more secure, but certainly less convenient.

Root certificates are needed when verifying a web server certificate. It is very important that a user has an authentic copy of these certificates. This is more or less ensured by shipping them together with the browsers. It is however easy to add more or even replace root certificates. Moreover, the browser trust model causes a server certificate to be trusted if it is successfully verified by any of the root certificates (since there is usually no central policy management, this might easily include an attacker's root certificate). Finally, browsers generally also do not yet check by default if a certificate has been revoked.

Users must recognize when they have a secure session with a web server. However, in today's browsers, there are only some limited visual indications (e.g., closed lock), and an unexperienced user is easily fooled by a spoofed web site as demonstrated by Felten *et al.* [15] and more recently by Yuan *et al.* [46].

If the user has a public/private key pair – for SSL/TLS client authentication, for SET, or for digitally signing documents – the private key will mostly reside on the hard disk of the machine. Even if it is protected by a pass phrase, it is still very vulnerable, for example due to Trojan horses. Users with such a software token are also hardly mobile. Smart cards are a solution, but for particular applications, they might be inconvenient. Moreover, smart card readers are currently not installed on each machine. Other special-purpose hardware, such as a Digipass [10], as sometimes used in e-banking, might be too costly for small applications, i.e., the investment for the customers and/or merchants would just be too high compared to the expected benefits.

Current end-user computing systems tend to offer more functionality at the cost of security. This is actually the reason why for example root certificates and private keys are so vulnerable on current end-user machines. Specifically, there is currently a lack of secure operating systems [22] and trusted components [34]. Today's PC and browser offer advanced functionality, but are (therefore) an insecure environment.

D.2 Wireless: problems

While the security problems on the WWW are currently more related to the secure management of the end-points, the security problems in some wireless systems are still with the protocols and algorithms themselves. For example, algorithms used by many GSM providers have been broken and 'over-the-air cloning' and real-time eavesdropping have been shown (at least in theory) to be feasible [32]. Security problems have been discovered in other mobile systems too [6, 19]. Most of these problems are due to non-public design of the algorithms and protocols, leakage and/or publication of the details to the general public afterwards, and discovery of flaws by the cryptographic community.

More conceptually, both GSM and WAP do not offer end-to-end security. GSM security only applies on the wireless link, i.e., from mobile phone to base station, but not from mobile phone to mobile phone. The fixed network is considered to be secure (more precisely, GSM intends to offer the same security level as the fixed network). In the WAP architecture, WAP devices communicate with web servers through a WAP gateway. WTLS is only used between the device and the gateway, while SSL/TLS can be used between the gateway and the server. From a security point of view, this means that the gateway should be considered as a person-in-the-middle. Note that WAP is now evolving into end-to-end security [42, 41].

Security seems to evolve in the good direction though. From a usability point of view on the other hand, mobile devices have still a rather limited functionality. They are not performant, and have often a quite poor human-device interface. Although mobile devices are getting more advanced, they will always be outsmarted by desktop PCs. Note that the complexity of the PC (e.g., multi-user operating system, data with executable content, ...) is the main reason why securing the end-points of the communication is such a difficult task, and remains an important problem on the WWW. As long as mobile devices stay quite simple and do not provide too much functionality, their security as an end-point will be more easy to cope with.

D.3 Motivation for a combined approach

By combining the World Wide Web with a wireless system, we want to come to practical and low-cost electronic commerce applications, which can fully exploit the broad functionality of the WWW. Two goals should hereby be achieved at the same time: *security* and *mobility*.

The WWW on its own does not seem to be sufficient for these applications. It surely provides broad functionality. When for example only fixed passwords are used, the WWW also offers mobility, i.e., a user can initiate transactions from any computer (e.g., a public terminal). Strong security is in that case however not achieved. Stronger security can be achieved by using for example cryptographic keys stored on the computer's hard disk. However, this does not allow for practical mobility. Special-purpose hardware tokens would increase the security of the application and provide mobility again. However, in an electronic commerce environment, consumers do not likely want to pay for a token that can only be used in the context of that application.

Wireless systems on their own are not suitable either. By definition, they offer mobility. Although there are some weaknesses in current systems, security in wireless systems tends to improve substantially. It is however clear that the GSM system is a rather limited environment. WAP offers a more general and WWW-like functionality, but in practice today's devices and networks do not satisfy the needs of merchants and customers. Mobile devices are generally expected to stay inferior to desktop computers.

This brings us to the motivation for a combined approach. Mobile devices are general-purpose devices which can be used as an extension to the WWW – instead of special-purpose devices – to offer more security and mobility without any extra cost. These mobile devices can be personalized and can store secret information such as cryptographic keys. They can be used in combination with any computer, i.e., the personal computer at the user's home, but also a public terminal, hereby providing mobility. Moreover, the computer terminal must not necessarily be completely trusted, as (part of) the security will rely on trusted and/or secret information that is securely stored in the device (and never leaves it, in case of secrecy).

In the remainder of this paper, this combined approach will be illustrated with an electronic payment system for the WWW that makes use of a mobile phone. This GSM based system is an alternative to the widely spread credit card based solution, offering more security and equivalent mobility and complexity (assuming that a mobile phone is standard equipment of many users). In addition, it might be suited for lower-price transactions.

E GSM based payment for the WWW

The main goal of the remaining part of the paper is to present a system in which the WWW and GSM environment are combined to improve overall security, mobility, and functionality. In particular, an architecture and protocol are developed in which: (1) a customer can initiate and complete an electronic *payment* over the GSM network where the network operator is an active participant; (2) the *pre-payment* related interaction is done via the WWW; (3) the customer receives a receipt with which he/she can pick up the goods (*post-payment*).

E.1 Involved entities

The following entities play an active role in this e-commerce system:

Customer The Customer wants to buy something via the WWW. Payment will be done via his/her GSM. The Customer will receive a receipt, with which he/she can pick up the goods (the system must work with both physically deliverable goods and electronically available goods). Obviously, the Customer should have a PC with Internet connection. This can also be a public terminal. He/she needs a mobile phone with SIM Application Toolkit functionality. The SIM card should be issued by a Network Operator that is running this electronic payment service. Optionally, there should be a connection between the mobile phone and the PC, and accordingly some extra software on the PC.

Merchant The Merchant wants to sell something via the WWW. He/she should have a web server, and an access point to the mobile network. Examples are an on-line bookstore, a pizza delivery chain, an electronic parts shop, etc.

Deliverer The Deliverer is the local (with respect to the Customer) representative of the Merchant. It will deliver the goods after having verified the receipt the Customer has obtained from the Merchant. The Deliverer should have some equipment to verify this receipt. An example is the pizza delivery boy/girl, etc. The Deliverer can also be another company that made an agreement with the Merchant. For example, the Merchant can send the goods to a gas station near the Customer; in this case, the gas station is the Deliverer where the Customer can pick up the goods.

Network Operator The N.O. plays the role of the bank. It will deduct the necessary amount of money from the Customer's balance (can be credit or pre-payment based), and add this amount to the Merchant's balance. A commission on this amount will be taken, or a periodical fee will be requested from the Customer and/or Merchant. In practice there will be multiple N.O.s: N.O.(C), N.O.(M) and N.O.(D), for the Customer, the Merchant and the Deliverer respectively (as shown in Fig. 1).

Note that in reality, and from a non-technical point of view, it might not be easy for any Network Operator to deploy an electronic payment service (e.g., banking license). Alternatively, the "Network Operator" could in this system be replaced by a real financial institution, which makes an agreement with one or more operators.

E.2 Architecture and protocol

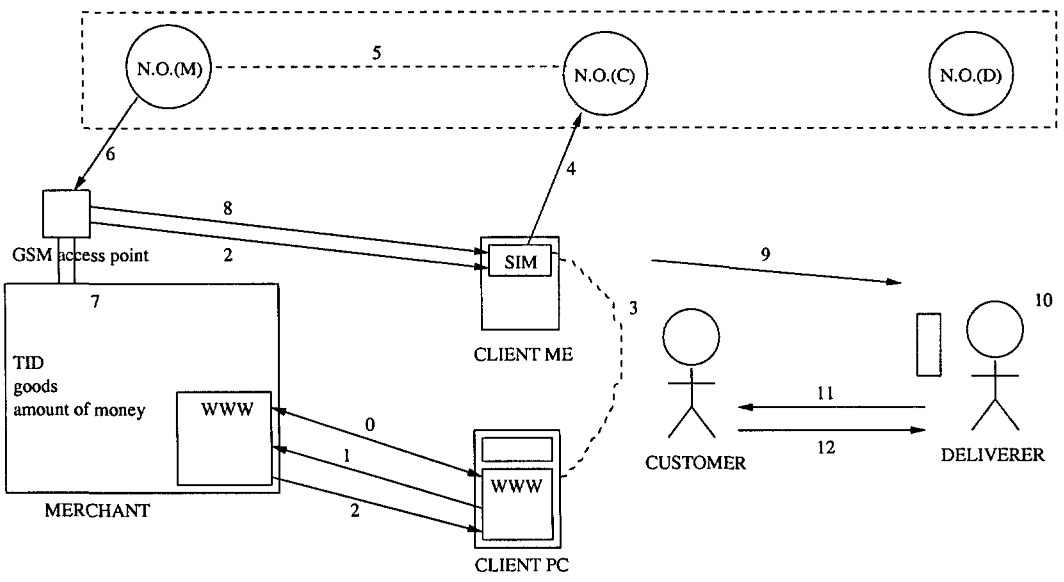
From a high-level point of view, the different entities perform the following interactions (see Fig. 1): after browsing and negotiating, the Customer requests a purchase; via an SMS message, the Merchant asks the Customer to pay the purchase; the Customer pays by sending an SMS message to the Network Operator; the Network Operator informs the Merchant about the successful payment; the Merchant sends a receipt to the Customer (also an SMS message); the Customer can use this receipt to pick up the goods at the Deliverer.

The protocol contains the following steps (see Fig. 1):

1. Purchase Request After browsing and negotiating (0), the Customer makes a *Purchase Request* via the WWW (1). The Merchant can choose the format and encoding of the message. It should at least contain a description of the goods, the amount of money to be paid, and the Customer's GSM number (in order to be able to send an SMS message to the Customer). The message will normally be sent through submission of an HTML form. The level of protection can be chosen by the Merchant, but it will normally be protected in transit by SSL/TLS. The form could also be digitally signed by the Customer (e.g., Netscape's Javascript signing capability, or Signed XML).

2. Purchase Confirm The Merchant sends a *Purchase Confirm* via SMS (2) to the Customer's mobile phone. This message should be in a standard format, and is optionally digitally signed by the Merchant. The message contains: (optionally) a description of the goods (either a hashed form of the description, or an abbreviated yet unique description of the goods, e.g., as in supermarket receipts), a Transaction ID (TID), a unique Merchant ID, the ID of N.O.(M), and the amount of money to be paid. The Merchant also sends a *Purchase Confirm* via the WWW (2). Note that this could already be included in the reply to the submission of the Purchase Request form.

3. Verification by the Customer The Customer verifies whether all the ordered goods are listed, and whether the amount of money requested equals the amount agreed on. The information in the SMS message should be the same



1. Purchase Request	7. Verification by Merchant
2. Purchase Confirm	8. Receipt
3. Verification by Customer	9. Presentation of Receipt
4. Debit Account	10. Verification by Deliverer
5. Inter-N.O.	11. Delivery of Goods
6. Delivery OK	12. Confirmation of Reception

Figure 1: GSM based payment for the WWW: architecture and protocol

as the information displayed in the browser. Authentication of the Merchant thus relies on both GSM (we assume that the Customer knows the number of the Merchant) and SSL/TLS, so the Customer's trust in the correct execution of the transaction increases. If the reply in the browser and/or the SMS message are digitally signed, the signatures are verified. Note that in current GSM phones such a signature must possibly be verified using additional software on the computer. This requires a connection between the mobile phone and the PC which can for example be provided by Bluetooth. An automatic verification and comparison of the reply in the browser and the SMS message can then also be made. The interface to the Customer is provided by the SIM Application Toolkit. A payment application is installed on the SIM card, which is invoked on receipt of a Purchase Confirm message.

4. Debit Account The SIM Application Toolkit application asks the Customer a confirmation for sending a *Debit Account* message (4) to the N.O.(C). This message includes the amount of money to be paid, the TID, the Merchant's ID and N.O.(M)'s ID. The authentication of the Customer relies on GSM entity authentication (the Customer's mobile phone number should be in the Merchant's database). The TID will allow verification by the Merchant afterwards.

5. Inter-N.O The N.O.(C) deducts the proper amount of money from the Customer's balance, and forwards the Debit Account message to N.O.(M). The N.O.(M) adds the amount to the Merchant's account.

6. Delivery OK The N.O.(M) sends a *Delivery OK* (6) to the Merchant. This message contains the amount of money and the TID, and can be digitally signed by the N.O.(M).

7. Verification by the Merchant The Merchant verifies if the *Delivery OK* message originates from the N.O.(M) (relying on GSM entity authentication). If added, the digital signature of the N.O.(M) is verified. The Merchant looks up the TID in his transaction database, and checks if the amount of money is the same as included in the corresponding Purchase Confirm messages.

8. Receipt The Merchant sends a *Receipt* (8) to the Customer via SMS. It contains: a (hashed) description of the goods, the TID, a timestamp (in order for the Deliverer to verify the freshness of the receipt), information on the Deliverer (optionally depending on the Customer's cell location, and including the Deliverer's GSM number), and information on the Customer (optionally including its GSM number, to allow verification of ownership of the receipt). The receipt is digitally signed by the Merchant. The receipt can only be used for the intended Deliverer as indicated. The TID and timestamp ensure that the receipt cannot be replayed by the Customer (i.e., the Deliverer should keep a list of previously received TIDs and should not accept receipts that are too old). GSM authentication is relied upon for authenticating the Customer.

9. Presentation of the receipt If goods are electronic and delivered via the WWW, a receipt is not needed. Goods are then downloaded using the TID. The Merchant keeps a list of which TIDs correspond to transactions for which a payment has been received. Physical goods should be retrieved at the Deliverer. The receipt is forwarded to the Deliverer (9), manually or through the SIM Application Toolkit, or the Customer just presents the receipt to the Deliverer on the screen of his/her own GSM.

10. Verification by the Deliverer The Deliverer just reads the receipt from the screen of the Customer's or his/her own GSM, or he/she verifies the receipt more properly by checking if the signature of the Merchant is valid. The Deliverer needs some infrastructure with GSM access point for this (e.g., a GSM connected to a laptop).

11. Delivery of goods If the receipt is valid, the Deliverer can be sure that the Customer is the one that has made (and paid) the purchase. The goods can thus be delivered (11). In case of electronic goods which are delivered directly by the Merchant's web site (not necessarily though, as the Deliverer might have its own web site), the Customer should be granted access based on the TID: after a Delivery OK message has been received, the Merchant enables the access to the information; the TID should not be known to other entities (however, note that the N.O. should be trusted not to misuse its knowledge of the TID).

12. Confirmation of reception After the Customer has obtained the goods, it can optionally be required that he/she confirms the reception of the goods (12), e.g., by digitally signing a specific message. This will prevent Customers from denying later on having received the goods.

F Analysis and remarks

The proposed GSM based electronic payment system for the WWW is analyzed further in this section. Some GSM specific comments are given, the security and privacy of the system is evaluated, and a comparison with a number of similar systems is made. Note that this section only intends to discuss this particular example, and not the general combined approach.

Only the essential steps of the proposed payment system are presented in this paper. It is clear that a real implementation of this system would require many extra features. For example, it is possible that the Customer completed the payment but did not receive a receipt. Other kinds of interrupted transactions might occur. To be able to cope with this, status and cancel requests should be built into the system.

F.1 GSM functionality

The protocol relies on SMS messages. These can only contain 160 characters, which should be taken into account when defining the exact content of the protocol messages. Note that GSM provides a mechanism to send long messages as a concatenation of multiple SMS messages. Since the protocol involves on-line bi-directional communication between the entities, there should be not much latency between sending and receiving SMS messages. This might be a problem in the case of international roaming.

The proposed system relies on GSM authentication. The participants can only verify the identity of their communication peers though if "caller identification" is supported by the mobile network and the phone, and if it is not disabled.

F.2 Security

The security features of SSL/TLS and GSM form together a basis for the security of the proposed electronic payment system. By having a close link between the two, the security is even improved.

The Customer can securely request a purchase via SSL/TLS. The Customer will receive a confirmation via this same secure channel, and also on its mobile phone. Therefore, the Customer can double-check the Merchant's identity, and the contents of the purchase, including the amount of money to be paid.

The Merchant can rely on the GSM network to be sure to receive an authenticated payment from the Customer via the Network Operator later on. Moreover, the Customer cannot cheat by requesting its Network Operator to deduct a smaller amount of money than originally requested by the Merchant. The Merchant would notice the smaller amount of money and not send a receipt.

The Deliverer can validate a receipt by verifying the digital signature of the Merchant, and by checking if the receipt is fresh. Thus, receipts cannot be forged, and cannot be replayed. Moreover, if the Customer's mobile phone number is included in the receipt, the Deliverer could rely on GSM authentication and check if the receipt is actually presented by the original initiator of the transaction (note that for some applications, Customers might desire to be able to forward the receipt to another party that in its turn can pick up the goods).

As on top of SSL/TLS and GSM, some crucial messages are digitally signed; this decreases the need for Customers and Merchants to trust each other (i.e., they only need to trust they use the right public key, which should be ensured by the certificates that are issued by mutually trusted CAs). For example, since the receipt is digitally signed, it cannot only be verified by the Deliverer, but also by a Judge, in case of a dispute. Note that the latter also requires that the receipt includes a unique and indisputable description of the goods that should be delivered.

The Network Operator is trusted to transfer the proper amount of money from the Customer's to the Merchant's balance. It is expected to do so, as its business would otherwise quickly collapse due to negative publicity.

In some sense, the Customer's mobile phone can be considered as a secure and personal device (and care should therefore be taken that it is not easily stolen or lost). The strength of the electronic payment system proposed in this example relies particularly on the security of such a device, which is combined with the advanced yet insecure environment provided by the PC and the browser.

F.3 Privacy

The presented electronic payment system seems to offer more security than today's widely used mechanisms; however, it does not really offer more privacy. Merchants know at least the mobile phone number of their Customers. This number does not necessarily reveal a Customer's real identity (as opposed to an ordinary credit card payment). There already exist phone books with GSM numbers though. One would for example certainly not be happy when this number would be used for advertisement purposes. In fact, for this reason, some people will be reluctant to release their phone number, while they freely disclose their credit card number to merchants. The ability of hiding numbers or anonymizing customers in another way, would thus be an improvement of the system. Just as with credit card payments, the Network Operator knows exactly which Customers are buying goods from which Merchants and for what amount of money. The Network Operator will not necessarily know the actual nature of the goods though.

F.4 Other approaches

Numerous other GSM based payment systems exist. GiSMo [16] is (was) a system intended for the Internet in which customers receive a random code through SMS via a central server. This random code is then entered via the computer in order to pay. Mint [23] is a system in which each terminal/shop has a unique phone number which the customer should just call at the time of payment. Similar alternatives are Jalda [20] and Paybox [28].

In the system presented in this paper, more payment related information is exchanged via GSM, which results in a closer link between the WWW and the GSM interaction. Conceptually, it is also more general and independent of the wireless system. With more advanced mobile devices and networks, such as UMTS, more secure schemes would be possible, following the same architecture and protocol, but with different content of (and another exchange mechanism of) the messages. For example, instead of an account based protocol, electronic cash like schemes could be used. Mobile devices with built-in smart card readers would be very useful for integrating smart card based payment means as used in the physical world.

G Conclusion

Electronic commerce is already a normal part of people's ordinary life. Mobile devices, and certainly mobile phones, are currently widely spread. This paper gave a brief overview of the security properties of the World Wide Web and some existing mobile systems. The main purpose of this paper was to suggest to use a wireless system as an extension to the WWW, to provide more security and functionality. To demonstrate this combined approach, a GSM based electronic payment for the WWW was presented.

Unlike most mobile phones, some mobile devices are powerful and advanced enough to allow more or less convenient browsing and shopping. Future mobile systems will also be more secure and will offer more functionality than the GSM system or than WAP. Yet, the concept of using an out-of-band channel for electronic payment, and the combined use of a mobile device together with a normal PC, will remain very useful. For the PC and its big screen will always be far more advanced than the mobile device, but will never be mobile.

Acknowledgements

Joris Claessens is funded by a research grant of the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT). This work was also partially supported by the Concerted Research Action (GOA) Mefisto-666 of the Flemish Government.

This work was mainly initiated during a research visit at the Information Security Research Centre (ISRC) of Queensland University of Technology (QUT), Brisbane, Australia. The first author wants to thank Dr. Mark Looi of the ISRC for the interesting discussions on this topic. The authors also want to thank Dr. Silke Holtmanns of Ericsson for the various pointers to existing mobile payment systems.

An earlier version of this article was presented at and published in the proceedings of the IFIP I-NetSec'01 conference [8]. The authors want to acknowledge the anonymous reviewers for their constructive remarks and suggestions.

References

[1] American Express. Private Payments. http://www.americanexpress.com /privatepayments/.

[2] Banxafe. http://www.banxafe.com/.

[3] Steven M. Bellovin. Security Problems in the TCP/IP Protocol Suite. *Computer Communication Review*, 19(2):32–48, April 1989.

[4] Simon Blake-Wilson, Magnus Nystrom, David Hopwood, Jan Mikkelsen, and Tim Wright. TLS Extensions. IETF Internet Draft, December 2001.

[5] Bluetooth. http://www.bluetooth.com/.

[6] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. In *Proceedings of the 7th ACM Annual International Conference on Mobile Computing and Networking*, pages 180–189, 2001.

[7] Clara Centeno. Mobile Payment Industry Fora – Consolidation of Initiatives Expected. *Electronic Payment Systems Observatory – Newsletter, ePSO-N*, (8):8–12, July 2001. Available at http://epso.jrc.es/.

[8] Joris Claessens, Bart Preneel, and Joos Vandewalle. Combining World Wide Web and wireless security. In Bart De Decker, Frank Piessens, Jan Smits, and Els Van Herreweghen, editors, *Advances in Network and Distributed Systems Security – Proceedings of IFIP I-NetSec'01*, pages 153–171. Kluwer Academic Publishers, 2001.

[9] Tim Dierks and Christopher Allen. The TLS Protocol Version 1.0. IETF Request for Comments, RFC 2246, January 1999.

[10] Digipass. http://www.vasco.com/.

[11] Donald Eastlake, Joseph Reagle, and David Solo. XML-Signature Syntax and Processing. IETF Request for Comments, RFC 3075, March 2001.

[12] ETSI. Digital cellular telecommunications system (Phase 2+); Security mechanisms for the SIM Application Toolkit; Stage 1. ETSI TS 101 180 (GSM 02.48).

[13] ETSI. Digital cellular telecommunications system (Phase 2+); Security mechanisms for the SIM Application Toolkit; Stage 2. ETSI TS 101 181 (GSM 03.48).

[14] ETSI. Digital cellular telecommunications system (Phase 2+); Specification of the SIM Application Toolkit for the Subscriber Identity Module – Mobile Equipment (SIM – ME) interface. ETSI TS 101 267 (GSM 11.14).

[15] Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Web Spoofing: An Internet Con Game. In *Proceedings of the 20th National Information Systems Security Conference*, pages 95–103, 1997.

[16] GiSMo. Was available at http:// www.gismo.net/, but is currently withdrawn by Millicom International Cellular S.A.

[17] IETF Working Group. IP Routing for Wireless/Mobile Hosts (mobileip).

[18] InternetCash. http://www.internetcash.com/.

[19] Markus Jakobsson and Susanne Wetzel. Security Weaknesses in Bluetooth. In D. Naccache, editor, *Topics in Cryptology – Proceedings of the Cryptographers' Track at RSA 2001*, Lecture Notes in Computer Science, LNCS 2020, pages 176–191. Springer-Verlag, 2001.

[20] Jalda. http://www.jalda.com/.

[21] Leslie Lamport. Password Authentication with Insecure Communication. *Communications of the ACM*, 24(11):770–772, November 1981.

[22] Peter A. Loscocco, Stephen D. Smalley, Patrick A. Muckelbauer, and Ruth C. Taylor. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. In *Proceedings of the 21st National Information Systems Security Conference*, pages 303–314, October 1998.

[23] Mint. http://www.mint.nu/.

[24] Mobile Electronic Signature Consortium. http://www.msign.org/.

[25] Mobile electronic Transactions. http:// www.mobiletransaction.org/.

[26] Robert Morris and Ken Thompson. Password Security: A Case History. *Communications of the ACM*, 22(11):594–597, 1979.

[27] NTT DoCoMo. i-mode. http://www.nttdocomo.co.jp/.

[28] Paybox. http://www.paybox.de/.

[29] Proton. http://www.protonworld.com/.

[30] Eric Rescorla. *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, 2000.

[31] Eric Rescorla and Allan M. Schiffman. The Secure HyperText Transfer Protocol. IETF Request for Comments, RFC 2660, August 1999.

[32] Bruce Schneier. European Cellular Encryption Algorithms. *Crypto-Gram*, December 1999.

[33] SET Secure Electronic Transaction LLC. SET Secure Electronic Transaction Specification. http://www.setco.org/.

[34] TCPA. Trusted Computing Platform Alliance. http://www.trustedpc.org/.

[35] The UMTS Forum. http://www.umts-forum.org/.

[36] Klaus Vedder. GSM: Security, Services and the SIM. In B. Preneel and V. Rijmen, editors, *State of the Art in Applied Cryptography*, Lecture Notes in Computer Science, LNCS 1528, pages 227–243. Springer-Verlag, 1998.

[37] Visa. 3-D Secure Authenticated Payment Program. http://international.visa.com/.

[38] Mike Walker. On the security of 3GPP networks. Invited talk at Eurocrypt 2000.

[39] Wireless Application Protocol Forum. WAP Certificate and CRL Profiles. Approved 22-May-2001.

[40] Wireless Application Protocol Forum. WAP Public Key Infrastructure. Version 24-Apr-2001.

[41] Wireless Application Protocol Forum. WAP TLS Profile and Tunneling. Version 11-April-2001.

[42] Wireless Application Protocol Forum. WAP Transport Layer End-to-end Security. Approved Version 28-June-2001.

[43] Wireless Application Protocol Forum. WAP Wireless Identity Module, Part: Security. Version 12-July-2001.

[44] Wireless Application Protocol Forum. WAP Wireless Transport Layer Security. Version 06-Apr-2001.

[45] Wireless Application Protocol Forum. WAP WMLScript Crypto Library. Version 20-Jun-2001.

[46] Yougu Yuan, Eileen Zishuang Ye, and Sean Smith. Web Spoofing 2001. Department of Computer Science, Dartmouth College, Technical Report TR2001-409, July 2001.

XML access control systems: a component-based approach

Ernesto Damiani and Pierangela Samarati
Università di Milano,
Dipartimento di Tecnologie dell'Informazione, 26013 Crema - Italy
AND
Sabrina De Capitani di Vimercati
Università di Brescia,
Dipartimento di Elettronica per l'Automazione, 25123 Brescia - Italy
AND
Stefano Paraboschi
Politecnico di Milano,
Dipartimento di Elettronica e Informazione, 20133 Milano - Italy

Keywords: component-based system, XML, access control

Received: February 15, 2002

We recently proposed an access control model for XML information that permits the definition of authorizations at a fine granularity. We here describe the design and implementation of an Access Control Processor *based on the above-mentioned model. We also present the major issues arising when integrating it into the framework of a component-based Web server system.*

A Introduction

XML [2] promises to have a great impact on the way information is exchanged between applications, going well beyond the original goal of being a replacement for HTML. Given the ubiquitous nature of XML, the protection of XML information will become a critical aspect of many security infrastructures. Thus, the investigation of techniques that can offer protection in a way adequate to the peculiarities of the XML data model is an important goal.

Current solutions do not address the peculiarities of the security of XML information. Web servers may easily export XML documents, but their protection can typically be defined only at the file system level. Our proposal, presented in [3, 4, 5], introduces an access control model for XML data that exploits the characteristics of XML documents, allowing the definition of access control policies that operate with a fine granularity, permitting the definition of authorizations at the level of the single element/attribute of an XML document.

The focus of this paper is the design and implementation of a system offering the services of our access control model. We first give in Section B a brief description of the approach. Then, in Section C we describe the high-level software architecture. Section D presents the IDL interfaces of the classes which implement the services of the access control system. Finally, Section E is dedicated to the integration of the access control system with Web based systems.

The analysis contained in this paper derives from the experience we gained in the implementation of the current prototype of the system (`seclab.dti.unimi.it/~xml-sec`); our results should be helpful to those considering the implementation of security mechanisms in the WWW/XML context.

B XML Access Control Model

The access control model we present is based on the definition of authorizations at the level of the elements and attributes of an XML document.

A natural interpretation for XML documents is to consider them as trees, where elements and attributes correspond to nodes, and the containment relation between nodes is represented by the tree arcs. Authorizations can be *local*, if the access privilege they represent applies only to a specific element node and its attributes, or can be *recursive*, if the access is granted/denied to the node and all the nodes descending from it (i.e., the nodes that in the textual representation of an XML document are enclosed between the start and end tags).

We identified two levels at which authorizations on XML documents can be defined, instance and DTD (Document Type Definition, a syntax defining the structure of the document). DTD level authorizations specify the privileges of all the documents following a given DTD, whereas instance level authorizations denote privileges that apply only to a specific document. The distinction between the two authorization types may correspond to the distribution of responsibilities in an organization, as DTD authorizations may be considered derived from the requirements of the global enterprise, whereas authorizations on the instance may be the responsibility of the creator of the document. We also assume that DTD authorizations are dominated by instance level ones (following the general principle that more specific authorizations win [8, 11] and that an instance level authorization is more specific than a DTD level one), but we also consider the need for an organization to have assurance that some of the DTD authorizations are not overruled. Thus, we permit the definition of *hard* DTD authorizations, which dominate instance level ones. For cases where instance level authorizations must be explicitly defined as valid only if not in conflict with DTD level ones, we designed *soft* instance level authorizations.

Each authorization has five components: *subject*, *object*, *type*, *action* and *sign*.
The subject is composed by a triple that describes the user or the group of users to which the authorization applies, combined with the numeric (IP) and symbolic (DNS) addresses of the machine originating the request. This triple can thus permit to define controls that consider both the user and the location. Wild card character * permits the definition of patterns for addresses (e.g., `131.*` for all IP addresses having `131` as first component, or `*.it` for all addresses in the Italian domain). The authorization applies on the request only if the triple of parameters of the requester is equal or more specific in all three components of the authorization subject. For example, an authorization with subject `<Student,131.175.*,*.polimi.it>` will be applied to a request from `<Ennio,131.175.16.43,pc.elet.polimi.it>`, if `Ennio` is a member of group `Student`. The object is identified by means of an XPath [17] expression. XPath expressions may be used to identify document components in a declarative way, but they can also use navigation functions, like `child`, offering a standard and powerful way to identify the elements and attributes of an XML document. The type can be one of eight values, arising from the combination of three binary properties: DTD level or instance level; local or recursive; normal or soft/hard. The eight types, in order of priority, are: local DTD level hard (LDH), recursive DTD level hard (RDH), local instance level (L), recursive instance level (R), local DTD level (LD), recursive DTD level (RD), local instance level soft (LS), recursive instance level soft (RS). Since currently, most XML applications offer read-only access, the action currently supported by our prototype is only *read*.
A positive authorization sign specifies that the authorization permits access, a negative sign instead forbids it. *XML Access Sheets* (XASs) are used to keep all the authorizations relative to a given document or DTD.

Authorizations are then evaluated according to the following principles:

- If two authorizations are of a different type, the one with the higher priority wins (e.g., between LD and LS, LD wins).
- If two authorizations have the same type, but the object of one is more specific, the more specific wins (e.g., a recursive authorization for an element is dominated by authorizations on its sub-elements).
- If two authorizations have the same type and are on the same object, but the subject of one is more specific, the more specific wins (e.g., an authorization for the `Public` group is dominated by an authorization for the specific user `Ennio`).
- When none of the above criteria is met, a site-specific general resolution policy is used (e.g., assuming a closed access control policy, the negative authorization wins).

We refer to the presentations in [3, 5] for a complete overview of the characteristics of our solution. In this paper we intend to focus on the design and implementation of a system for fine-grained access control.

C Software Architecture: An Outline

For the access control technique outlined in Section B to be of any interest from the software designer point of view, it must be suitable for clean integration in the framework of XML-based WWW applications. To clarify this point, we shall briefly introduce the use of an *XML Access Control Processor* (ACP) as a part of a *component-based Web service* [7], where a set of reusable components are responsible of processing user *requests*.

The sample *UML Sequence Diagram* shown in Figure 1 gives a general idea of the internal operation of our processor and of its integration in a Web server system. For the sake of simplicity, in this Section we shall not deal with the transformation of the XML document, which is hidden inside a container ACP object. Also, Figure 1 does not show provisions for persistence management and caching. The ACP object wraps up entirely the computation of access permissions to individual elements and the final transformation to be performed on the XML document. The standard operation of a Web server receiving a HTTP request (**1**) from a user is represented in Figure 1 by the creation of a transient *Connection Handler* object (**2**). Then, a *Processor* is activated by the Connection Handler, and an *ACP object* is instantiated (**3**). In turn, ACP creates a *Subjects* object which fully encapsulates the subjects' hierarchy (**4**). After getting the available data about the user/group of the requestor, together with the IP address and symbolic name (**5**), ACP signals to a static *Loader/Parser* object to upload the requested XML document (**6**). The Loader/Parser translates the document into a low level object data structure based on the *Document Object model (DOM)* (not shown in Figure 1) more suitable for modification. Then, the ACP modifies the data structure according to the permissions, using the services of the transient *Subjects* object

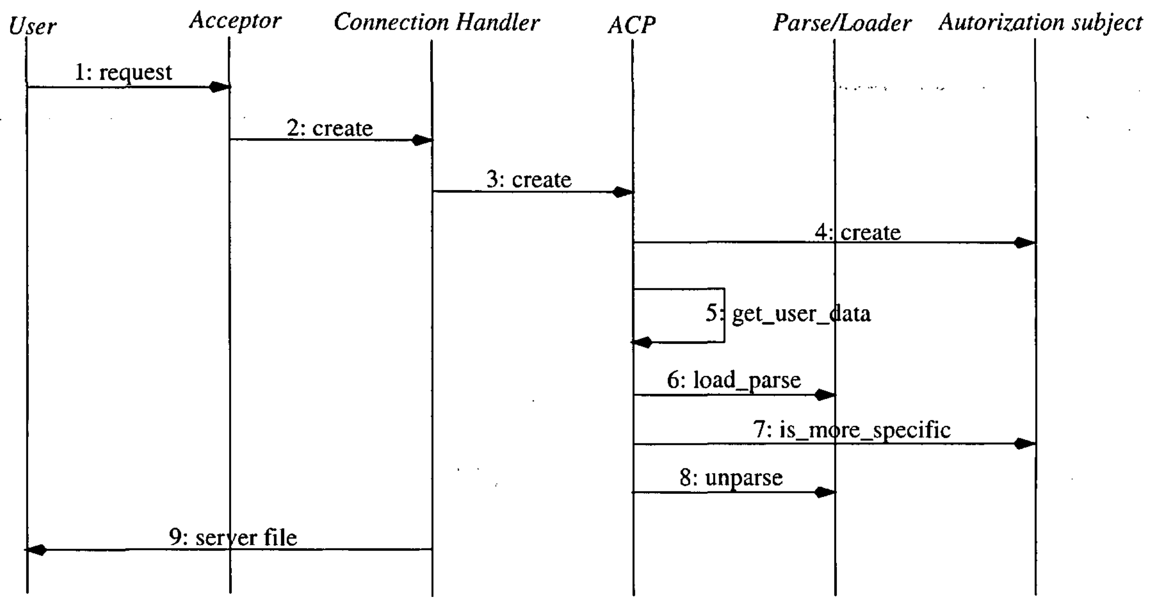


Figure 1: Sequence diagram

which fully encapsulates the subjects' hierarchy. Messages sent to the *Subjects* object (7) allow the ACP object to position the requestor in the subjects' hierarchy. After computing the transformation, the ACP object signals to the Parser (8) that the data structure can be returned to its text format, ready to be served to the user by the Connection Handler (9).

From the architectural point of view, it should be noted that our design is fully *server side*: all the message exchanges of Figure 1 except the connection itself (1) take place on the server. *Client-side* processing strategies (including client-side caching, and caching proxy servers) have been traditionally used for HTML. However, client-side solutions have been found to be less apt at XML-based Web services [7], where the contents to be transferred usually require extra processing. There may well be cases where *negotiation* could be envisioned between the client and the server as to the kinds of XML content transformations that are possible by the server and acceptable to the client; but it is clear that client-side techniques must be excluded from any sound implementation of access control. As we will see in Section D.3, the fictitious ACP object is indeed a complex object inheriting from Java *Servlet* class. A different design approach to XML access control enforcement could involve the use of a server-side XSLT engine [9] to compute transformations. However, programming the transformation engine provides fuller control over XML parsing and caching techniques, as well as on memory management, with respect to the declarative paradigm of XSLT.

D The XML-AC Package

The interface offered by the XML-AC system can be represented by a set of classes modeling the entities and concepts introduced by the access control model. Two major class families are used: one constitutes an extension of the DOM Interface defined by the W3C, the other describes all the concepts on which the ACP system is based.

D.1 Architectural Objects: the SecureDOM Hierarchy

Our system, like most XML applications, internally represents XML documents and DTDs as object trees, according to the Document Object Model (DOM) specification [16]. DOM provides an object-oriented *Application Program Interface* (API) for HTML and XML documents. Namely, DOM defines a set of object definitions (e.g., `Element`, `Attr`, and `Text`) to build an object-oriented representation which closely models the document structure. While DOM trees are topologically equivalent to XML trees, they represent element containment by means of the object-oriented *part-of* relationship. For example, a document element is represented in DOM by an `Element` object, an element contained within another element is represented as a child `Element` object, and text contained in an element is represented as a child `Text` object. The root class of the DOM hierarchy is `Node`, which represents the generic component of an XML document and provides basic methods for insertion, deletion and editing; via inheritance, such methods are also defined for more specialized classes in the hierarchy, like `Element`, `Attr` and `Text`. `Node` also provides a powerful set of navigation methods, such as `parentNode`, `firstChild` and `nextSibling`. Navigation methods allow application programs to visit the DOM representation of XML documents via a sequence of calls to the interface. Specifically, the `NodeList` method, which returns an array containing all the children of the current node, is often used to explore the structure of an

XML document from the root to the leaves.

We extended the DOM hierarchy associating to the members of the class hierarchy a Secure variant. Each Secure variant extends the base class with references to all the authorizations which can be applied to the node. Internally, each class separates the references to authorizations into 8 containers, depending on the authorization type. Each container internally keeps a list of positive and negative authorizations of the type. The IDL interface common to all *Secure* classes, written in IDL, the OMG-CORBA standard *Interface Definition Language* [13], is:

```
interface Secure {
  void addAuthorization(in int position,
         in Authorization authorization);
  AuthorizationList defineFinalLabel(
    in AuthorizationList labelsUpperLevel);
  boolean isAllowed();
  void prune();}
```

Each Secure variant extends the base class with references to all the subjects of the authorizations which can be applied to the node. The answer to method isAllowed derives from the analysis of all the authorizations. From this interface it is possible to define the interfaces of each Secure variant of the DOM classes, using multiple inheritance in IDL definitions. For example, the definition of the SecureNode class is interface SecureNode: Node, Secure {}. Recently, security provisions for the XML data model have been adopted by some commercial software products: for instance, Tamino (www.software-ag.de) provides an authorization check to grant or deny access to XML nodes (elements and attributes of documents) stored in Tamino's data store. Tamino's approach is related to our inasmuch it makes access control available at the structural level, that is, each XML node or any of its descendants can be protected individually based on its position in the document tree. However, Tamino's authorizations are specified in an attribute of an additional Access Control Element, which needs to be added for each secured node. In contrast, our DOM extension imposes a limited increase in the cost of the document representation, as authorizations are kept separate. Also, our proposal uses XPath in authorizations and permits the declarative specification of authorization objects.

D.2 Application Objects: The Access Control Classes

We describe here the main classes of the Access Control Processor: UserGroup, User, AuthorizationLabel, AuthorizationType, AuthorizationSubject, and Authorization.

Class UserGroup describes the features common to a user and a group: both have a name and appear in the user/group hierarchy. The services offered by the class are the storage of the hierarchy on users/groups, method addChild that permits to add a new user/group in the hierarchy, and method isDescendent that permits to determine if the user/group belongs, directly or indirectly, to another group.

```
interface UserGroup{
  attribute string Name;
  void addChild (in UserGroup childToAdd);
  boolean isDescendent(in UserGroup ancestor);}
```

Class User is a specialization of class UserGroup and extends it with all the information specific to users, like the real person name. Method checkPassword implements the cryptographic function that determines if the password returned by the user corresponds to the stored value. Method setPassword permits to change the password.

```
interface User:  UserGroup{
  attribute string FirstName;
  attribute string LastName;
  boolean checkPassword(in string passwordToCheck);
  void setPassword(in string newPassword);}
```

Class AuthorizationLabel contains an enumerative type that describes the three values (positive, negative, and undefined) of the security label that can be assigned to a node, after the evaluation of the existing authorizations. Its methods permit to set and retrieve the value.

```
interface AuthorizationLabel{
  enum Label_t (positive, negative, undefined);
  void setPositive();
  void setNegative();
  void setUndefined();}
  boolean isPositive();
  boolean isNegative();
  boolean isUndefined();}
```

Class AuthorizationType describes the possible types of authorization. Its methods permit to set and to retrieve the authorization type (local or recursive, on the document or on the DTD, and hard or soft).

```
interface AuthorizationType{
  enum AuthType_t (LDH, RDH, L, R, LD, RD, LS, RS);
  void setLocal();
  void setRecursive();
  void setOnInstance();
  void setOnInstanceSoft();
  void setOnDTD();
  void setOnDTDHard();
  boolean isLocal();
  boolean isRecursive();
  boolean isOnInstance();
  boolean isOnInstanceSoft();
  boolean isOnDTD();
  boolean isOnDTDHard();}
```

Class AuthorizationSubject describes the triple ⟨user-group, IP address, symbolic address⟩ that identifies the subjects to which the authorizations must be applied. The class offers methods to get and assign the components of the addresses and a method isEqualOrMoreSpecific to determine if one subject is equal or more specific than another subject.

```
interface AuthorizationSubject{
  void setUserGroup(in UserGroup userGroupToSet);
  UserGroup getUserGroup();
  void setIpAddress(in string IPAddrToSet);
  string getIPAddress();
  void setSnAddress(in string SymbAddrToSet);
  string getSnAddress();
  boolean isEqualOrMoreSpecific(
         in AuthorizationSubject asHigher);}
```

Class `Authorization` represents the authorizations that are defined on the system. Each authorization is characterized by a subject (class `AuthorizationSubject`), an object (represented by an XPath expression, managed by classes defined in an external XSL implementation), the sign (represented by an `AuthorizationLabel` component for which value *undefined* is not admitted), the action (currently a simple string), and finally the type (represented by a component of class `AuthorizationType`).

```
interface Authorization{
  attribute AuthorizationSubject subject;
  attribute XPathExpr object;
  attribute AuthorizationLabel sign;
  attribute AuthorizationType type;
  attribute string action;}
```

D.3 Deploying the Package

We implemented the above classes in Java and used them to realize a prototype of the Access Control Processor with a Java servlet solution. Java servlets, designed by Sun and part of the Java environment, appear as a set of predefined classes that offer services that are needed for the exchange of information between a Web server and a Java application. Examples of these classes are `HttpSession` and `HttpRequest`. Java servlets constitute a simple and efficient mechanism for the extension of the services of a generic Web server; the Web server must be configured to launch the execution of a Java Virtual Machine when a request for a URL served by a servlet arrives, passing the parameters of the request with a specified internal protocol.

The Java classes we implemented could also be used in a different framework, using a solution like JSP (Java Server Pages). Actually, JSP is internally based on servlets, but it offers an easier interface to the programmer, requiring the definition of HTML/XML templates which embed the invocation of servlet services. We have already demonstrated the use of the prototype inside a JSP server.

There are several other architectures that could be used and whose applicability we plan to investigate in the future. Since we gave an IDL description of the classes that constitute the implementation of our system, it is natural to envision a solution based on the distributed object paradigm, using protocols like RMI/IIOP (for the Java implementation) or the services of a generic CORBA broker (where the services are implemented by objects written in a generic programming language).

E Integration with Web-based systems

We are now ready to describe how our access control system can be integrated in a Web-based framework for distribution and management of XML information. This architecture needs to include a number of components and a careful study of their interaction with access control is of paramount importance to achieve an efficient implementation. Some of the solutions that we describe have not yet been implemented in the current prototype, but we plan in the near future to integrate all of them into the system.

E.1 Linking XAS to XML Documents and DTDs

In our approach, authorizations are expressed in XML. Each XML document/DTD is associated with an *XML Access Control Sheet* (XAS) that includes the authorizations that apply to the document. As XASs contain access control information for XML documents and DTDs, links must be provided allowing the system, upon receipt of a HTTP request for an XML document, to locate the XAS associated with both the document itself and its DTD. In current XML practice, association between XML documents and their DTDs is made by either *direct inclusion* (the DTD is embedded in the XML document) or by *hypertext link* (the XML document contains the URL of its DTD). Neither technique seems appropriate for linking documents and DTDs to XASs as they would interfere with the normal processing of XML documents, and pose the problem of managing access control for legacy documents not linked to any XAS specification. Luckily enough, we can rely on the abstract nature of XML *XLink* specification [6] to define *out-of-line* links that reside *outside* the documents they connect, making links themselves a viable and manageable resource. The repertoire of out-of-line links defining access control mappings is itself an XML document, easily managed and updated by the system manager; nonetheless it is easily secured by standard file-system level access control. We propose to set up a suitable *namespace*, called AC, which is for the time being aimed at reserving the standard tag name <XAS> to denote off-line links between documents, DTDs and XASs. The DTD of the documents containing the mappings from XML documents to DTDs and to XASs can be written as follows:

```
<!ENTITY % xlink
        " type    CDATA  #FIXED 'arc'
          role    CDATA  #FIXED 'access control'
          title   CDATA  #FIXED 'access control'
          actuate CDATA  #FIXED 'auto'
          from    CDATA  #REQUIRED
          to      CDATA  #REQUIRED">

<!ELEMENT XAS EMPTY>
<!ATTLIST XAS  % xlink
xmlns:xlink CDATA 'http://www.w3.org/1999/xlink'>
```

Note that, in the private documents specifying link sets for each site and at the DTD level, the name of the XAS element will be preceded by the mention of the AC namespace in order to avoid ambiguity. In the above DTD definition, we rely on a reusable XML *entity* to group the attributes needed to set up an out-of-line link between a document and its access control information. Namely, out-of-line links are identified by the `type` attribute being set to `"arc"`, and by the presence of required `from` and `to` attributes instead of the usual `href` used for embedded links. The `actuate` attribute is set to `"auto"`, meaning that the traversal of the link will be automatically made by the system and not revealed to the user. Finally, the `role` and `title` attributes are used primarily for descriptive purposes and are therefore not mandatory.

E.2 XML-AC Support for Sessions

In the current prototype, sessions are managed by class `HttpSession`, a component of the Java servlet environment. Class `HttpSession` keeps track of the series of requests originating from the same user. Using the services of `HttpSession` it is possible to ask only once to the user to declare his identity and password. The implementation of class `HttpSession` permits to manage sessions in two modes, with or without cookies. When the client has cookies enabled, `HttpSession` may store a session identifier in the client cookies and use it to identify the request; if cookies are not enabled, sessions are identified by storing the session identifier as a parameter of the requests that are embedded into the page which is returned to the user. Since users often do not enable cookies, it is important to be able to manage sessions independently.

We observe that the solution we implemented, based on the services of class `HttpSession`, is adequate for our context, where the goal was a demonstration of the capabilities of the access control model. An environment with strong security requirements should probably plan a different implementation of the session management services, using adequate cryptographic techniques to protect the connection.

E.3 A Multithreaded Server Framework

To guarantee efficient and effective integration of access-control in the framework of Web-based systems, two basic problems must be solved:

Quality of Service The emergence of the World Wide Web as a mainstream technology has highlighted the problem of providing a high quality of service (*QoS*) to application users. This factor alone cautioned us about the risk of increasing substantially the processing load of Web server.

Seamless Integration A second point to be mentioned regards how to provide XML access control as seamlessly as possible, without interfering with the operation of other presentation or data-processing services. Moreover, the access control service should be introduced on existing servers with minimal or no interruption of their operation.

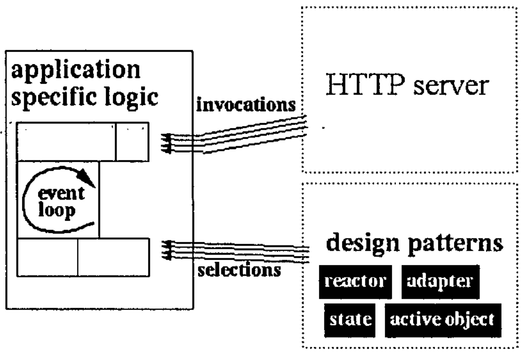


Figure 2: Cocoon-style multi-threading technique

To deal with these problems, we chose an integrated (yet modular) approach, that supports reuse allowing for different deployment solutions according to implementation platforms' performance profiles. In fact, besides being deployed as a single-thread servlet invoked by the Connection Handler, as in our current prototype, our processor can be easily interfaced to a *Dispatcher* registered with an *Event Handler*. Dispatcher-based multi-threading can be managed *synchronously*, according to the well known *Reactor/Proactor* design pattern [15] or *asynchronously*, as in the *Active Object* pattern. In this section we shall focus on the former choice, as it facilitates integration of our XML access control code in the framework of existing general-purpose server-side transformers based on the same design pattern like *Cocoon* [1]. Figure 2 depicts the Reactor-based multi-threading technique.

In order to avoid being a potential bottleneck for the server operation, our Access Control system needs to manage effectively a high number of *concurrent requests*. Multi-threaded designs are currently the preferred choice to implement Web-based, high-concurrency systems. This is also our design choice for our components. However, it must be noted that no Java-based design of multi-threading components has full control on thread management: when running on an operating system that supports threads, the *Java Virtual Machine* automatically maps Java threads to native threads [10], while when no native thread support is available, the JVM has to emulate threads. In the latter case, the emulation technique chosen by the JVM implementors can make significant difference in performance. In the sequel, we shall briefly describe the Java thread management technique used for the implementation of our processor, providing full synchronization between threads when accessing the same DOM and `AuthorizationSubject` objects. To clarify the synchronization problem associated with multi-threading, consider two access control tasks that need to be executed in parallel (see Figure 3(a)). For the sake of simplicity both tasks are naturally subdivided into four *atomic* non-interruptible subtasks, loosely corresponding to actions from (4) to (7) of Section C. In a "naive" multi-threaded implementation

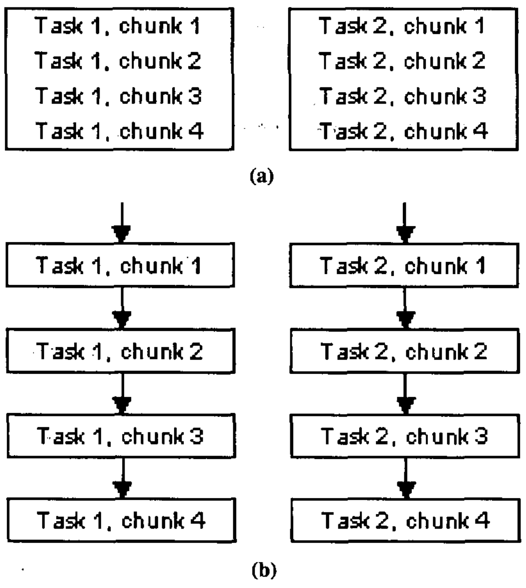


Figure 3: Two AC tasks to be executed in parallel (a) and their subdivision into four atomic sub-tasks (b)

```
Runnable[] task = new Runnable[]
{
 new Runnable(){ public void run(){/* execute sub-task 1 */}},
 new Runnable(){ public void run(){/* execute sub-task 2 */}},
 new Runnable(){ public void run(){/* execute sub-task 3 */}},
 new Runnable(){ public void run(){/* execute sub-task 4 */}},
};
for( int i = 0; i < task.length; i++ )
{task[i].run();
 Thread.getCurrentThread().yield();
}
```

Figure 4: Sample Java code for the synchronous dispatcher

```
Runnable[] two_tasks = new Runnable[]
{
 new Runnable(){public void run(){/* execute task 1, sub-task 1 */}},
 new Runnable(){public void run(){/* execute task 2, sub-task 1 */}},
 new Runnable(){public void run(){/* execute task 1, sub-task 2 */}},
 new Runnable(){public void run(){/* execute task 2, sub-task 2 */}},
 new Runnable(){public void run(){/* execute task 1, sub-task 3 */}},
 new Runnable(){public void run(){/* execute task 2, sub-task 3 */}},
 new Runnable(){public void run(){/* execute task 1, sub-task 4 */}},
 new Runnable(){public void run(){/* execute task 2, sub-task 4 */}},
};
for( int i = 0; i < two_task.length; i++ )
 { two_tasks[i].run();
   Thread.getCurrentThread().yield();
}
```

Figure 5: The interleaving dispatcher

of our processor, each task would be executed on its own thread. However, the only way to preserve atomicity using this technique would be to explicitly synchronize threads by means of *semaphores*. Fortunately, the additional complexity and overhead involved in explicit synchronization can be easily avoided in our case.

Synchronous dispatching A synchronous dispatcher can be used to solve the synchronization problem by simulating multi-threading within a single Java thread. To illustrate the evolution of our design from a single task divided into portions to a synchronous dispatcher, consider first the subdivision of each task of Figure 3(a) into four independent sub-tasks, depicted in Figure 3(b). From the Java implementation point of view, each sub-task can now be straightforwardly defined as the `run()` method of a `Runnable` object [12]. Then, the objects can be stored into an array, and a *scheduler* module can be added executing the objects one at a time. `Sleep()` or `yield()` calls mark the transition between sub-tasks.

As anticipated, this code is a simple implementation of Schmidt's Reactor design pattern [15]. The effect is essentially the same as several threads waiting on a single *ordered binary semaphore* that is set to `true` by an event. Here, the programmer retains full control over the sequence of subtask execution after the event. In our AC processor, however, a slightly more complex technique should be used, as we need to execute complex transformation tasks concurrently, each of them being subdivided into atomic sub-tasks. To deal with this problem, the synchronous dispatcher of Figure 4 can be easily modified [12] to provide *interleaving* (Figure 5).

The behavior of the code in Figure 5 allows for a multi-threading *cooperative* system (in which threads explicitly yield control to other threads). Of course, this synchronous dispatching technique is aimed at native multi-threaded operating systems, where all the subtasks are executing on a single operating system-level thread. In this case, there is no synchronization overhead at all, and no expensive context switch into the host operating system's kernel. It should be noted that several dispatchers could be used, each running on its own thread (as in Sun's *green thread* model [14]), so that cooperative and preemptive threads may share the same process.

F Conclusion

In this paper we presented the major results of the study we did before the implementation of the processor for the proposed access control model for XML data. Most of the considerations we present are not specific to our system, but can be of interest in any context where services for the security of XML must be implemented.

There are several directions where our work can be extended and that offer interesting opportunities. For instance, we focused on multi-threading techniques to obtain efficient concurrent execution of access control tasks. However, synchronization overhead is obviously not the only performance problem. Other techniques rather than round-robin interleaving could be adopted: e.g., the XML access-control service could adaptively optimize itself to provide higher priorities for smaller requests. These techniques combined could potentially produce a system highly responsive and with an adequate throughput. The next release of the ACP plans to implement the prioritized strategy.

Acknowledgments

The authors wish to thank Daniel A. Menascé for interesting discussions about XML processing performance issues. This work was supported in part by the European Community within the FASTER Project in the Fifth (EC) Framework Programme under contract IST-1999-11791 and by the Italian MURST DATA-X project.

References

[1] Apache Software Foundation. Cocoon, a Java publishing framework. http://xml.apache.org/cocoon, 2000.

[2] T. Bray, J. Paoli, C.M. Sperberg-McQueen, and E. Maler. *Extensible Markup Language (XML) 1.0 (Second Edition).* World Wide Web Consortium (W3C), October 2000. http://www.w3.org/TR/REC-xml.

[3] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Securing XML documents. In *Proc. of EDBT 2000,* Konstanz, Germany, March 2000.

[4] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Controlling access to XML documents. *IEEE Internet Computing,* 5(6):18–28, November/December 2001.

[5] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Design and implementation of an access control processor for XML documents. *Computer Networks,* 33(1–6):59–75, June 2000.

[6] S. DeRose, E. Maler, and D. Orchard. *XML Linking Language (XLink) Version 1.0.* World Wide Web Consortium, June 2001. http://www.w3.org/TR/xlink.

[7] J. Hu, I. Pyarale, and D. Schmidt. Applying the proactor pattern to high performance web services. In *Proc. of the 10th International Conference on Parallel and Distributed Computing,* Las Vegas, Nevada, October 1998.

[8] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems,* 26(2):214–260, June 2001.

[9] M. Kay. *XSL Transformations (XSLT).* World Wide Web Consortium, December 2001. http://www.w3.org/TR/xslt20/.

[10] D. Lea. *Concurrent Programming in Java.* Addison Wesley, 1996.

[11] T.F. Lunt. Access Control Policies for Database Systems. In C.E. Landwehr, editor, *Database Security, II: Status and Prospects,* pages 41–52. North-Holland, Amsterdam, 1989.

[12] B. Marchant. Multithreading in java. http://www.javacats.com/US/articles/multithreading.html, 1996.

[13] T.J. Mowbray and R.C. Malveau. *CORBA Design Patterns.* John Wiley & Sons, 1997.

[14] M.L. Powell, S. Kleiman, S. Barton, D. Shah, D. Stein, and M. Weeks. *SunOS Multi-thread Architecture.* Sun Microsystems, 1998.

[15] D. Schmidt. *Reactor: A Object Behavioral Pattern for Concurrent Event Demultiplexing and Dispatch,* chapter 29. Pattern Languages of Program Design. Addison Wesley, 1995.

[16] World Wide Web Consortium (W3C). *Document Object Model (DOM) Level 1 Specification Version 1.0,* October 1998. http://www.w3.org/TR/REC-DOM-Level-1.

[17] World Wide Web Consortium (W3C). *XML Path Language (XPath) 2.0,* December 2001. http://www.w3.org/TR/xpath20.

How aspect-oriented programming can help to build secure software

Bart De Win, Bart Vanhaute and Bart De Decker
Departement of Computerscience, K.U.Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium
{bartd,bartvh,bart}@cs.kuleuven.ac.be

Keywords: aspect-oriented programming, security, separation of concerns

Received: January 23, 2002

Since many applications are too complex to be designed in a straightforward way, mechanisms are being developed to deal with different concerns separately. An interesting case of this separation is security. The implementation of security mechanisms often interacts or even interferes with the core functionality of the application. This results in tangled, unmanageable code with a higher risk of security bugs.
Aspect-oriented programming promises to tackle this problem by offering several abstractions that help to reason about and specify the concerns one at a time. In this paper we make use of this approach to introduce security into an application. By means of the example of access control, we investigate how well the state of the art in aspect-oriented programming can deal with separating security concerns from an application. We also discuss the benefits and drawbacks of this approach, and how it relates to similar techniques.

A Introduction

In the open world of the Internet it is very important to use secure applications, servers and operating systems in order to avoid losing valuable assets. However, developing a *secure* application in an open, distributed environment is a far from straightforward task.

Security should never be considered a minor issue when developing software. Adding security to an application as an afterthough is almost always a bad idea, and will very often lead to bugs and vulnerabilities. Security should be an issue in each phase of the development process, from the first gathering of requirements to the testing and final deployment (see the Common Criteria [1]).

During requirements gathering and even during (high-level) analysis of the problem, it is not too difficult to take security considerations into account. They can be dealt with rather independently of the application. However, later in the development cycle it becomes harder and harder to correctly handle security requirements. Besides the fact that both the application and the security mechanisms become more elaborate here, the real problem lies in the interaction between the application functionality and the security concerns. At the base of this problem is a structural mismatch between the application and the required security solution. Confidentiality for instance requires both sealing and unsealing of sensitive information. Although they are logically joint and in fact very similar, they are typically spread over several places in the application.

The source of this structural mismatch does not lie in the way the application is modularized. Restructuring the application will only shift or transform the mismatch. State-of-the-art development approaches just can not handle this kind of modularity. What is needed is support for dealing with this structural mismatch explicitly in every phase of the development.

Aspect-orientation is an attempt at answering this need. It has constructs to declare how modules cross cut one another. In this paper we use AspectJ, a specific tool that helps dealing with cross cutting at the implementation level. As a beneficial side effect of using this tool, the implementation of the security mechanism and the basic logic can be reused for other applications, when properly designed.

The structure of this paper is as follows. First a more detailed description of the context of our approach is explained. Then, we will give a short introduction to AspectJ, an aspect-oriented programming language for Java. Through a concrete example we will explain how this can be used to secure an application. This mechanism will be generalized in order to construct a framework of security aspects, after which the advantages/disadvantages of the approach will be discussed. We end this paper with a section on related work where we compare the aspect-oriented approach with other existing techniques.

B Applicability of the approach

As we will show in this paper, the technique of aspect-oriented programming can help considerably in designing secure software. However, we do not want to claim it is the silver bullet. Some problems map nicely onto the ideas of aspect-oriented programming and hence can be solved quite elegantly, while others cannot. In this section we discuss the scope of this novel technique by grafting it upon the process of secure software engineering.

For this discussion, we will look at the development of secure software from two orthogonal viewpoints. A first, straightforward viewpoint comprises the well-known phases during the development of a typical application. This starts from requirements gathering, through analysis, design and implementation to end with the actual deployment. Remark that we are not interested in the sequential order of the phases, merely in the different levels of detail.

The second viewpoint focuses on the different responsabilities in an application. For security, we can distinguish three different categories. The first one involves the core business functionality of the application. The second category corresponds with code implementing basic security operations (such as how to encrypt something): it is part of the implementation, but is at the same time totally unrelated to the business logic of the application. The third category is the set of code that specifies where and how to use the security operations in the application. This third type of responsability actually constitutes the relationship between business and security logic in an application.

In order to clarify the last viewpoint, let us apply it to a typical online banking system. The code that implements how to check the balance of your account and how to transfer money from one account to another is part of the business logic of the application and belongs as such to the first category. Then, there is code that implements how to secure communication data from eavesdropping and tampering. This code could be reused for different types of applications, i.e. all applications requiring that information (in transit) should be secured. Hence, this code belongs to the second category. Finally, some code in the application is responsible for activating the security mechanisms at the right place and at the right time, i.e. every time account information is transferred from the bank to the client. This code links the security mechanisms to the application and is therefore part of the third category. Looking at this example in another phase during the development cycle as defined in the first viewpoint will produce similar results.

Merging the two orthogonal viewpoints results in a table as shown in Figure 1. As you will notice, the distinction between application, relationship and security code is apparent[1] in every phase of the development process. In every cell of this table, uncareful development can result in security problems. To give an idea of the possible problems, we filled in the different categories of the Common Vulnerabilities and Exposures[3] (CVE). For instance, the category *access validation errors* originates from implementation errors in security related code. *Race condition errors* are not only present in security code, they can also arise from the uncareful incorporation into the application.

Note that the table in Figure 1 is not complete in the sense that it does not contain all known security problems. CVE primarily focuses on security flaws in operational systems. Thus, most of the problems are situated in the lower rows of the table.

Aspect-oriented programming, and more in particular the AspectJ tool we use, (currently) covers only a specific part of the table. It provides a way to cleanly separate different responsabilities in a system and it offers a means to specify how they should be combined. However, compared to standard object-oriented techniques, it does not offer enhanced support to design and implement the actual business logic, nor the security logic. As such, AspectJ primarily operates at the inner column of the table. Furthermore, its tool support is still limited to the design and the implementation phase. This demarcates the theoretical scope (represented by the grey rectangle) of the approach discussed in this paper. Applying AspectJ to problems situated outside this scope will result in unnatural, forced solutions.

C Introduction to AspectJ

In this section we will briefly discuss the principles of the AspectJ language [2]. AspectJ is a Java language extension to support the separate definition of crosscutting concerns. In AspectJ, *pointcuts* define a collection of specific points in the dynamic execution of a java program. Pointcut definitions are specified using primitive pointcuts designators such as the execution of a method, the creation of a specific type of object, etc. Primitive pointcuts can be combined using logical operators.

On pointcuts, *advice* can be defined in order to execute certain code. AspectJ supports before and after advice, depending on the time the code is executed. E.g. before advice on the execution of a method will make sure that the code specified in the advice will be executed before the particular method is actually executed. In addition, both advices can be combined into one, the around advice. The use of the pointcut and advice constructs will become clearer when we discuss a concrete example.

The definition of pointcuts together with the specification of advice on these pointcuts forms an aspect definition.[2] An aspect is also similar to a class and can as such contain data members, methods, etc. Instances of an aspect can be associated with an object, but also with other runtime elements like control flow. The instance is automatically created when the target of the association (e.g., a specific object) is active. To conclude, an aspect defines extra functionality and a description of where it should be applied.

To deploy the aspects in a concrete application, AspectJ provides a special compiler that parses all application and aspect code to produce, through some intermediate transformations, the woven application in the form of Java bytecode.

[1]We use dotted lines in the first and last row of the table to denote that at these phases the distinction between the three responsabilities is less clear, since at that time you are dealing with one monolithic system, rather than with separate parts.

[2]AspectJ also supports other constructs like Introduction. Since they are not used in this paper, we will not discuss them here.

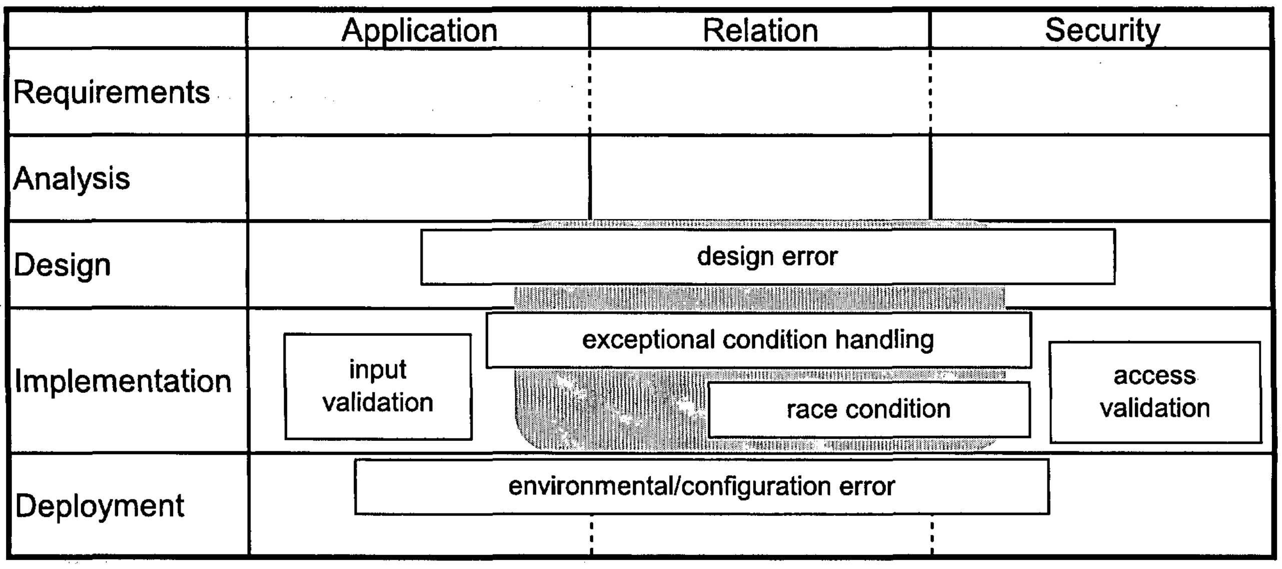


Figure 1: Table illustrating the application area of the approach.

D Security as an aspect

The technique of aspect-oriented programming helps us to tackle the problems described in the introduction. It provides a mechanism to combine separate pieces of code easily, which encourages the separate implementation of non-functional issues like security. Using this divide and conquer strategy, the overall complexity of the problem is reduced considerably. Moreover, it allows different specialists (e.g., an application engineer, a security engineer, ...) to work simultaneously and to concentrate on their field only.

D.1 An example: Access Control

The example presented in this section discusses how to perform access control in an application. We have chosen this problem because it clearly shows that security related code can be separated from the functionality of the application in an elegant way.

Basically, access control can be described as follows: at a certain point, the application requires credentials from the user, after which access to certain resources is allowed or denied based on the user's identity. However, this abstract view hides several decisions. The key to convert the above description into an aspect-oriented application is the identification of the important domain concepts and their mutual dependencies.

First, what is the exact entity that has to be authenticated? From a user-oriented view[3], the user of the global application might be a reasonable decision here. In this case the user has to login once, after which this identity is used during the rest of the application. However, the granularity of this approach will clearly not suffice for some applications, like a multi-user or a multi-agent system. A second approach consists of linking the identity to a certain object in the application. Here, login information will be reused as long as the actions are initiated by the same object. On the other hand, the identity of the user might change over time. It is then necessary to associate the identity with the initiator of a certain action. In this case, an authentication procedure is required every time the specific action is initiated.

Next, for what resources do we want to enforce access control? Again, one can think of different scenarios. An identity might require access to one resource instance (e.g., a printer). When more instances are available, one could have access to the whole group or to only a particular subgroup. In case of different resource types the identity could require access to a specific combination of these resources. In general, this will often correspond to a combination of (some parts of) application objects.

A last but not less important consideration deals with specifying how and where the resources are accessed. This path from the authenticated entity to the resources is necessary to pass login information to the access control mechanism. In a distributed system for instance, authentication and access control might be performed on different hosts. In that case, authentication information must evidently be passed to the access control mechanism in order to ensure correct execution. One obvious example of such access path is the invocation of a specific service of a resource.

Each of the above concepts (identity, resource and access path) is actually a crosscutting entity to the application and maps closely to an aspect. In fact, the three concepts capture the conceptual model of access control and they can as such be used for most access control problems. Note that we did not discuss any issues concerning concrete mecha-

[3]From another point of view, the source of the application code might be the subject of authentication. While the mechanism to establish the correct identity of the code originator might be different, the overall authorization mechanism described in this paper will still be applicable.

nisms for authentication and access control. Although certainly relevant, it is important to realize that these are implementation decisions and will depend on the underlying security architecture.

Figure 2 shows the details of one particular case of access control, where each user is authenticated once and where access is checked for each invocation of a particular service. The implementation of the other variants of access control would be fairly similar. To make it more readable, the aspect code of the example is written for a minimal application that consists of a *Server* implementing a *ServerInterface* with a method *service* and a *Client* invoking this *service*.

The **Identification** aspect is used to tag the entities that must be authenticated. The idea is that the subject included in the Identification aspect is used to check whether access is allowed or denied[4]. In the example, every object of the class Client is considered as a possible candidate. By using the perthis association, the subject information will be available as if it were glued to the particular Client object.

The serviceRequest pointcut of the **Authentication** aspect specifies all places where the service method of the ServerInterface is invoked. Through the use of the powerful percflow construct, the Authentication aspect, and more important, its Subject data member, travels along with the invocation. Thus, it is able to pass the authentication information to the access control mechanism. Before the method is actually invoked, the identity information from the Identification aspect is copied to the local Subject of this aspect. If the Client was not yet authenticated, this is the right place to do this.

Finally, the **Authorization** aspect checks access based on the identity information received through the Authentication aspect. This check is performed for every execution of the service method (checkedMethods pointcut). In this example, the login and access control phase are written in pseudo code. The actual code will make use of the underlying security architecture. In our implementation, we have used the Java Authentication and Authorization Service [15] for this purpose.

Weaving the above aspects into the application will result in a new, more secure version of the application. In the latter, the access controlling code defined in the Authorization aspect will be executed before every invocation of *service()*. At this point, the application will continue its normal execution if access is granted, however an exception will be thrown if the (un)authenticated entity is not allowed to do so. As such, conventional[5] use of the method *service()* will be restricted to certain users depending on the security policy, just as would have been the case by coding the access control mechanism directly into the applition code.

[4]All objects that don't have an Identification aspect will not be able to execute *service*.

[5]By predicting the output of the aspect weaver, one might be able to circumvent this access control mechanism under certain circumstances. We discuss this problem in detail in section E.

D.2 Generalization of the example

The deployment of each of the crosscutting entities described in the previous section depends heavily on the actual type and implementation of the particular application. For example, an email client will work on behalf of one user, while a multi-user agenda system will want to distinguish his users. Also, objects representing a user will clearly differ in structure and behaviour between separate applications. In general, it is impossible to define one set of aspects that will be applicable to all possible applications. Therefore, a more generic mechanism is desirable that separates the implementation of security mechanisms from these choices.

Given the previous example one might notice that the deployment decisions are actually contained in the pointcut definitions, which define where and when an advice or an aspect has to be applied. For this purpose, AspectJ has the ability to declare pointcuts abstract and afterwards define them in an extended aspect. Using this mechanism, it is possible to build a general authorization aspect and redefine the included abstract pointcuts depending on a specific application. To illustrate this technique, we have applied it to the example of the previous section. The result is sketched[6] in figure 3. In order to use these generic aspects in a concrete situation, one has to extend the abstract aspects and fill in the necessary pointcuts based on the specific security requirements of the application.

A major advantage of this generalization phase is the ability to reuse the core structure of the security requirement. Since this will be similar for every situation, it is not necessary to reinvent the wheel for every case. It should be properly designed by a qualified person only once, after which aspect inheritance enables easy reuse.

D.3 Towards a framework of security aspects

For a secure distributed application, other security requirements besides authentication and authorization must be considered, such as confidentiality, non-repudiation, etc. We will now briefly describe how some could be implemented using aspects.

Encryption of objects is required for *confidentiality* and *integrity*. This is a quite straightforward task using the Java JCA/JCE [12]. Two issues have to be considered. First, one has to decide where and how to insert this into the application. One possibility is to encrypt objects while they are written to a specific stream. For this case, the stream can be wrapped by a specific encryption stream. Another possibility is to encrypt objects whenever they are serialized. Therefore, the readObject() and writeObject() methods of the object should be overridden to include encryption here. Second, there is the issue of how to get or store the cryptographic keys. Similar to the identity in the previous section,

[6]A real implementation would have extra work-arounds for some limitations of the current version of AspectJ. See discussion.

```
aspect Identification perthis(this(Client)) {
    public Subject subject = null;
}

aspect Authentication percflow(serviceRequest()) {
    private Subject subject;

    pointcut serviceRequest(): call(* ServerInterface+.service(..));

    pointcut authenticationCall(Object caller):
        this(caller)
        && serviceRequest()
        && if(Identification.hasAspect(caller));

    before(Object caller): authenticationCall(caller) {
        Identification id = Identification.aspectOf(caller);
        if(id.subject == null) {
            <login>
            subject = id.subject;
        }
    }

    public Subject getSubject() {
        return subject;
    }
}

aspect Authorization {
    pointcut checkedMethods() : within(Server) && execution(* service(..));

    Object around(): checkedMethods() {
        Authentication au = Authentication.aspectOf();
        Subject subject = au.getSubject();
        boolean allowed = <check access control>;
        if(allowed) {
            return proceed();
        } else {
            throw new AccessControlException("Access denied");
        }
    }
}
```

Figure 2: Aspect code for object-based access control

```
abstract aspect Identification perthis(entities()) {
    abstract pointcut entities() ;

    public Subject subject null ;
}

abstract aspect Authentication percflow(serviceRequest()) {
    private Subject subject;

    abstract pointcut serviceRequest() ;

    ...
}

abstract aspect Authorization {
    abstract pointcut checkedMethods() ;

    ...
}
```

Figure 3: Generalized aspect code for access control

one has to find some entity in the application with which the keys will be associated. The implementation will vary according to how the keys are to be acquired.

Non-repudiation requires the generation of proof for certain events in the system, e.g. the invocation of a specific method. This is quite similar to the problem of access control described above. One crosscutting entity defines the identity that wants to generate the proof. Another entity stores and manages these proofs. And finally, a third entity defines where and how proofs should be generated and passed along.

In the end, a combination of all the security aspects could form the basis of an aspect framework for application security. This framework will consist of generalized aspects for each of the security requirements. Note that several aspect implementations, depending on different underlying security mechanisms, may be included for the same security requirement. The deployment of the framework for a concrete application will then come down to choosing the appropriate aspects and defining concrete pointcut designators for them. A more elaborate discussion on this security framework can be found in [18].

E Discussion

The separation of a (complex) application into an application specific part, a security part and a part that details the relation between the two is a noble goal, in the spirit of advanced separation of concerns. The technology of and the ideas behind aspect-oriented programming hold a promise of achieving this goal.

Although the technology has not yet fully matured, the current possibilities of AspectJ already allow us to fill up a number of gaps in the table introduced in section B, where state-of-the-art tool support is lacking:

- The mere fact that this kind of separation is possible at the code level already makes the management and maintenance of this code easier. There can be distinct packages for pure application functionality, for pure security code and a package that defines the points where that security is to be applied. This actually suggests three distinct tasks to develop a secure application[7]: build the application, develop a generic security aspect architecture and specify the aspect deployment pointcuts.

- Security should be applied at all times, if it is to be applied correctly. By looking at the definition of the pointcuts in the aspect that implements the particular security concern, a security engineer immediately knows all the places where this concern will be used, given that the AspectJ compiler does its job correctly.

- The implementation of the security mechanisms does not have to be copied several times. All the implementation code can be gathered within a small number of advices, perhaps all within one source file. As a result, when changes have to be made or when bugs need to be corrected, the programmer can focus on that one part.

- Many bugs in security software are caused by a difference in how conditions are interpreted by the security code versus how these are handled in the application. This difference is for instance manifested in the

[7]A similar separation of these tasks has been described in [16].

CVE class of vulnerabilities named exceptional condition handling. The aspect-oriented approach makes the interactions between the application and the security mechanisms explicit such that differences are easier detected.

Furthermore, analogously to the way object-oriented languages influenced earlier steps in the design process, we are now seeing new design techniques that deal more explicitly with separation of concerns. Over time, every cell in the table in Figure 1 might be supported.

For our work, the use of the current version of AspectJ (1.0) has also some drawbacks. On the one hand there are some technical issues regarding the current implementation of AspectJ. It is expected that these will be solved in later releases of the tool. On the other hand there are problems that are more fundamental in nature. For this, we suspect at least a redesign of the aspect-oriented tool is needed.

- For each method call that has some security concern, the AspectJ compiler will insert one or more extra calls. Therefore, the generated code is less efficient, and introduces more overhead than a direct implementation would have. Unfortunately, this is the price to pay for the genericity of our approach. However, it is certainly not worse than some other systems discussed in the next section. Building a more complex, but less general aspect combination tool could solve this.
- If not all code in the application can be trusted, one has to be very certain the generated code does not add any security holes. For instance in the case of authorization: it should not be possible for a client to call the end-functionality of a server through some other means in order to circumvent the authorization checks. This means the security implementer has to have a very clear idea of what the aspect tool produces. In this respect, the output of the AspectJ compiler can currently not be trusted yet, because the original functionality is only moved into a new method with a special name. However, this is only a problem if not all source code[8] is under the control of the AspectJ compiler. The fact that AspectJ is not a formally proven language only increases this problem.
- The implementer of the security code still has to have very detailed knowledge of security mechanisms, their strong and weak points, how to implement them. As AspectJ is a generic tool, it does not help the programmer here, apart from providing a better modularization of the problem. However, this is not a particular problem of AspectJ, but rather of our approach to the problem.
- Generalization of aspects is currently still difficult (see also [8]). The limitations of the aspect language and weaver sometimes makes it very hard or even impossible to generalize certain cases. As a result, a completely reusable security framework can not be developed yet.

[8]Current releases of AspectJ can only do the necessary transformation of source code. This means that weaving aspects in compiled code is not possible yet.

F Related work

There are already a large number of security architectures proposed or implemented in Java, e.g. [7]. Also, Sun has released JAAS [15] for authentication and authorization, SSE for secured network communication, and there are proposals for a secure RMI implementation. These will often already realize the intended result, and can therefore be used in the implementation of the security aspects. The combination of existing technologies with aspect-oriented programming is not expected to pose severe problems. The added value of aspects in this case is the possibility to have a much more flexible security policy, and this at a granularity that corresponds better with the application, i.e. at the level of method calls and objects. Some of the proposed architectures also have a fine granularity, but the configuration and mapping onto what happens inside an application can be fairly difficult.

By using a number of object-oriented design patterns [11], the existing security architectures also try to be independent of an application structure, and they all succeed in this to some degree. The drawback of this design is that the structure of the solution becomes more complex and harder to understand. With an aspect-oriented approach these implementations can be designed in a more natural way.

Transformations in AspectJ happen on the level of source code. One could argue there already exist numerous tools to manipulate text files. These are also able to insert code in a generic way into a program. However, the aspect-oriented approach has much less chance of introducing bugs. The constructs aspect-oriented transformers work on, are not mere text elements, but language constructs. These map more naturally onto the entities a security policy would speak about.

Other tools are available that work on the level of byte code [6,14]. This has the advantage that one can add his own aspects even when no source code is available for the application. Again, the disadvantage is that on the level of byte code, a lot of the application logic is already lost. Reconstructing this is often hard, and giving correct descriptions of how a series of byte codes has to be changed to for instance implement authentication will be even harder. Checking and debugging the result will also be difficult.

Meta level architectures [5,16,17] also make it possible to separate application from security implementation [4,19]. They offer a complete reification of what is going on in the application: the events of sending a message, starting the execution, creating an object all get reified into a first class object. Because the meta-program has control over these reified entities, it can intervene in the execution of the base application. In comparison to aspect-oriented programming this mechanism is much more powerful, but it is also heavier. Moreover, the development of meta-programs for security is more complex, because the programmer is forced to think in terms of meta-elements, which are only indirectly related to the application.

Other approaches [10] also use the basic idea of introducing an interceptor between clients and services, for instance to do extra access control checks. They are similar to meta-level architectures in that they also intervene in the communication between client and service, but the intervention is less generic (and less heavy): the interceptors are mere decorators around the services. In simple situations, they can be specified fairly easy, perhaps through some declarative description. However, when more and more application state needs to be taken into account, writing decorators becomes very hard, or even impossible due to the bounded possibilities of the declarative language.

There is also research into a more generic, declarative description of security properties for an application [9,13]. Such a language would be especially valuable to define the relation between application and security mechanisms as illustrated by the middle column of figure 1. The real challenge here is to think of the right abstractions the description will consist of. This is not at all an evident matter, certainly if a goal is to be generic. We think it is better to first experiment with a generic aspect-oriented language as described in this paper. From these experiments, we would hope to distill the important abstractions.

G Summary

This paper presented the use of aspect-oriented programming to add security to an application. By means of the example of access control, we first demonstrated the feasibility of this approach. In order to construct a more generic solution, we suggested to abstract relevant pointcuts out of the aspect implementation. This enabled us to separate the security mechanisms from the actual policy, which promotes the reuse of the mechanism implementations. After briefly discussing some other security requirements, we touched upon the feasibility to build a security aspect framework. Finally, we discussed the advantages and disadvantages of our approach.

The most important advantage of this approach is the separation of the application and the security related code. This considerably simplifies the job of the application programmer. Moreover, the security policies are gathered in one place, which makes it easier to check whether all the requirements are met. Still, we think that the deployment of these generalized aspects remains quite difficult. We would like to focus our research in the future on this issue, for example by automating the generation of concrete pointcuts based on a simplified high level description.

Acknowledgement

We would like to thank Frank Piessens for his valuable input on how to improve the presentation of our work.

References

[1] International standard ISO IS 15408 common criteria for information technology security evaluation (parts 1-3), version 2.1, September 2000.

[2] AspectJ Website. http://www.aspectj.org/, 2001.

[3] Common Vulnerabilities and Exposures. http://www.cve.mitre.org/, 2001.

[4] M. Ancona, W. Cazzola, and E. Fernandez. Reflective Authorization Systems: Possibilities, Benifits and Drawbacks. In *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, 1999.

[5] S. Chiba. A MetaObject Protocol for C++. In *Proceedings of the 1995 Conference on Object-Oriented Programming*, 1995.

[6] S. Cohen, J. Chase, and D. Kaminsky. Automatic Program Transformation with JOIE. In *Proceedings of the 1998 USENIX Annual Technical Symposium*, 1998.

[7] B. De Win, J. Van den Bergh, F. Matthijs, B. De Decker, and W. Joosen. A Security Architecture for Electronic Commerce Applications. In *Information Security for Global Information Infrastructures*, pages 491–500. IFIP TC11, Kluwer Academic Publishers, 2000.

[8] Bart De Win, Bart Vanhaute, and Bart De Decker. Towards an open weaving process. In K. De Volder, M. Glandrup, S. Clarke, and R. Filman, editors, *Workshop on Advanced Separation of Concerns in Object-Oriented Systems*, pages 1–6, 2001.

[9] D. Evans and A. Twyman. Flexible Policy-Directed Code Safety. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.

[10] T. Fraser, L. Badger, and M. Feldman. Hardening COTS Software with Generic Software Wrappers. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Longman, 1994.

[12] L. Gong. Java Security Architecture. http://java.sun.com/security, 1998.

[13] D. Hagimont and L. Ismail. A Protection Scheme for Mobile Agents on Java. In *Proceedings of the International Conference on Mobile Computing and Networking*, 1997.

[14] R. Keller and U. Holzle. Binary Code Adaptation. In *Proceeding of the 1998 European Conference on Object-Oriented Programming*, 1998.

[15] C. Lai, L. Gong, A. Nadalin, and R. Schemers. User Authentication and Authorization in the Java Platform. In *Proceedings of the 15th Annual Computer Security Applications Conference*, 1999.

[16] Bert Robben, Bart Vanhaute, Wouter Joosen, and Pierre Verbaeten. Non-functional Policies. In P. Cointe, editor, *Meta-Level Architectures and Reflection*, volume 1616 of *Lecture Notes in Computer Science*, pages 74–92. Springer-Verlag, 1999.

[17] R. Stroud and Z. Wue. Using Metaobject Protocols to Satifsy Non-functional Requirements. In *Advances in Object-Oriented Metalevel Architectures and Reflection*, 1996.

[18] B. Vanhaute, B. De Win, and B. De Decker. Building Frameworks in AspectJ. ECOOP2001 Workshop on Advanced Separation of Concerns, 2001.

[19] I. Welch and R. Stroud. Using Reflection as a Mechanism for Enforcing Security Policies in Mobile Code. In *Proceedings of the Sixth European Symposium on Research in Computer Security*, 2000.

Problems in practical use of electronic signatures

Jaroslav Janáček and Richard Ostertág
Department of Computer Science,
Faculty of Mathematics, Physics and Informatics, Comenius University,
Mlynská dolina, 842 48 Bratislava, Slovak Republic
{janacek,ostertag}@dcs.fmph.uniba.sk

Keywords: electronic signature, digital signature, key management, price vs. security

Received: 23 January 2002

The boom of electronic commerce requires existence and use of reliable means of authentication of communicating parties. We need an analogy of hand–written signature that can be applied to electronic documents. We shall call such a concept an electronic signature. There is a current trend to make electronic signatures meeting some requirements equal to hand-written signatures. This makes it a very powerful tool but, on the other hand, there are some sensitive security issues that have to be discussed and that the potential users should be aware of.

In this article we would like to point out some of these issues and especially those that all electronic signature users are affected by. We have to say, it is very unwise to use a software only implementation on a non-dedicated hardware to create electronic signatures if the responsibility for such signature is not limited. Using a hardware cryptographic module that is incapable of displaying the data to be signed may still be unsatisfactory. Using a module without an independent input for authorization data does not help much, a module with an independent input is significantly better but the danger is still far from negligible. Using a dedicated system to create electronic signatures can solve many of the problems. On the other hand, it is the most expensive solution considered. We will describe a solution that is both relatively cheap and reasonably secure. We also suggest some affordable key management solutions.

A Introduction

The boom of electronic commerce requires existence and use of reliable means of authentication of communicating parties (although some cases of electronic commerce are certainly possible without it, there are many that are not). We need an analogy of hand–written signature that can be applied to electronic documents, and that meets at least the following requirements:

1. it can only be created by the entity it represents,
2. it allows the recipient of a document with this analogy of signature to verify that the document has not been subsequently modified (this includes the requirement that the signature analogy cannot be copied from one document to another),
3. it is capable of identifying the entity it represents.

We shall call such a concept an *electronic signature*. Cryptography offers means of *digital signatures* that can be (and are) used to achieve this goal.

Digital signatures are based on asymmetric cryptographic algorithms. They use a pair of keys – a private key and a public key. The private key has to be known only to the entity that is to create its digital signatures, the public key must be known to any entity that is to be able to verify the digital signatures. In an ideal solution it would be impossible to derive the private key from the corresponding public key. In the existing solutions there is no known algorithm for deriving the private key from the corresponding public key in a reasonable amount of time or space (i.e. in time that is polynomial with respect to the key length). A digital signature is a piece of information calculated from a document and a private key. Using the public key corresponding to the private key used to create the digital signature it is possible to verify whether a given document is identical (more precisely identical with high probability) to the document from which the digital signature has been calculated. On the assumption the private keys are only known to their owners and the public keys are known to all intended recipients of documents, the digital signatures meet the requirements 1 and 2 stated above.

To meet the requirement 3 we need to establish a trusted link between a public key and the owner of the corresponding private key. This is being achieved using *certificates*. A certificate is a digitally signed document issued by a trusted third party – *Certification Authority (CA)* that binds a public key to some sort of identity of its owner.

Electronic signatures have been used for a while now in many systems. Their use has been typically based on an agreement of the users of the system. There is a current trend to make electronic signatures meeting some requirements equal to hand-written signatures. This makes it a very powerful tool for electronic commerce and other electronic communication but, on the other hand, there are some sensitive security issues that have to be discussed and that the potential users should be aware of. In this article we would like to point out some of these issues and especially those that all electronic signature users are affected by. We will discuss several possible implementations and try to identify their advantages and disadvantages.

By the term *signatory* we mean the person who is responsible for creation of an electronic signature. We will assume a legislation that makes a signatory responsible for any electronic signature created using the signatory's private key while the signatory's certificate containing the corresponding public key is valid. We will assume that a signatory can make his or her certificate invalid by requesting its revocation at any time.

B What do you really sign?

Let us assume that a signatory wants to electronically sign a document. The signatory's private key has to be stored somewhere and the signatory will use some hardware and software to create the signature (unless he or she is an experienced and fanatic mathematician who wants to spend his or her life calculating the signature by hand).

B.1 Software only implementation on a non-dedicated system

Let us assume the signatory does not have any specialized or dedicated hardware to support electronic signature creation. He or she uses a standard computer with a standard operating system and an application to create electronic signatures. Other applications are also used on the computer and it is potentially connected to the Internet. The private key is stored in a file on a diskette, CD or on the computer's hard disk. In order to protect the private key against other person's access it will be encrypted using a passphrase the signatory will remember (let us assume the encryption used is strong enough and the passphrase is hard to guess).

The first step of signing a document is usually reading it. In case of an electronic document it usually means using an appropriate viewer to display the document in a human readable form. But does the viewer always display all relevant information? (Let us consider the type of the document to be known, so that we can exclude different interpretation of the data than expected.) Is there nothing that remains hidden until you activate some function of the viewer? For less experienced users this can be a potential danger. For this reason it is safer to sign only documents of simple types (like plain text file) or, if you have to sign a complex document (e.g. MS Word), clearly specify what information is to be considered relevant by the recipient (such as only the text that is displayed when using a default installation of a particular viewer).

The next step would be to use a signature creation application to create the electronic signature for the document. It will need the signatory's private key, so it will read it from the file, the signatory will type his or her passphrase, the application will decrypt the private key and use it to calculate the electronic signature. The question here is whether the signature creation application cannot use the private key to sign another document (that the signatory is not aware of) or even export the private key to someone else. Even if you trust the particular application vendor, can you be sure it has not been modified by someone else? Or there may be a bug in the application that allows an attacker to make it misbehave. For example a bug in PGP has been dicovered that allowed an attacker to obtain the user's private key from a message signed using a corrupted private key file [14]. We have assumed a non-dedicated computer system in this subsection. Can you be sure that none of the software you have installed contains a malicious code to modify your signature creation application? If someone else has access to your computer or if it is connected to the Internet, it is hardly possible to be aware of everything that has happened to it. Standard operating systems have many security problems and it is often difficult even for experienced systems administrators to detect sophisticated attacks and modifications. In some cases it may be impossible without connecting the computer's hard disk to another computer (because the firmware of the computer could have been modified to prevent loading of an unaffected operating system from another medium). It would seem, at the first sight, that using a signature creation application loaded from a physically read-only medium should help. But if the operating system has been modified, it is possible to load a different application without the user knowing about it. And it is even not necessary to modify the signature creation application, it is enough to monitor the keystrokes and obtain the signatory's passphrase.

To conclude this subsection we have to say, it is very unwise to use a software only implementation on a non-dedicated hardware to create electronic signatures if the responsibility for such signature is not limited to a reasonable amount (what is a reasonable amount is up to you). The only advantage of this approach is the price – you do not need any special hardware, the only additional software you need is the signature creation application (and this is often integrated in e-mail clients or available for free). But the security risk is very high.

B.2 Using a hardware cryptographic module

Another solution, often prefered, includes the use of a specialized hardware cryptographic module to store and use the private key. It ranges from a cryptographic chip-card to complex, tamper-proof devices. An example of security requirements for a secure signature creation device can be found in [2]. A memory-only chip card does not belong here, it is just another type of storage device and all the problems from the previous subsection apply to it. We assume the cryptographic module is connected to a non-dedicated computer system (just as described in the previous subsection). The module contains the private key stored in itself and does not allow the key to be exported. We can therefore assume no-one is able to get a copy of the private key. The module is usually protected against use by an unauthorized user by requesting some authorization data (a passphrase or a PIN) to be entered by the user before a signature can be created. The first issue is whether the authorization data should be required for each signing operation or whether an unspecified number of documents can be signed after a single entry of the authorization data. In case the number of documents to be signed is not regulated it opens possibilities for signing documents the user is not aware of.

Another issue is how the authorization data are entered. In case of a chip-card it is usually through the computer. And, as we mention in the previous subsection, the computer's operating system may be modified or an application may be running to monitor the keystrokes. Examples of attacks against several electronic signature creation applications can be found in [13]. The attacker cannot get hold of the private key but he or she could instruct the cryptographic module to sign an arbitrary document as long as the module is connected and ready. If the cryptographic module has its own means of input for the authorization data the situation is better – if it requires an authorization of each signing operation it prevents signing without the signatory's knowledge.

However, as long as the cryptographic module cannot display the document it is about to sign, there is still a great potential for an attacker to have an arbitrary document signed. All the attacker has to do is to arrange for his or her document to be sent to the cryptographic module instead of the document the signatory wanted to sign. The module will require the signatory to enter the authorization data and will sign the attacker's document. Well, the signatory's document will not be signed after this, but how will he or she find that out? The attacker's software can arrange for something pretending to be a valid signature to be created and the software to check the signature may be modified to acknowledge it. When the document with the pretended signature is checked in another computer (or perhaps using an application the attacker does not know about), it will be found out the signature is invalid, but it may be too late.

We have shown in this subsection that using a hardware cryptographic module that is incapable of displaying the data to be signed may still be unsatisfactory when connected to a non-dedicated system. Using a module without an independent input for authorization data does not help much, a module with an independent input is significantly better but the danger is still far from negligible.

B.3 Using a dedicated system

As can be seen from the previous subsections, the remaining problem is mainly the possibility of modification of a document between the signatory's viewing it and the signature calculation. What is needed is a system that the signatory can (reasonably) trust and that is capable of carying out the whole process of displaying the document, obtaining authorization data from the signatory, obtaining the private key and calculating the electronic signature. It should be impossible or difficult enough to modify the system or change its operation. This leads to another solution – a dedicated system, i.e. a system (hardware and software) that is used only for a specific task(s), such as the electronic signature creation. It can be a specialized hardware with its own display and keyboard or it can even be a standard computer (perhaps, supplemented with a cryptographic module from the previous subsection). In the case that a standard computer is to be used, only the necessary software should be installed and network connection should be avoided or limited to a very simple interface for receiving the documents and sending out the signatures. The software has to be trusted (we will discuss this issue later). It is also necessary to protect the computer from unauthorized physical access. It may be wise to consider placing it into a tamper-proof box. If there is more than one person who can gain access to the computer, the software (including the operating system) should be installed on a read-only medium to prevent modifications.

Using a dedicated system to create electronic signatures can solve many of the problems we have dicussed. On the other hand, it is the most expensive solution we have considered. While the cost of the software only implementation on a non-dedicated system may be negligible (or even zero), the cost of a card reader and a cryptographic chip-card is in the order of 100 EUR[1], the cost of a dedicated computer is in the order of 1000 EUR, the cost of a tamper-proof computer may be even in the order of 10000 EUR.

B.4 Can you trust the hardware and software?

In all the previous subsections we have assumed the availability of hardware and software for signature creation that can be trusted. But what would make you trust a particular product? This is a difficult question that we cannot provide a satisfactory answer to. But if a dedicated system with no network connection is used and the signatures created are stored in a standard format, the system cannot be controlled remotely and cannot export any information. The only possible way to export information would be to incorporate it into the value of the electronic signature. But if this is in a standard format, possibilities of such information leakage are limited.

[1]100 EUR is approx. US $90, therefore the orders of the approximate prices are the same in USD.

B.5 How to use electronic signatures in a home environment?

Typical home users of electronic signatures cannot be expected to spend 10000 EUR on special equipment to use electronic signatures. What should such users do to use electronic signatures and not to do it in a way comparable to signing (in hand) blank sheets of paper and leaving them in front of the door? We will describe a solution that is both relatively cheap and reasonably secure.

This solution uses the same hardware the user uses for other purposes. It can optionally use a hardware cryptographic module to store the private key, e.g. a card reader with a cryptographic chip-card. The user will have a CD with a bootable copy of an operating system and all necessary applications for the signature creation. When the user wants to sign a document, he or she will disconnect all network interfaces and modems, reboot the computer from this CD, use an appropriate viewer from the CD to view the document (can be loaded from the computer's hard disk) and a signature creation application from the CD to create the electronic signature. The signature will be saved in a standard format back to the computer's hard disk. Then the user will shutdown the system, remove the CD, switch the computer off, reconnect any network and modem cables and restart the normal system. While this is certainly less comfortable than doing everything within the normal system, it is significantly more secure. Regarding the issue of trusting the software, there is a possible information exporting channel if the computer's hard disk is used. Another problem, already mentioned, is the possibility of the computer's firmware modification to prevent booting correctly from the CD (but to load a piece of unwanted software before the "safe" operating system is loaded). This possibility exists but exploiting it is rather difficult and some of the new PC's provide for disabling the firmware modification without physical hardware manipulation. If your PC allows this, use it (you will also protect your computer against several viruses that utilize the firmware modification feature).

To conclude this section we will suggest a few general rules to follow when using electronic signatures:

- avoid signing complex documents (if you are not sure what they really contain)
- if you have to sign a complex document, specify details regarding its interpretation – such as a particular configuration of a viewer that is to be used
- if you cannot use a dedicated secure system, use the procedure of booting a separate operating system from a medium that is not accessible during normal computer's operation (such as described in this subsection)
- if the appropriate legislation allows you to limit your liability for damages caused with respect to electronic signatures, make use of it (i.e. if you can have a key to create signatures with unlimited responsibility and a key to create signatures with your liability limited to 500 EUR, do not use the former to sign an order of goods with the price of 100 EUR)

C Key management

In the life cycle of a key pair, four main stages can be identified: key generation, key storage, key usage and key disposal. Each of these stages brings its own problems, which we should be aware of. Then these problems ought to be avoided in practice or solved to some acceptable level. Some of them will be mentioned in following subsections.

C.1 Key generation, storage and usage

During the key pair generation we ought to have following points on mind: strength, uniqueness and confidentiality (which have to be maintained all the time) of the newly generated private key. Users can choose if they will generate their key pairs or if they yield this generation to another party, e.g. certification authority. Each of these approaches has its own pros and cons.

The main advantage of key generation by user is high level of user's control over this process. Nevertheless full control is hard to achieve. Even if the key pair is generated by the user, it is unlikely, that every step is done by himself or using user's designed software on user's designed hardware. That is why, even in this case, the user have to trust some software and hardware on which this software is running. On the other side, the user can isolate this system from other parties, test it at will, and in other ways minimize the level of unavoidable trust. Beside of design flaws, incorrect implementations, and other unintended mistakes, it is also necessary to consider intended attacks. In this spirit we have already discussed trustworthiness of ordinary PCs and their operating systems. From that it is clear, that a security-oriented user must turn his or her attention to some kind of specialized devices, for example smart cards.

The user does not have an opportunity to study the internal architecture and implementation of a particular device. He or she can only check if the communication protocol is met, if the generation produces a valid key pair, if enciphering and deciphering is correct and so on. In this way it is very hard to verify the quality of the device, for example the quality of generated keys (e.g. randomness of used random or pseudorandom generator). Even more, if there officially is no possibility to retrieve the private key from the card using the implemented communication protocol, there may be "secret unofficial" ways to retrieve it. For example, if randomised cryptography is implemented on a smart card, then every plaintext has more than one corresponding ciphertext. Then the smart card can gradually export secret information. One of the possible methods how to achieve this, is based on the idea, that the user will not notice, if the ciphertext is not chosen randomly among all corresponding ciphertext. We can use the least significant bit (LSB) of ciphertext to divide them to two nearly equal sized sets.

Then if the smart card wants to secretly transmit one bit e.g. zero, it will choose a ciphertext randomly from the set of ciphertext with zero LSB. If bit one has to be secretly transmitted, then the ciphertext is analogously chosen randomly among ciphertexts with LSB equal to one. In this way any secret information from the smart card can leak. Of course one bit is not enough, but the card can cyclically transmit bit by bit from, e.g. the private key. The user can easily detect, that something is wrong in this solution, because if he or she extracts the sequence of LSB from the sequence of produced ciphertexts, then this sequence will be periodical. Using more clever strategies to hide information it is possible to avoid this. For example, a smart card may use a strong cryptographic hash function with secret key to divide the set of all corresponding ciphertexts instead of LSB. Such hidden communication channels are called *covert cannels*. Chances to detect advanced covert channels using black box testing are negligible.

An attack can be driven not only against the cryptographic strength of the implemented algorithms, but also against the very way of their implementation. An attacker can employ timing or power consumption measurement and derive the secret information from this. For example, execution of different arithmetic operations may result in different levels of power consumption. Which arithmetic instructions are executed can depend on the bits of the private key, and so on. These attacks are called differential power attacks and timing attacks. For more in depth discussion see [4, 3, 7].

It follows that an ordinary customer using a crypto card must also trust the card. When the user is deciding which card to buy, he or she will probably choose some card from a trustworthy and reputable manufacturer to ensure its maximum conformity to actual knowledge in the field. But how can the user ensure himself that the card is genuine and not some forgery probably with "enhanced" functionality (e.g. covert cannel implemented). One possible solution is to include in all authentic cards a private key for this brand of cards. If those cards are secure, no attacker can retrieve the private key and thus he or she cannot insert it into a forged card. This card then cannot authenticate itself and is refused. A smart card, beside of creating digital signature, enciphering, deciphering, ..., can also generate a key pair, that is then used inside it. This solution has the advantage that the private information never leaves the smart card. As a result the security of the private information is greatly increased and the user does not need to evaluate the process of transferring it from a generator to the device. To protect devices with stored private key from theft and consequential misuse to generate false digital signatures in name of legal user, it is essential to equip such devices with some kind of authentication (e.g. PIN) before it start to operate.

Yielding the generation to another party is often motivated by better possibilities of this party to generate a strong key pair. However, high level of the user's trust to this party is required, because it is improbable that the user can influence or supervise the generation process here in any way. Certification authorities often provide this kind of service. CA needs key generation for its own use and the large number of clients allow the CA to buy and use excellent software and hardware equipment that is often also too expensive (especially for typical user). Apart from that, CA cannot assume that its technical personnel can be trusted and should use tamper-proof devices. This will guarantee to the CA, and also to the customers, that the generated keys are kept secret even if the operator tries to misuse the generating device (or such manipulation is detected and the key is thrown away). One of such measures can be mutual authentication of the generating device and the crypto card receiving the generated key pair. In this way it is impossible for the operators to insert a false key into the user's device from a fake generator.

Such measures increase the user's level of trust, that the CA has not made a copy of his or her private key (e.g. because the CA is unable to do it). But he or she should be also ensured, that CA has not used his or her private key already. Thus the crypto card must have a feature by which the user can detect its first use.

Existence of two identical key pairs assigned to different customers can lead to serious security consequences. If one of them realizes that they have the same public key, he or she can then easily conclude that they also have the same private key. This fact should be considered a compromise of the private keys of both users, and both users' certificates for the corresponding public keys should be revoked. CA should guarantee uniqueness of the generated keys (or keys with their certificate) at least among keys that CA manages.

More detailed information about requirements on CA practice, management, operation, PKI - key management life cycle and other related topics can be found in [10, 11].

C.2 Key lifetime and disposal

Every cryptographic algorithm, which is not absolutely secure (in Shanon sense), can be successfully defeated by a brute force attack[2]. It is only a question of time. This time can be used as the upper bound on the time necessary for a successful attack. When making practical estimates it is also necessary to consider better attacks that decrease the time needed for a successful attack (e.g. for survey of the attacks on RSA see [5]). Mathematical lower bound would be more appropriate, but if a brute force attack is possible, then we can have luck and hit the right key on the first try. Also such estimate must take the progress in cryptography and technology into account, which give us better attacks on faster computers with more memory.

In [6] we can find the following key size estimates for RSA, based on historical factoring records: $s = 4.23 \cdot (y - 1970) + 23$. Key of size s became publicly breakable in the year y. Standards such as ANSI X9.30, X9.31, X9.42, and

[2]More precisely also some cryptographic algorithms, which are not absolutely secure, can be hard to attack, because for too many different keys we can receive different and correctly looking plaintext - so called ideal security.

X9.44 and FIPS 186-2 all require a minimum of a 1024-bit RSA or Diffie-Hellman key. Based on more elaborate estimate from [6] 1024 key should be good enough until 2020. In the following table the key sizes for different cryptographic algorithms that can be considered equivalent (attacker need the same amount of time and other resources of equal price to break them) are shown. For more details see [6]. Another study of key lenghts can be found in [9].

Key		
Symmetric	Elliptic Curve	RSA
56	112	430
80	160	760
96	192	1020
128	256	1620

When the end of key lifetime period arrives, the key should not be used any more, because there is a high risk, that a successful attack has already been done. After such break, the attacker can emit false signatures, what must be avoided. To guarantee discontinuity of the key use, also the key certificate should expire simultaneously or before the end of the key lifetime.

After that, attackers may be able to create false digital signatures, but they will be useless to them because the public key certificate will already be invalid at that time. On the other hand, they can create valid signatures, if they can backdate signed documents. This can be avoided for example by strict usage of time stamps[3] on all legally signed documents by this key pair.

Disposal of keys is sometimes done much earlier than at the end of the estimated lifetime. For example, if we want higher level of security, we can dispose keys every year. Then it is also necessary to trustworthily destroy this information. Department of Defense in [8] push around some procedures for erasing classified information. For example when data (e.g. a private key) on a floppy disk should be cleared, then the floppy disk have to be degaussed with a degausser or all addressable locations must be overwritten with a single character.

C.3 Suggestions for key management in a home environment

It is probable, that many users will choose to minimize the cost of their solution. So they will decide to use an ordinary PC and free software downloaded from the Internet. The Key pair will be generated by this software. They will store their keys on a harddisk or a diskette. This solution, although not optimal, can be made acceptable if the user uses an environment like the one described in the subsection B.5 and keys are stored in encrypted form.

Users with higher priority on security are encouraged to use smart card with built-in key generation and signature creation. In this way the private key is never exported and thus its transition is avoided. It is also reasonable to expect high quality of generated key pair. If a user obtains a key pair on a smart card from a CA, he or she should require the card to be able to indicate its first use.

[3]A time stamp is a document issued and signed by a trusted third party confirming existence of another document at a particular time.

Conclusion

Considering electronic signature legislation being prepared in many countries of the world we can expect that electronic signatures will become widely used in practise. Not only IT security specialists but also common computer users will come in touch with electronic signatures, therefore it is necessary that these people are informed about the security issues regarding the electronic signature use. It is necessary to find a solution that will be a compromise between a very secure but very expensive one (that can be used by entities that can afford it) and one that is cheap but extremely insecure. We have tried to discuss some of the security issues and suggested a solution that could be suitable for home users. There are many other security issues to discuss, such as the time stamping services, certificate issuing and revocation, certification authorities acreditation, certificate and CRL archiving, PKI structure, different types of certificates and others. Some of them are discussed in [12]. We should mention that similar security problems are also relevant to other security sensitive applications, such as Internet-banking systems, even when they are not based on electronic signatures.

References

[1] Arnd Weber (1998) See What You Sign. Distribution of Risks in Implementations of Digital Signatures, *LNCS 1430*, Springer-Verlag, IS&N'98 Antwerp, pp. 509–520.

[2] CEN/ISSS WS/E-Sign Group F (2001) CEN/ISSS WS/E-Sign Workshop Agreement Group F, *CEN/ISSS WS/E-Sign N 136, 137*, http://www.ict.etsi.org/eessi/EESSI-homepage.htm.

[3] Paul Kocher, Joshua Jaffe, and Benjamin Jun (1999) Differential Power Analysis, *Proceedings of CRYPTO '99*, http://www.cryptography.com/timingattack/.

[4] Paul Kocher (1996) Timing Attacks on Implementations of Diffe-Hellman, RSA, DSS, and Other Systems, *Proceedings of CRYPTO '96*, Springer-Verlag, pp. 104–113, http://www.cryptography.com/dpa/Dpa.pdf.

[5] Dan Boneh (1999) Twenty Years of Attacks on the RSA Cryptosystem, *Notices of the American Mathematical Society (AMS), Vol. 46, No. 2*, pp. 203–213,

http://www.ams.org/
notices/199902/boneh.pdf.

[6] Robert D. Silverman (2000) A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths, *Number 13 – April 2000 – RSA Laboratories Bulletin*,
http://www.rsasecurity.com/
rsalabs/bulletins/bulletin13.html.

[7] Martin Stanek (1998) Útoky na čipové karty, *1. medzinárodná konferencia Bezpečnosť informačných systémov vo finančnom sektore*, Bratislava.

[8] Department of Defense, Department of Energy, Nuclear Regulatory Commission and Central Intelligence Agency (1995) National Industrial Security Program Operating Manual (NISPOM) - DoD 5220.22-M,
http://www.dss.mil/isec/Nispom.pdf.

[9] Daniel Olejár, Martin Stanek (2000) Moderná kryptológia – stav, problémy, perspektívy, *Konferencia Bezpečnosť dát 2000*, Bratislava.

[10] European Telecommunications Standards Institute, Technical Committee Security (SEC) (2000) ETSI TS 101 456 – Policy requirements for certification authorities issuin qualified certificates, V1.1.1
http://www.etsi.org/tbnews/0012_esi.htm.

[11] Bundesamt für Sicherheit in der Informationstechnik (1997) BSI Manual for Digital Signatures – on the basis of the Digital Signature Act (SigG) and the Digital Signature Ordinance (SigV), Version 1.0,
http://www.bsi.de/aufgaben/projekte/
pbdigsig/main/spezi.htm#Manual.

[12] Daniel Olejár (2001) Electronic signature – selected problems of the electronic signature law and of its implementation, *Community Army Technology Environment 2001, International Scientific NATO PfP/PWP Conference, Security and Protection of Information*, Brno.

[13] Adrian Spalka, Armin B. Cremers, Hanno Langweg (2001) The Fairy Tale of ‚What You See is What You Sign‘ – Trojan Horse Attacks on Software for Digital Signatures, *Proceedings of the IFIP WG 9.6/11.7 Working Conference Security & Control of IT in Society – II*, Bratislava.

[14] Vlastimil Klíma, Tomáš Rosa (2001) Attack on Private Signature Keys of the OpenPGP format, PGP programs and other applications compatible with OpenPGP,
`http://www.i.cz/en/pdf/`
`openPGP_attack_ENGvktr.pdf`

Securing Web-based Information Systems: A Model and Implementation Guidelines

C. Margaritis, N. Kolokotronis, P. Papadopoulou and D. Martakos
Department of Informatics and Telecommunications,
National and Kapodistrian University of Athens,
University Campus, 157 71 Athens, Greece
Tel: +3017275225, Fax: +3017275214
h_margar@cc.uoa.gr, {nkolok,peggy,martakos@di.uoa.gr}
AND
P. Kanellis
Andersen
377 Syngrou Ave., 175 64 Athens, Greece
Tel: +3019470275, Fax: +3019425681
panagiotis.kanellis@gr.andersen.com

Keywords: Information Security Strategy, Security Semantics, Web, Systems Development

The decentralised nature of web-based information systems demands a careful evaluation of the pantheon of security issues in order to avoid the potential occurrence of business risks that could not be easily mitigated. This paper presents an integrated approach based on a rigorous multi-level and multi-dimensional model based on the realization that information security is not merely a technical solution implemented at each one of the endpoints of the inter-organizational application. Through synthesis and aiming to contribute towards implementing the most effective security strategy possible, the approach has as a starting point the overall business goals and objectives. Based on those it aids the development of a strategy from the lower levels of securing data in storage and transition to the higher levels of business processes. Its use and applicability is demonstrated over 'Billing Mall' – a system for Electronic Bill Presentment and Payment.

1 Introduction

As organisations are rushing to revamp business models and align operations around e-commerce initiatives, information systems (IS) play a central role in the definition of the new value adding activities. It is without doubt that in the very near future, a large percentage of all commercial activities will be taking place in a virtual world. References [17] and [11] emphasise that such systems must be thought of as 'servicescapes' – enablers of a virtual realm where products and services exist as digital information and can be delivered through information-based channels.

As such, the achievement of strategic goals such as increasing market share will be directly related to the reliability of the technological infrastructure of organizations. It follows that the occurrence of business risks is now more eminent as the corporate network, processes, and critical business data are vulnerable to attacks by anyone having Internet access [1][5][15][16]. What has been observed however is that most organizations treat the Internet simply as a transport medium. The result is that Internet security remains a relatively technical, local and distinct issue from the corporate level IS design and management [15]. We advocate that, as security is the dependent variable for the success of web-based IS, the formation of any information security strategy should begin by taking into account the business vision, goals and objectives. Furthermore, it should not be approached as an afterthought, but rather it has to be designed and evolve concurrently with the development of the system. Any other way to approach this issue could result to a badly designed IS where purposive failure "...quickly leads to massive fraud, system failure, and acrimonious lawsuits" [6]. In summary, the definition of any effective information security strategy should thus be a well planned and concentrated effort initiated at the corporate level, and not be seen only as a local technology issue, or as an ad hoc mix of particular technical solutions to specific problems.

Taking into consideration the above issues, this paper offers an integrated approach for developing and implementing an information security strategy for IS operating in web environments. Based on a comprehensive multi-level and multi-dimensional model, it defines the issues and sets the guidelines for infusing security both at a low and higher level. The section that follows presents the model and its building blocks for aiding the implementation of an effective security strategy. Its application is demonstrated in section 3 over a web-based Electronic Bill Presentment and Payment (EBPP) system developed for the Hellenic Telecommunications Organization (OTE), and currently in its final deployment phases. A concluding discussion closes the article.

2 An Information Security Strategy Model

The use of security models and frameworks has been very much of a specialty area. The assumption that security is largely a technological issue and an afterthought that has to be addressed during a system's implementation phase, may explain the fact that relevant works are absent from the IS literature. However, as [2] notes "...a developmental duality of information systems security exists, that results because the information system and its security are treated as separate developments. This duality may cause conflict and tension between a system and its security". The model that is presented herein was developed taking the above issue under consideration. It acquired an added importance as it was developed during our attempt to define an information security strategy for 'Billing Mall' – a system for on-line bill presentment and payment whose intended users range from corporate customers to households. Realizing that the majority of current and potential Internet users are alert to the security issue through media over-exposure, it was clearly understood that security was a dependent variable for the level of adoption, and subsequently the future success of the system.

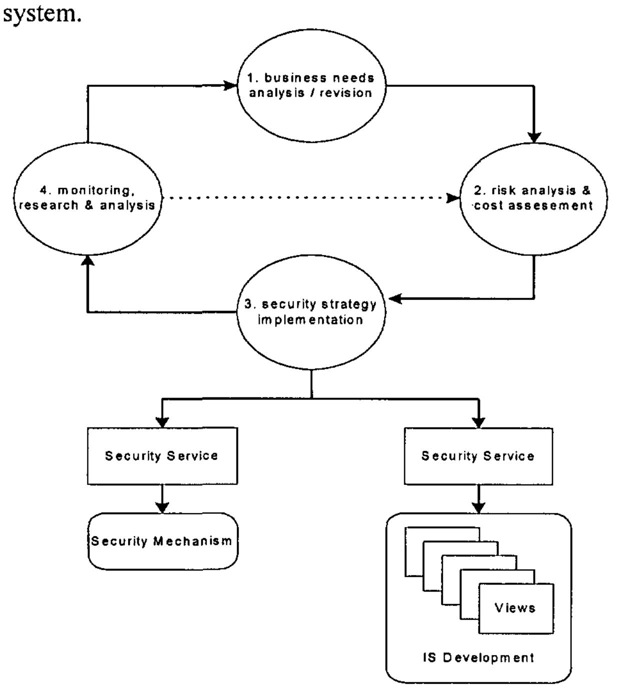

1. business needs analysis / revision
2. risk analysis & cost assesement
3. security strategy implementation
4. monitoring, research & analysis
Security Service
Security Service
Security Mechanism
Views
IS Development

Figure 1: Information security strategy model.

The model, which is depicted in figure 1, portrays a cyclic iterative process for designing and deploying an information security strategy. Each iteration is performed in the context of a phased IS development plan with the emphasis on the various activities and the level of engagement changing from one iteration to the next [9]. Thus, at the early phase of building the information security strategy the focus is on gaining a high-level understanding of the overall security requirements, shifting to weighting risks and costs as more iterations take place and finally to ensuring that the system maintains the quality levels required in meeting business objectives.

The steps identified, namely *Business Needs Analysis*, *Risk Analysis*, *Security Strategy Implementation*, and *Monitoring, Research & Analysis*, are described in the remainder of this section.

2.1 Business Needs Analysis

As already mentioned, security should be examined as an integral part of the overall strategic plan. Thus, any approach to security should start with an analysis of the business needs in order to provide a solid foundation for setting a strategy. Business Needs Analysis is the task of creating and maintaining an IS strategy that correctly reflects the overall mission and goals of the organisation. Understanding business objectives and organizational as well as inter-organizational requirements is fundamental for identifying the security requirements for a web-based IS. Since such a system may surpass the organization's boundaries and extend across multiple organizational entities [18], a deep understanding of business goals at the strategic level is deemed necessary to enable a clear estimation of the demanded security. The analysis starts by identifying the objectives and business activities of the enterprise and its major units. Goals are set for each objective and Critical Success Factors (CSFs) are highlighted to emphasize what must go right if these are to be met.

2.2 Risk Analysis and Cost Assessment

Since the information owned by an organisation is of critical importance, the information resources that are to be protected in terms of their value to the business goals and CSFs identified during the previous step of our model, together with their owners and physical location should be identified. In addition, it has to be specified against whom the previously defined organizational assets should be protected. All these issues have to be considered in conjunction with the cost of deploying the security strategy. Cost assessment will also ensure the provision of management support, an essential part for developing the strategy and a prerequisite for its future application success [15].

The distributed nature of web-based systems implies the existence of a multitude of vulnerabilities and threats that have to be thoroughly examined to guarantee a secure environment for commercial transactions. Potential risks should be identified at all levels, including network services, architecture, operating systems and applications Amongst others, typical business risks include the theft and alteration of data, unauthorised access to sensitive information, inability to meet customer needs quickly and the loss of business. Hence, the purpose of risk analysis is to facilitate decision-making about the desired level of security as well as the methods that should be adopted for preventing risks.

Risk quantification should be undertaken, including a cost assessment of the possible damage associated with each threat against the cost of preventing the threat in terms of time, expenses and resources. The identified risks should then be categorised according to their probability and the severity of their impacts (see figure 2), and prioritised with respect to the cost needed for their elimination. Certainly one needs to consider first those threats resulting in greater losses (classes D and C), but still not to ignore threats of less probable financial impact, occurring more frequently (class B). In completing this step the organisation should be able to outline strategic security objectives. These are general objectives, which may be defined for instance, in terms of the levels of confidentiality, integrity, availability and accountability that the enterprise wishes to attain. The objectives along with a set of rules and practices that regulate how assets are administered, protected and distributed should be described in detail in a Corporate Information Security Policy (CISP) document.

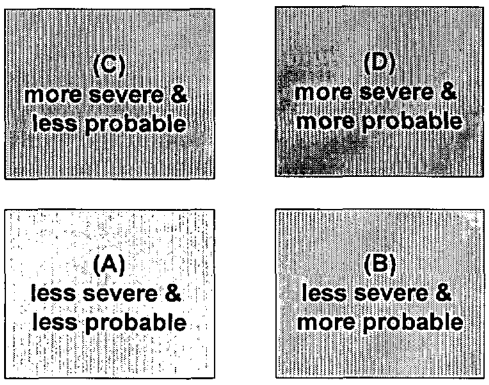


Figure 2: Risk Classification

2.3 Security Strategy Implementation

When risk analysis is completed, the next step is to implement the organisation's information security strategy. The strategy should aim to ensure the most effective use of resources, and will, where appropriate constitute a consistent approach to security across a range of different systems. The influence that the implementation step exerts upon the overall systems development lifecycle is twofold as the information gathered during the previous steps is utilised to (a) identify security services that should be offered by technical infrastructure components, such as standard protocols and commercial off-the self products; and (b) extend the system analysis and design tasks by infusing the security semantics of business processes.

2.4 Identifying Security Services

The identification of security services involves the selection of those that need to be provided by the technical infrastructure in order to protect the organization's information assets from known and unknown threats (see figure 1). The process includes the analysis of a plethora of commercially available products and protocols by examining relevant industry reports and best practice guides.

Undoubtedly, this is the most difficult part of the security strategy development plan, since this step involves the identification of the security services that need to be offered in order to protect the organization's information assets from known and unknown threats (see figure 1). The process includes the analysis of a plethora of commercially available products and protocols by examining relevant industry reports and best practice guides. Not all security services are used for the protection of all kinds of information resources, since different classes of data require different levels of security. Classes of security services include integrity, confidentiality, authentication, accountability and auditing, authorisation, availability, and non-repudiation. In order to provide these security services to a web-based IS, we have to consider (a) the security mechanisms offered for data in transit, and (b) the security mechanisms offered for data in storage. These are illustrated in tables 1 and 2 respectively.

When data in transit is considered (table 1), protocols offering security services are divided into three main categories depending on the International Standards Organisation's (www.iso.ch) Open Systems Interconnection (OSI) layer they operate, namely the network, transport and the application layer. Security mechanisms at the application layer can be further categorized according to the specific structure and nature of the data that needs to be protected.

Layer	Protection	Mechanism
network/ Internet	host-to-host	IP Security (IPSEC), IP Authentication header (AH), IP encapsulating security payload (ESP), network layer security protocol (NLSP), point-to-point tunnelling protocol (PPTP)
transport/ session	process-to-process	secure sockets layer (SSL), transport layer security (TLS), open financial exchange (OFX)
application	data structure-specific	secure hypertext transfer protocol (S-HTTP), pretty good privacy (PGP), privacy enhanced mail (PEM), secure multipurpose Internet mail extensions (S/MIME)
	data nature-specific	secure electronic transactions (SET), open financial exchange (OFX)

Table 1: Mechanisms used to enforce the security policy for data in transit

In general, it is easier to protect corporate assets from third parties outside the corporate network than from its employees who intentionally or accidentally may cause severe security incidents. Thus, it is of crucial importance to ensure that everyone inside the corporate network complies with the corporate security strategy guidelines. This means that security for data in storage does not only depend on the technology used, but also on the proper administration of systems, as well as the observance of related business procedures, physical access controls, and audit functions. Not all business requirements and objectives are identical. Consequently, security mechanisms for data in storage are not absolute - there is not one standard that will fit all businesses and industries. In table 2, we present the dominant mechanisms (hardware/software based) currently available for safeguarding critical data in storage within the organisation.

Type		**Solutions**
Hardware		smart cards (PVC, EMV), other tamper-proof devices, screening routers, biometric devices
software	operating system level	password-based authentication, password expiration and filtering, Kerberos-based distributed authentication, access control lists (ACL), security identifiers (SID)
	database management system level	password expiration, password standards enforcement, break-in detection and evasion, dormant user ID identification, centralised security administration, comprehensive report generation, maintenance of audit logs
	application level	anti-virus software, audit log analysers, firewalls, backup utilities

Table 2: Mechanisms used to enforce the security policy for data in storage

2.5 Defining Security Requirements at Business Process Level

Our discussion thus far has focused on the implementation of a security strategy mainly at the lower infrastructure level. We agree with [2] that a security strategy should evolve concurrently with the design of the system and not be approached as an afterthought. As such, any integrated approach should address also how security could be possibly implemented at a higher level, i.e. the business process level. IS that support business transactions are developed based upon well-defined business process models. A business process is defined by a senior or middle manager – usually with the help of an outside consultant – and contains the following components: information flows between organizational units involved (e.g. business units, departments, agents, etc.), tasks to be performed, information sources and their usage and structure, and behaviour of all the components involved.

In order to arrive at a complete understanding of the security requirements at the business process level, [12] suggested examining a business transaction from at least five different perspectives/views, each one extended accordingly in order to capture the security semantics:

❑ The *business process view* representing the flow of work in terms of activities and participating entities from the viewpoint of the whole business process. It is used both as a means to communicate the architecture of the system to the stakeholders and to guide the modelling efforts for the other four views.

❑ The *informational view* representing the information entities, their structure and any relationships between them.

❑ The *behavioural view* showing what tasks and activities are associated with the various objects, the events that trigger these activities and the message exchanging that occurs between them.

❑ The *dynamic view* representing for each information entity all possible states and any transitions that may occur within the life cycle of the information entity.

❑ The *structural view* showing where and by whom tasks and activities are performed.

The above can guide the analyst towards acquiring a holistic view of any business process – from the highest to the lowest level. We adopt those views – placing them within the 'security strategy implementation' stage of our model in order to capture the security semantics of business processes during the analysis and design tasks. Their practical application is demonstrated in the next section of the paper.

Most existing research in the engineering of secure IS has used formal methods in the context of a conventional process model [3]. In general, a waterfall process works well for systems where requirements and design issues are well understood from the outset [8]. In the past many security critical systems exhibited these characteristics and in these environments conventional formal methods were generally deemed adequate. However, they are much less useful in an environment where security and other design goals may be in conflict [2]. Pressures to compete against smaller or more flexible firms in global marketplaces are mounting. In response, organisations are attempting to achieve new forms that foster rapid adaptation to change. These competitive trends are forcing organisations to develop new forms of IS that are more open and adaptable to changes.

In such an environment, a multi-dimensional approach integrating security semantics with business transaction models offers significant advantages such as the following:

❑ The security ramifications of different design alternatives can be explored before the decision is made to commit to any single one.

❑ A basic verification strategy can be laid out early in the process in order to avoid the unpleasant possibility that a workable design is impossible to verify.

❑ Decisions to bypass security in order to meet other goals are made consciously early in the process, avoiding thus the possibility to be discovered as a result of a security incident much later.

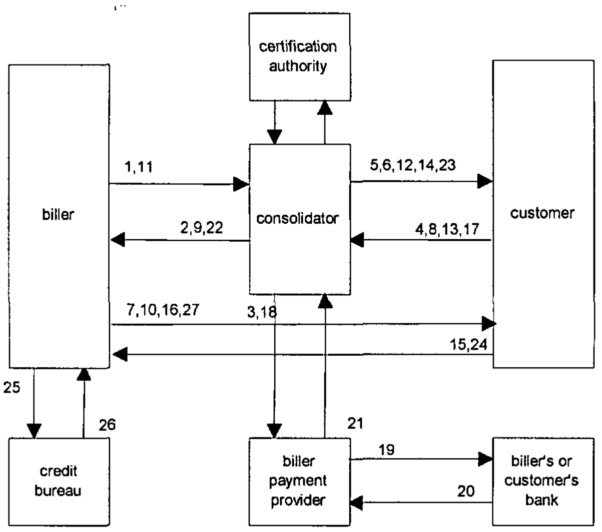
2.6 Monitoring, Research and Analysis

The monitoring, research and analysis step of our model can be performed using both internal and/or external auditors. A plethora of solutions that are available widely by software vendors, such as audit log analysers and intrusion detection mechanisms can provide valuable information regarding potential implementation flaws. Their effective use may pinpoint to the need for further analysis and a re-evaluation of the risk level in the context of 'live' operating conditions (indicated by the broken line connecting steps 4 and 2 in figure 1). This exercise may identify implementation flaws or weaknesses of the current strategy, or serious gaps emanating from a change in the business scope and activities. In the former case the modification of existing security mechanisms or the addition of new ones might be deemed necessary. The latter case may constitute the starting point for radical changes in the security strategy, which have to be adequately addressed during subsequent iterations of the steps in the model. In this section we provided a comprehensive model for aiding the definition and deployment of an information security strategy from a multi-level and multi-dimensional perspective. What follows is a description of how this model was used to define and implement the security strategy of 'Billing Mall' – an EBPP system developed for the Hellenic Telecommunications Organisation (OTE).

3 Information Security Strategy Implementation

The initial response of the market to various commercial applications regarding EBPP systems is indicative of their future potential in becoming contenders for a permanent place in the worldwide Internet infrastructure. According to industry analysis, within 3-5 years the majority of bills will be presented and paid electronically [7]. In the United States alone it is projected that by taking the 'paper' out of the billing process, EBPP could save billers, customers and other constituents over $2 billion annually by 2002 [10]. 'Billing Mall' (http://alexandra.di.uoa.gr) is such a system, offering facilities for bill presentment and payment, customer application processing and personalised marketing (see figure 3). The system provides electronic delivery of bills to customers through the presentment of bill information in both summarised and detailed form, and secures electronic payment of a single or multiple bills upon customer request. Customer Application Processing (CAP) provides the means to customers who wish to order a new product or service that is available by OTE to do so. Finally, Personalised Marketing (PM) offers the necessary functionality and support needed for the effective promotion of products and services based on a customer's identified needs and characteristics.

The architectural model of the system is based on the Open Internet Billing (OIB) [7] model. According to OIB, a central service provider, the Consolidator, collects and stores electronic summary bills from registered billers. While offering a single point of access for viewing and paying bills, it provides the customer with the option to have access to the biller's web site for detailed bill information. When the customer visits the web site requesting to see a detailed bill, the Biller presents him with informative messages regarding products and services available. The customer is also provided with a facility for placing orders for the advertised products and/or services.



1. Biller enrols to consolidator to offer services, *2*. Biller receives certificate from Certification Authority (CA), *3*. Biller Payment Provider (BPP) receives certificate from CA, *4*. Customer enrols to consolidator and selects billers, *5*. Customer receives certificate from CA and login account, *6*. Announcement of new biller participating in EBPP service, *7*. New biller providing EBPP service, *8*. Request for receiving and paying bills from the new biller, *9*. Request for including the new biller in EBPP service is forwarded to biller, *10*. Notification of EBPP service becoming active for customer, *11*. Bill summary is made available to consolidator, *12*. Notification of a new bill made available for viewing and paying, *13*. Customer logs in, *14*. Bill summary is accessed by customer, *15*. Request for accessing detailed bill information, *16*. Detailed bill information and personalised marketing, *17*. Customer initiates bill payment, *18*. Payment request is forwarded to BPP, *19*. Payment execution is originated, *20*. Payment execution is completed, *21*. Notification for completion of payment, *22*. Notification for bill payment execution and remittance information, *23*. Notification for successful execution of bill payment, *24*. Order submission for biller's products and/or services, *25*. Request for information about risk of crediting customer for purchase of ordered products and services, *26*. Information about credit risk associated with customer, *27*. Notification about acceptance or rejection of submitted order.

Figure 3: The 'Billing Mall' Internet Bill Presentment and Payment System

An evaluation of the critical factors for the successful deployment and consequent adoption of the system imposed the need for the parallel development of a comprehensive security strategy. Aiming to guarantee an integrated approach to the multilateral issue of security, the model described in the previous section has served as the basis for the design and implementation of the security strategy.

Following the stages prescribed by the model, a business needs analysis has been conducted first, providing the foundation for the strategy. In this context, business goals and Critical Success Factors (CSFs) were clearly defined, indicating the need for a system guaranteeing secure electronic transactions associated with all types of offered services. In order to mitigate the cost of deploying a secure communication mechanism for financial transactions between the Consolidator and the Banks, it was decided that the existing infrastructure currently in use for fund transfer between financial institutions in Greece should be leveraged. This implied the need for including an additional entity to the OIB model, the Biller Payment Provider (see figure 3), serving as an intermediary between the Consolidator and the Banks.

The next step towards the implementation of the security strategy was to conduct a risk analysis as a proactive diagnosis of the vulnerabilities and threats that could possibly hinder the proper operation of the system and should be effectively addressed during the security strategy implementation phase. To this end, the resources that were to be protected were identified at both organizational and inter-organizational levels, in terms of the information stored, the applications and the hardware used and the underlying network infrastructure. These corporate assets were deemed necessary to be protected from internal as well as external attacks, either intentional or accidental. A number of entity-centric and cross-organizational risks were identified. The results of this process suggested that the potential vulnerabilities and threats should be effectively addressed by carefully selecting and applying risk prevention, detection and response methods. The analysis revealed that the OIB model was not adequate to provide the anticipated level of security and reliability that is essential for the networked business processes. Thus, it was decided that it had to be extended in order to accommodate the establishment of a Certification Authority (CA) issuing and disseminating digital certificates to the customers (see figure 3). Furthermore, as a means for addressing the risk of insolvent customers, issuing payment transactions that could not be completed due to insufficient credit, a Credit Bureau entity was added to the architectural model of the system (see figure 3). The functional role of this entity is the provision of information related to the credit status of customers, eliminating the possibility of financial damage.

The security mechanisms required for managing the identified risks were identified next. Since 'Billing Mall' requires the exchange of large amounts of financial information, the first task was to evaluate the security features of existing protocols in the field. Between Open Financial Exchange (OFX) (www.ofx.net) and Secure Electronic Transaction (SET) (www.setco.org), the former was found more appropriate mainly because (a) it supports the use of channel-level as well as application-level security, and (b) its security architecture is expandable and customisable. The SSL protocol met the requirements defined by the deliverables of the first two steps of the framework for ensuring the confidentiality and the integrity of data in transit. However, some constraints had to be put into place concerning the cryptographic algorithms used, as well as the size of the session key. In contrast to the OFX specification [4], both server and client side certificate-based authentication is required by Billing Mall at channel-level security in order to eliminate security risks. Thus, password encryption is not required as the specification dictates for authenticating the user, who is provided with the additional capability of encrypting vital information inside the OFX message, such as credit card number and/or bank account data, with the OFX server's public key.

For this reason, only one entity satisfying the requirements imposed by the European Community's 1999/93/EC directive on electronic signatures was decided to play the role of the certification authority. The certificates issued by the CA are based on the PKCS #6 extended-certificate syntax standard [13], because of its flexibility in defining new PKCS #9 selected attribute types [14], and its compatibility with applications requiring the use of X.509 certificates. In order to facilitate certificate and key management, from the customer's point of view, smart card technology was decided to be a basic part of the overall design. As far as 'Billing Mall' is concerned, a defensive policy is enforced regarding the amount for which an issued certificate can be used. This limit, which is interpreted as the amount that the user is willing to risk per transaction, is determined by the user and may be accepted or rejected by the CA and the Credit Bureau.

Firewalls, as expected, constitute the first line of defence for all entities (this does not include the Customer) participating in the 'Billing Mall' system. It is suggested that important information should only be accepted from, and delivered to servers with a specific IP address, which means that any network package sent by an unknown IP address is automatically rejected. Example procedures taking advantage of this feature are that (a) the Consolidator only accepts bill summary information from a small set of IP addresses in the Biller's domain, and (b) the Consolidator only forwards Customer's payment requests to the specific BPP IP address. This technique allows some degree of resistance against attacks such as the 'denial of service' attack and IP spoofing.

Our aim during the design of the 'Billing Mall' was that the objectives of the information security strategy had to be integrated in the development process. It is suggested in [12] that in order for this to be achieved, a business transaction must be viewed from multiple perspectives with each view extended by the security semantics of the information security strategy. In the following section we present an example of analysing the different views of the 'BILL-PAYMENT-ORDER' business process (step 17 in figure 3) and its security requirement 'non-repudiation'. In the example we use the following notation: components of existing model or attributes, which are not affected by security requirements, are described using normal text. The attributes with relevance to non-repudiation are given in bold face.

3.1 Business Process View

As required by the European Community's 1999/93EC Directive documents in electronic business transactions must be signed digitally. A digital signature 'seals' the data to be transmitted and is created by the private key of the signatory using asymmetric cryptography. In order to study what effects a digital signature has, we will first refer to the business process view in our example. Business process modelling is typically performed in order to capture the commercially important activities. This can often lead to design conflicts once the security requirements are taken into account. In order to eliminate this tension, the supporting entities and activities that are necessary to realise the system function the way it is envisioned initially, need to be captured as well. Furthermore, since the business process view is used to guide the modelling efforts from many angles, the security semantics of the business transactions are captured in a consistent and integrated manner.

Figures 4 and 5 graphically depict the 'BILL-PAYMENT-ORDER' process using Unified Modelling Language (UML) use case and activity diagrams. The use case diagram in figure 4 depicts the scenarios and actors involved in the business process of our example, while figure 5 shows the activities performed in completing the PayBill use case. In order to meet the "non-repudiation" requirement, our model has been extended by the appropriate actors (Certification Authority, Signature Manager), use cases (Certificate Renewal, Payment Order Archival) and activities (Verify Digital Signature, Verify Certificate Validity).

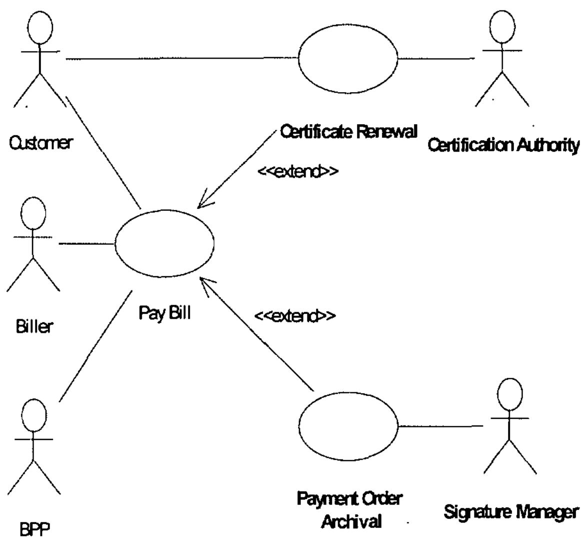


Figure 4: Business Process View extended by security semantics

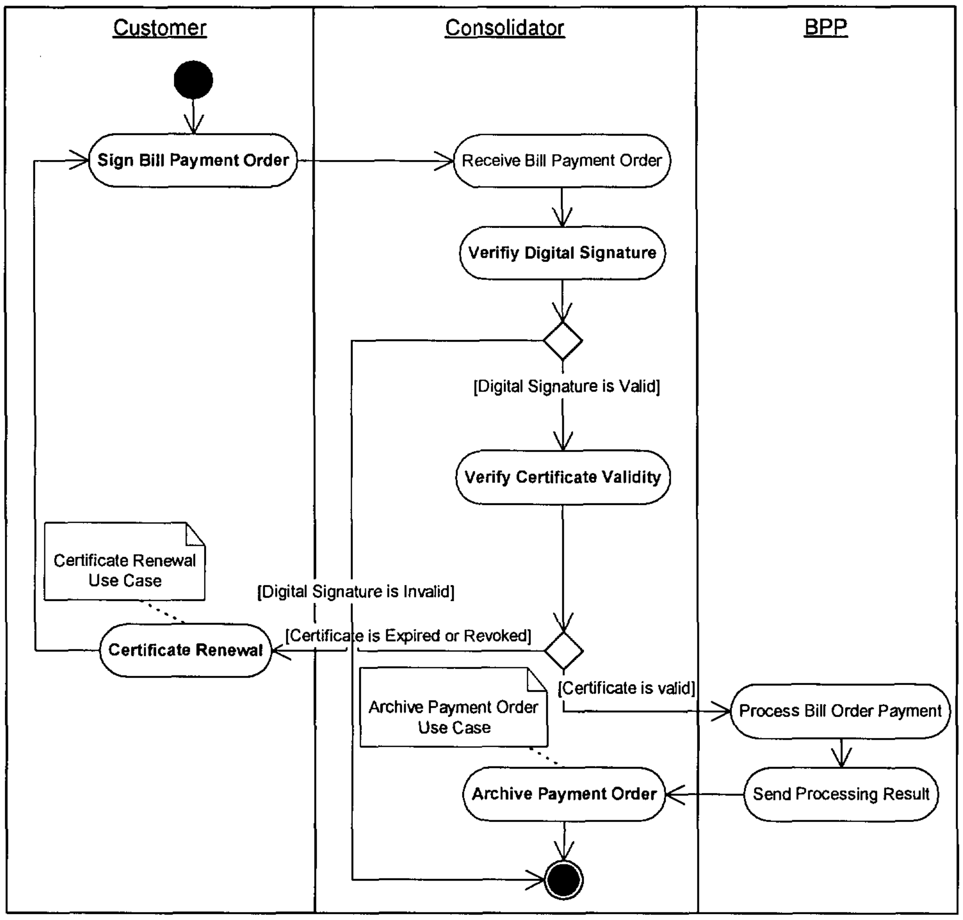

Customer
Consolidator
BPP
Sign Bill Payment Order
Receive Bill Payment Order
Verifiy Digital Signature
[Digital Signature is Valid]
Verify Certificate Validity
Certificate Renewal Use Case
[Digital Signature is Invalid]
Certificate Renewal
[Certificate is Expired or Revoked]
[Certificate is valid]
Archive Payment Order Use Case
Process Bill Order Payment
Archive Payment Order
Send Processing Result

Figure 5: Activity diagram illustrating security semantics of 'Pay Bill'

3.2 Informational View

According to the European Community's 1999/93/EC directive in order to sign an electronic document the 'seal' or digital signature of each signatory and the corresponding certificates are necessary. Accordingly, to effectively carry out the BILL-PAYMENT-ORDER process and to establish non-repudiation, the informational view of the transaction has to be extended by information about the signatories, the certificates used, and the trusted parties (CA) responsible for issuing the certificates. The analysis and modelling can be performed using UML class diagrams. In figure 6 we have extended the class diagram containing the customer-biller relationship of our example by appropriate classes and member fields necessary for supporting non-repudiation. These are:

- a new class CERTIFICATION AUTHORITY
- a new association class CERTIFICATE
- modification of the existing COMMITAL class by adding the appropriate fields for the digital signatures, and information about what algorithms were used for signing. The COMMITAL class is used to model any kind of document that should be signed by a customer (bill payment order), a biller (bill statement) or both (service level agreement).

In addition, customer and biller are specializations of a generic type signer, which must have a certificate relating the signer to a certification authority.

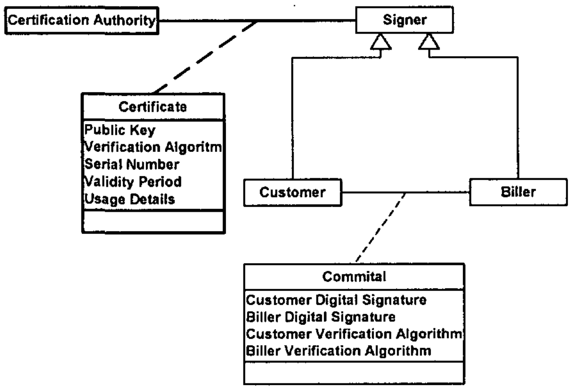


Figure 6: Informational View extended by security semantics

3.3 Behavioural View

The interactions and corresponding information flows between the entities involved in the BILL-PAYMENT-ORDER process can be analysed through the behavioural view. For the modelling of this view, UML sequence diagrams can be used. In order to assure non-repudiation, the behavioural view of the process must be modified as depicted in figure 7.

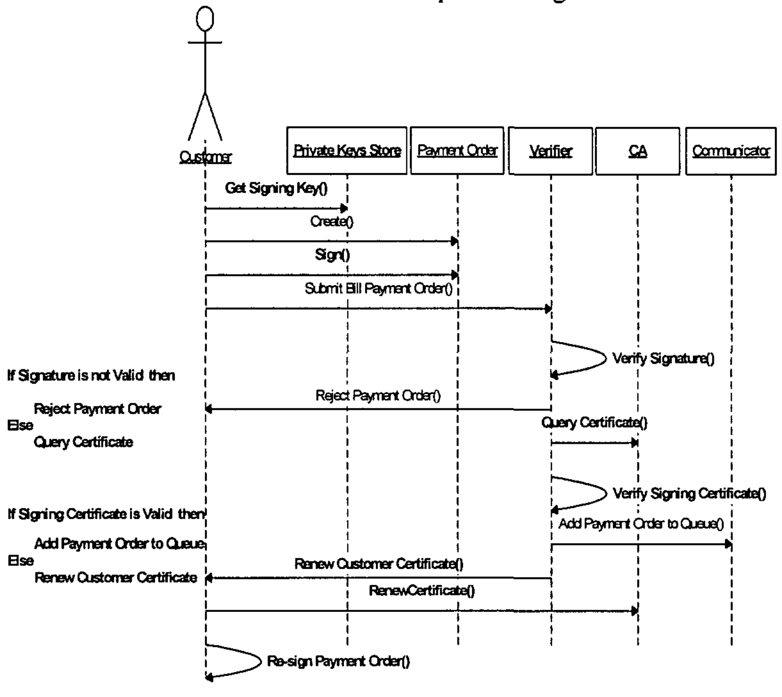


Figure 7: Behavioural View extended by security semantics

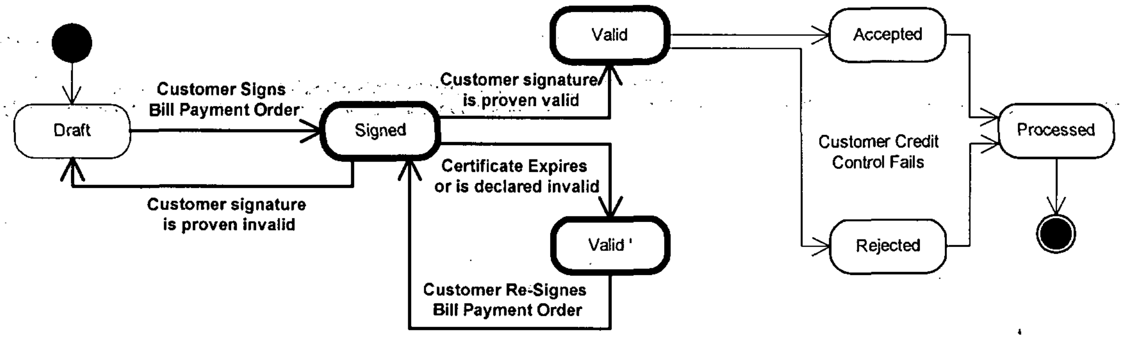

Draft
Customer Signs Bill Payment Order
Signed
Customer signature is proven valid
Valid
Customer signature is proven invalid
Certificate Expires or is declared invalid
Valid '
Customer Re-Signes Bill Payment Order
Accepted
Customer Credit Control Fails
Rejected
Processed

Figure 8: Dynamic View extended by security semantics

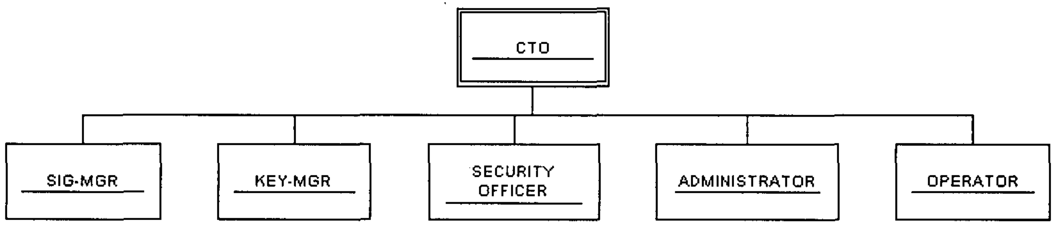


Figure 9: Organizational View extended by security semantics

The customer must digitally sign the bill payment order and the signature must be verified. In addition, because the certificate of a public key may have expired, further actions are necessary to guarantee the provability of digitally signed documents. These actions lead, for example, to extensions of the behavioural view of an object class (Verifier) responsible for validating the integrity and provability of the payment order.

Again, in figure 7 necessary extensions due to security requirements are given in bold face (the sequence diagram has been enhanced by the use of scripts for accommodating complex scenarios involving conditions and iterations).

3.4 Dynamic View

The process of executing a bill payment order and establishing non-repudiation raises a number of security issues emanating from state transitions that various entities undergo. These can be highlighted via an analysis and modelling of the dynamic view. In figure 8 we show the life cycle of the BILL-PAYMENT-ORDER in terms of the participating entities and their different states, using a UML state-chart diagram. As [12] have emphasised in a similar example, the state '*valid*' is important security-wise as it represents an object of type bill payment order, which although signed, the certificate of the signatory is expired or is revoked. In this case this becomes clear, as the payment order must be re-signed.

3.5 Structural View

As expected, 'non-repudiation' affects the structural view as well. Meeting critical security requirements may result in the creation and introduction of new roles with specific responsibilities. Organizational charts have to be modified in order to mirror emerging needs. In this example, a new role (Signature Manager) can be created for an employee whose main responsibility will be to check the validity of archived digital signatures, re-sign documents with certificates that are no longer valid, and monitor in general all activities in this context. Additional roles may be needed for key management (figure 9).

Using the five views for analysing and modelling the security semantics of business processes as proposed initially by [12], the preceding sections offer a summarised view of a single process and the security requirements that had to be infused and performed by the 'Billing Mall'. It becomes clear that by modelling and analysing the security semantics of the business transactions it supports, the IS and its security are not treated as separate developments. As the former becomes part of the design process, the possible duality as a cause for conflict [2] is eliminated.

4 Conclusion

In this paper we presented an integrated approach for the development of an information security strategy based on a rigorous multi-level and multi-dimensional model. The position that any security strategy must evolve concurrently with the design of the system and not be approached as an afterthought is reflected in the model, which (a) monitors closely the development phases of an IS, and (b) addresses security at the business process level. Enabling the practitioner to evaluate and use the available security tools and techniques in a consistent manner, the structure of the model enforces the view that any security strategy must be conducted primarily at a higher level, and not be seen merely as a local technology issue. Without doubt we believe that the approach presented herein could be further refined and enhanced. We hope that its further adoption will possibly result to enhancements and modifications, incrementing thus its value regarding its applicability in commercial contexts. 'Waterproof' security of large inter-organizational systems is an issue of immense complexity, but we believe that we have at least made a few but necessary steps towards meeting this challenge.

Acknowledgement

We gratefully acknowledge the *Greek Secretariat for Research and Technology (GSRT)* for financing the 'Billing Mall' project. We would also like to thank the partners involved in the project: Athens University of Economics and Business, Cyberce, Datamedia, Dias, Sysware, Teiresias and the University of Crete.

References

[1] ABELA, A and J.R SACCONAGHI (1997). Value exchange: The secret of building customer relationships on line. The McKinsey Quarterly , 2, 216 –219.
[2] BASKERVILLE, R. (1993). Information Systems Security Design Methods: Implications for Information Systems Development, ACM Computing Surveys, 25(4), 375-414
[3] BOEHM, B. (1988), A Spiral Model of Software Development, IEEE Computer, May, 61-72.
[4] CHECKFREE Corp., INTUIT Inc. and MICROSOFT Corp. (1998) Open Financial Exchange. Specification 1.5.1.
[5] DERIVION Corp. (1999). Internet Billing and the Mid-Tier Biller: Enjoying the Benefits of Electronic Bill Presentment and Payment without Operational Compromise. Available at http://www.derivion.com/ index9.html
[6] HUGHES, E. (1997). A long-term perspective on electronic commerce. Networker, Nov/Dec, 38 –50.
[7] JUST IN TIME SOLUTIONS Corp. (1999). The Value of Internet Billing. Available at http://www.justintime.com/internetbilling/index.html
[8] KEMMERER, R. A. (1990), Integrating Formal Methods into the Development Process, IEEE Software, Sep, 37-50
[9] KRUCHTEN, P. (2000), The Rational Unified Process – An Introduction, Addison – Wesley.
[10] OUREN, J., M. SINGER, J. STEPHENSON and A. L. WEINBERG (1998). Electronic bill presentment and payment. The McKinsey Quarterly, 4, 98 –106.
[11] PAPADOPOULOU, P., A. TRIANTAFILLAKIS, P. KANELLIS and D. MARTAKOS (2000). A generic framework for the deployment of an Internet billing servicescape. In Proceedings of the 1st World Congress of Electronic Commerce, Hamilton, Ontario, Canada, January 19-21.
[12] RÖHM, A.W., PERNUL, G. and HERRMANN, G. (1998). Modelling secure and fair electronic commerce. In Proceedings of the 14th Annual Computer Security Applications Conference, Scottsdale, AZ., Dec. 7-11, IEEE Computer Society Press.
[13] RSA DATA SECURITY Inc. (1993). PKCS #6: Extended –Certificate Syntax Standard, version 1.5.
[14] RSA DATA SECURITY Inc. (1993). PKCS #9: Selected Attribute Types, version 1.1.
[15] SEGEV, A., J. PORRA and M. ROLDAN (1998). Internet security and the case of bank of America. Communications of the ACM, 41, Oct, 81 –87.
[16] WALKER, K.M. and L.C. CAVANAUGH (1998). Computer security policies and SunScreen firewalls. Sun Microsystems Press.
[17] WANNINGER, L., C. ANDERSON and R. HANSEN (1997). Designing Servicescapes for Electronic Commerce: An Evolutionary Approach. Available at http://www.misrc.umn.edu /wpaper
[18] YANG, J. and PAPAZOGLOU, M.P. (2000). Interoperation Support for Electronic Business. Communications of the ACM, 43, June, 39-47.

Implementing information integrity technology – a feedback control system approach

Vijay V. Mandke
Center for Information Integrity Research
Unitech Systems (I) Pvt. Ltd.,
B-64 (First Floor), Gulmohar Park , New Delhi-110049, INDIA
E-mail: vijaymandke@satyam.net.in
AND
Madhavan K. Nayar
Unitech Systems, Inc.
1240 E. Diehl Road, Suite 300, Naperville, Illinois 60563, USA.
E-mail: mnayar@unitechsys.com

Keywords: integrity technology, feedback approach, control system, usefulness-usability-integrity paradigm, cost-benefit analysis, decision process, integrity attributes, integrity quantifiers, integrity profile, integrity index

Received: February 23, 2002

The paper begins with the question of a structure for integrity objective based on information Usefulness-Usability-Integrity paradigm suggesting criticality of Information Integrity for competitive advantage. If integrity researchers find it difficult to convince business managers of this criticality, it is because of want of a construct of an economic framework for Information Integrity. Thus the paper outlines an approach to cost-benefit analysis of Information Integrity to decide analytically on investing for improved, optimal integrity. This calls for quantifying intrinsic integrity attributes of accuracy, consistency and reliability. Towards this, the paper discusses the choice of information model for integrity improvement, followed by alternatives for quantification of integrity attributes and development of integrity profile and cumulative information integrity index as a means for demonstrating integrity improvement. This is followed by presentation of information integrity technology implementation steps within the framework of feedback control system approach. Finally, the paper details components of a platform to facilitate developing thus emerging information integrity technology as a software product.

1 Introduction

Errors in computerized information systems were relatively manageable as long as there was homogenous system environment and centralized control over information. Emerging trends of globalization, changing organizational patterns, strategic partnering, and electronic commerce and distributed computing have changed all this, resulting in loss of integrity in information systems. These errors are essentially caused by on-line factors of change, complexity, communication, conversion and corruption (the 5 C's). These factors have their presence in IS mainly through system environment that is external to computing (and hence the application) system and overlaps the user environment. In spite of application controls, it is these external factors that then introduce in information systems, errors that are made but not corrected [Mandke and Nayar, 1997; Nayar, 1996].

It is within above framework of errors in computerized information systems and their integrity implications, that research investigations presented at IFIP TC 11 WG 11.5 Second Working Conference [Mandke and Nayar, 1998] identify intrinsic integrity attributes of accuracy, consistency and reliability which, irrespective of nature of use, any IS must satisfy. Research investigations further observe that depending on the context and nature of use there can be other optional integrity attributes of security and privacy that can be seen as extrinsic or subjective integrity attributes specific to area of use [Mandke, 1996]. Other such subjective attributes of integrity could be: interpretability, understandability, tractability, flexibility, etc. [Wang and Strong, 1993].

It is to ensure above integrity attributes that research investigations presented at IFIP TC 11 WG 11.5 Second Working Conference [Mandke and Nayar, 1998] have proposed need to incorporate on-line learning and error correcting mechanisms in the IS models. Specifically, to account for errors in IS that are made but not corrected, they propose incorporation of automatic feedback control systems with error detection and correcting technologies for improved information accuracy, consistency and reliability; technologies that maximize integrity of information systems – Information Integrity (I*I) Technologies. They further argue that, when incorporated, it is such Information Integrity Technology that would also facilitate demonstrating improved integrity of information obtained, rather than merely trusting the computerized information systems.

There are obvious difficulties in implementing such automatic feedback control systems, the most important being to study error patterns. Specifically, it is not possible to track and analyse every bit of data/information for all times as it flows through the information system stages. Way out here is to consider Information Integrity Technology that takes a sample of input data at the output or at an intermediate point of an appropriately identified stage or sub-system of the IS and then follows or keeps track of the sampled records at output or intermediate points of subsequent stages (sub-systems), at a given point of time or at different points of time over a required time interval [Mandke and Nayar, 1997; 1998].

2 Quantifying Information Integrity Attributes: A Basis

This brings forth the central question as to what will be the nature of such an Information Integrity Technology. To answer this, it is first necessary to consider the question of a structure for integrity objective.

2.1 Usefulness-Usability-Integrity paradigm

When one is modelling data and information, it is useful to consider them as integral to a core information system (IS) wherein: (a) "data" is seen as raw material, "processing/transformation/conversion" as the system function, "information or data product" as processed data which is used to trigger certain action or in management or decision or to gain better understanding of what data implies, and (b) the pre- and post- processing communication channels (comprising data communication and particularly distributed transaction processing network) are components of IS [Mandke and Nayar, 1998]. This essentially is a decision process model of an IS in the sense as data *is* processed/transformed/converted to form information *for* use, choices are made among various alternatives and this fact is not limited to IS for management or policy decision but includes all the routine operating decisions that are made at all levels of the supply chain. In fact, even in a very routine and everyday activity, when the information is transmitted without changing form, as might be the case in a telephone system, there *is* a decision to be made as to the objective of the transmission [Mandke and Nayar, 1999; Matthews, 1971].

In the analysis of databases and the integrity objective, on the one hand, this IS- based visualization facilitates a need to view databases along with their data acquisition and information utilization cycles, while on the other hand it requires information always to be identified in the context (i.e., environment) of its objective or goal. In other words, for the study of integrity objective, firstly it becomes meaningful to necessarily take a system's view of Information Integrity (I*I) by going beyond data integrity and further covering the requirements of process integrity, medium integrity, people integrity, and the output integrity, all these requirements together ensuring the information system integrity; and, secondly, data and information requirements - whatever may be the level at which the IS is considered (strategic, control or operational) – are required to be modelled in the context of their respective goal(s).

In search of a structure for integrity objective definition, the above then provides a basis for the Usefulness–Usability–Integrity paradigm. Specifically, Usefulness, implying the contribution of the information to task achievement, refers to the *relevance* attribute of the information for its intended purpose. For example, the recent history of a stock's price may be useful in deciding whether to buy or sell a stock. However, the recent history of the price of corn or oil may not be useful at all in deciding whether to buy or sell the stock. Against this, Usability, implying the presentation format and accessibility of information, refers to *feasibility* factors (attributes) such as availability, accessibility and understandability which help make it possible and easy to use the information. For example, information may be usable because it is available on the Internet, because it is presented in an intuitively obvious format or because it can easily be imported into a spreadsheet or database.

Literature identifies an universe of information attributes; namely, accuracy, usability, reliability, independence, timeliness, precision, completeness, relevance, sufficiency, ease of understanding, freedom from bias, consistency, trustworthy, brief, etc (as many as twenty three of them) [Delone and McLean, 1992; Mandke, 1996]. Within the framework of the approach mentioned here, appropriate attributes from these (and such others) concerning context, goal, and nature of information use, i.e. relevance and feasibility of use, then can be categorized under the Usefulness and the Usability objectives. In such case then, with all relevance and feasibility related requirements categorized or separated, drawing on integrity research investigations in security, auditing and quality arenas and in the information systems area, a basis emerges to identify Integrity objective, i. e., the dependability or trustworthiness of information, in the form of accuracy, consistency and reliability attributes of information covering correctness and appropriateness aspects [Mandke and Nayar, 1997; 1998].

2.2 Information Integrity-A Critical IS Requirement: Need for Economic

Framework for Information Integrity

Information requirements of usefulness, usability, and integrity are determinants of information value. Integrity attributes of accuracy, consistency and reliability are fundamental or basic to the information requirements of usefulness and usability and, therefore, to the value of information; and as a result a critical requirement of an IS. Trivial as it may look, this observation is not that obvious as to be found from the difficulty that researchers from the fields of EDP, auditing, data quality, computer science and information systems have in convincing the business managers to put their dollar on improving integrity of their information systems and of information there from.

This is because much of the research efforts in defining integrity have invariably been addressed without reference to developing a scientific structure for costs and benefits associated with Information Integrity. Briefly, the integrity research in computer science has its origin in study of secured computer systems and of confidentiality of information. Here, security has normally been taken to mean confidentiality, integrity and availability. Researchers involved with information security issue are at ease with this terminology *except* that the meaning of the word "integrity" is not adequately resolved, the word being frequently used to describe a range of attributes (or requirements) such as: validity of information in a computer system; correctness and protection of Trusted Computing Base and Procedures; reliability, accuracy, faithfulness, non-corruptibility and credibility of information transmitted; internal consistency of a system (a correctness aspect) and external consistency (correspondence with reality– an appropriateness aspect) ; etc [Mandke and Nayar, 1997]. In other words, the integrity research effort has been either very pragmatic, and/or technological, or almost semantic and in any case there is no reference to the cost benefit framework for Information Integrity – an aspect so crucial to business decisions [Tallberg, 1999].

In accounting/auditing research there seems to have been no corresponding debate concerning the exact meaning of "integrity". Specifically, the auditor assesses control risk, according to Statement of Auditing Standard: SAS 55, as determined by the relevant parts of the entity's (Auditee's) internal control structure. With respect to accounting information, relevant part of the internal control structure is thus made up of three parts (categorizations): the control environment, the accounting system, and the control procedures. This certainly offers a way of structuring the analysis of different possible control mechanisms. However, there is a problem in that there is no explicit coupling to cost and benefits, in the sense that items in different categories can be compared. The categorization in three parts is essentially ad-hoc.

Then there is the COSO report that provides an extended framework, but it is qualitative in nature. It sees internal control, from the management point of view, as consisting of five interlocking factors: monitoring, information and communication, control activities, risk assessment, and control environment. However, the same line of inadequacy that is levelled at SAS 55 above, that is, lack of explicit cost-benefit links between the components of model, applies here [Tallberg, 1999].

This points to the need to develop an economic framework for Information Integrity facilitating cost benefit analysis of Information Integrity so as to scientifically arrive at Integrity attributes and establish Information Integrity as a critical requirement for competitive advantage in business decisions at strategy, control and operational levels. Development of such an economic framework is a separate research study and does not form part of the current investigation. However, it is submitted that IS-based visualization of data and information when modelled as decision process (as discussed earlier) is amenable to developing such an economic framework.

2.3 Cost Benefit Analysis of Information Integrity: Outlining the Approach

Briefly, a generic business process covers entire supply chain from concept to delivery. A competitive business strategy calls for a good understanding of business process, which in turn requires choice of a good business model. With advances in computer-integrated systems and in data and information driven technologies, it has become possible to obtain process data and information on current basis and to manipulate it 'smarter' for strategic advantage. What this leads to is modelling the entire supply chain *emphasizing* 'information' as against material, flow, energy, etc. as has been the practice when, constrained by non-integrated technologies, businesses were mainly concerned with only 'standard' product in high volume syndrome for strategic advantages. And, specifically, what this leads to is a closed loop information and control system based model of a business process IS view of which generic business process is an integral part. It is by systematically controlling the information processing under this business process IS view that competitive advantage can be achieved in complex and changing business environments of today.

As mentioned earlier, core IS model can be represented as a decision process model. Traditionally, within the system-engineering framework, decision process is viewed to comprise of stages of forecasting (prediction), evaluation of alternatives and selection (for control or any other type of use as the case may be). However, what one has now is an information and control system based model of a business process and this IS is a complex, open system in that, at all levels, the information it processes is accompanied by respective objective and is impacted by and impacts its environment. As a result, for it, the more workable, extended model of a decision process spans multiple stages. They are: initial problem recognition (goal setting); identifying information variables for a complex problem system (goal); constructing problem solving opportunity and constraining spaces; developing information structure, and information structure dynamics models; and undertaking customized planning & design for development of alternatives for the evaluation of final choice for delivery of flexible information decision (itself an information), and the control implementation, i. e., the information use part [Mandke and Nayar, 2001].

What is significant is that all of the above stages $\{S_i\}$, from goal setting to final choice of information for use and control implementation, by themselves are complex information processing stages and, therefore, involve information gathering and processing activities with reference to their respective information bases. And of still greater implication is the reality that at each stage (S_i), these information gathering and processing activities are affected by uncertainties; resulting in errors in information processed from stage to stage [Mandke and Nayar, 2001].

The cost benefit analysis of Information Integrity then needs to be carried out with reference to this information processing reality between and within the sub-systems/components of the complex, open system that the business process IS view is. As indicated earlier, development of cost benefit analysis methodology is outside the scope of the present investigation. But in the rest of this sub-section, for the purpose of analytical explanation to the criticality of Integrity objective for competitive advantage (as against definitional approaches normally reported in the literature; for example computer security research), an effort is made to outline an approach to the cost benefit analysis of Information Integrity [Tallberg, 1999].

Consider any information processing stage (S_i) of the information and control system representing the business process IS view. Such IS can be viewed as formed by a number of core IS models connected in series and parallel [Mandke and Nayar, 1999; Matthews, 1971]. It is recalled that core IS model to which data and information are integral is modelled as a decision process. To outline the cost benefit analysis methodology of Information Integrity, one can consider such decision process.

The decision purpose can be taken to process/transform/convert data as in core IS to deliver information decision (by itself an information) which can be seen as a decision outcome so as to achieve better information use (for example better control). Thus the purpose of processing data/information through the core IS can be taken as "improvement in information use", which in turn then can be considered as the strategic or competitive advantage.

It is understood that this "improvement" (shown as "Δ") will be a function of the information (I) being processed under the stage $\{S_i\}$ and, accordingly, it can be represented by [ΔIU(I)], where IU(I) denotes the variable giving the upper bound of information use as function of "I" (given that such function can be defined). Let "α(I)" denote Usefulness factor and "β(I)" Usability factor. Both factors, functions of "I", may take values between (0,1] and, accordingly, can be seen as appropriately defined proportionality variables. Then, the improvement in information use at stage (S_i) is given by Equation (1).

$$\Delta IU(I)\,|_{S_i} = \left\{ [\alpha(I) \times \beta(I) \times IU(I)]\,|_{S_i} \right\} \quad (1)$$

On the face of it, Equation (1) would seem to give the benefit from the stage (Si) core IS viewed as a decision process. But, reality is different as one is dealing with complex, open core IS models and they *have* errors. As a result there *is* a question about the integrity of information "I" at the stage (Si).

Specifically, suppose there is a question regarding the accuracy of information, and let [A(I)] denote the concerned integrity quotient, which takes values between (0,1]. Then, the "benefit" or improvement in information use from information processing at stage (S_i) would get modified to as in Equation (2).

$$\Delta IU(I)\,|_{S_i} = \left\{ [\alpha(I) \times \beta(I) \times IU(I)]\,|_{S_i} \right\} \times \left\{ A(I)\,|_{S_i} \right\} \quad (2)$$

Of course our main objective is to outline an approach to cost benefit analysis of Information Integrity. Having considered the benefit, this then brings the question to that of costs. As can be seen, the correct assessment of benefit from the information processing at the core IS model under consideration can be done *only* when, from the benefit as accruing under Equation (2), the costs of information processing are accounted for. In other words, result of Equation (2) does not correctly state the benefit. What are these cost components then?

It is suggested that these cost components are those of originating information "I" [denoted by $COST_{OI}$ (I)], of analyzing integrity quotient of A(I) [denoted by $COST_{ANAL}$ {A(I)}], and the opportunity cost of analyzing A(I) [denoted by $COST_{OPPORT}$ {A(I)}]. Accordingly then the "benefit" in the form of improvement in information use as accruing at the information processing stage (S_i) is as given in Equation (3).

$$\Delta IU(I)\,|_{S_i} = \left\{ [\alpha(I) \times \beta(I) \times IU(I)]\,|_{S_i} \right\} \times \left\{ A(I)\,|_{S_i} \right\}$$

$$- \left\{ \text{COST}_{OI}(I)\,|_{Si} + \text{COST}_{ANAL}\,\{A(I)\}\,|_{Si} + \text{COST}_{OPPORT}\,\{A(I)\}\,|_{Si} \right\} \qquad (3)$$

Accounting for dynamic situations characterizing the information flow, if one considerably simplifies the query at hand and assumes $\alpha(I)$ and $\beta(I)$ to be given (something not to be the case in real world problem solving), the functions IU(I) and A(I) having their own respective first order transients with corresponding steady state values [here of upper bound value for IU(I) and value equal to numerical one for A(I)], and assumes all cost functions to be exponentially increasing with time, then what emerges from Equation (3) is that $\Delta IU(I)$ at the stage (S_i) under consideration *will* have a maximum value at a given time, and , among other things, *for* a given (what can be seen as an optimum) value of Integrity quotient "A". In other words there *is* an optimum I*I at which overall increase in information use benefit is maximum; achieving that I*I (implying accuracy, consistency, and reliability - if they can be quantified) is a costly process; and, to meet the demands of competitive advantage, resource commitment for achieving improved I*I, preferably optimum I*I, is critical.

3 Information Integrity Attribute Quantifiers

Having developed the basis for Integrity objective, with reference to requirements of cost-benefit analysis of Information Integrity, next question is to explore approaches to quantification of intrinsic integrity attributes of accuracy, consistency and reliability, and to suggest a method for demonstrating integrity improvement in information obtained and in system integrity.

3.1 Choice of Data/Information Model

This calls for first deciding on a workable data/ information model. Specifically for the problem objective at hand, data can be modelled by a triple $< e_i, a_i, v_i >$ representing input to the core information system, and information by a triple $< e_0, a_0, v_0 >$ representing output from the information system; $< e, a, v >$ representing datum a triple <entity, attribute, value> as developed by the database research community. This representation, which permits treating data/information as formal organized collection, allows segmenting integrity issue into issues concerning entities, attributes and values thereby making it feasible to study IS integrity analytically [Mandke and Nayar, 1998].

As networked computerized information systems contain errors that are made but not corrected, it is the above data/information model that needs to be further improved by replacing triple $<e, a, v>$ by triple $< e, a, v + \eta>$ wherein η represents error or noise component responsible for inaccurate, inconsistent and unreliable information and, thereby, for loss of integrity in IS. It may be mentioned that this more realistic representation of data/information model is most simplistic in that it is only accounting for information item on value (v) for the error that is made but not corrected. Certainly such errors can also be present even at the stage of "view" definition where "view" consists of entity types.

Considering that these error implications are present at each stage of an information system, namely; data origin stage, communication channel prior to processing stage, processing stage, communication channel at post processing stage and output stage, there are integrity implications at each stage of the IS and at the overall system level, as shown in Figure (1) [Rajaraman, 1996].

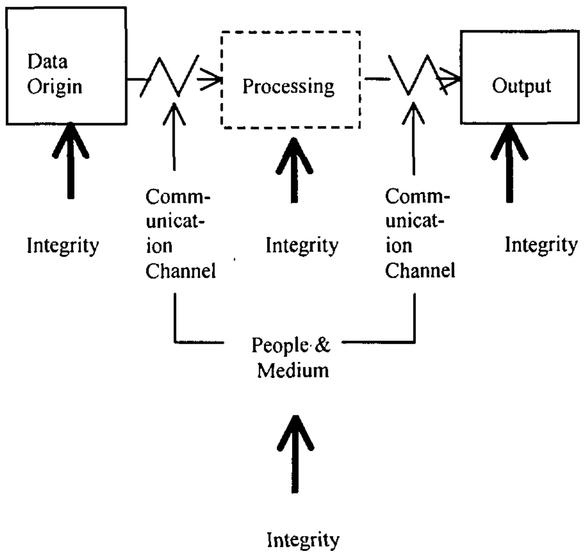


Figure 1. Conceptual Presentation of Integrity of an Information System

What is important for the investigation at hand is that integrity of the overall Information system is ensured if the integrity requirements of all parts of the system as in Figure (1) are ensured [Rajaraman, 1996]; integrity being defined in terms of attributes of accuracy, consistency and reliability whose quantification being the query to be pursued. In what follows this section addresses this query.

3.2 Accuracy

Accuracy is the degree of agreement between a particular value and an identified source. It can be assessed by identifying the relevant established source (standard) and by determining an acceptable tolerance. Specifically, the identified source provides the correct value – preferably the value corresponding to the optimum integrity. It can be an object or relationship in the real world; it can also be the same value in another database, or the result of a computational algorithm.

Given that value of data/information is expressed in a numerical, accuracy of the data/information can be quantified in a number of ways [Redman, 1992; AT&T Publication, 1992; Svanks, 1984; Ameen, 1989]:

i) Difference between the actual value (i.e., value of the identified source) and the value processed by the information system.

ii) $$\text{Error Ratio} = \frac{\text{Actual Error}}{\text{Acceptable Error}}$$

iii) $$\text{Accuracy Index} = \frac{\text{Number of correct values}}{\text{Number of total values}}$$

iv) Number of records examined: R

Number of records with at least one defect of loss of Accuracy: D1

$$\text{Percent Defective} = \left[\frac{D1}{R} \times 100 \right]$$

$$\text{Accuracy Index (A)} = \left[1 - \left(\frac{D1}{R} \right) \right]$$

Note: Percent Defective is a quantifier used extensively in statistical quality control.

v) Number of defects (cases of loss of accuracy) detected: D

Number of records examined: R

$$\text{Defects/Losses of accuracy per record} = \frac{D}{R}$$

$$\text{Accuracy Index (A)} = \left[1 - \left(\frac{D}{R} \right) \right]$$

It may be mentioned that defect denotes accuracy violation, i.e., presence of error, and hence the absence of accuracy. Ratios based on defects/errors can be converted into accuracy ratio by the transformation:

Accuracy Ratio = 1 – Defect (i.e., Error) Ratio.

Understandably, notion of accuracy quantified as above has many issues not considered here. What if correct value of the identified source is undefined, or simply unknown? And of course what if data/information is say a name or has an alphanumeric value or is a video image; how is error or defect defined then?

3.3 Consistency

Consistency is the degree to which multiple instances of a value satisfy a set of constraints. The multiple instances may exist across space (such as databases or systems) or over time.

Thus, consistency is with respect to a set of constraints and data/information is said to be consistent with respect to a set of constraints if it satisfies all constraints of the data/information model [AT&T Publication, 1992]. Constraints can apply to the same attributes in different entities (such as the salary attribute in the entities of several employees); they can also apply to different attributes in the same entity (such as the salary level and salary attributes in the entity for a particular employee).

Given the number of constraints specified (CS) and the number of constraints for which error/defect detected in the sense constraints are not satisfied (CE), consistency can be quantified as follows [Svanks, 1984]:

$$\text{Consistency Index (C)} = \left[1 - \left(\frac{CE}{CS} \right) \right]$$

3.4 Reliability

Reliability, which traditionally is a large concern in the system development lifecycle model, is a little complex attribute to define as it has a dual meaning in modern technical usage. In the broad sense, it refers to a wide range of issues relating to the design of large systems (complex computerized information system [CIS] included), which are required to work well for specified periods of time. In such a case, the term "reliability" includes descriptors such as "quality" (commonly understood from the traditional "standard' product angle) and "dependability", and is interpreted as a qualitative measure of how a system matches the specifications and expectations of a user. From this point of view for an IS the definition of reliability given as "accuracy with which information obtained represents data item in whatever respect the information system processed it" [Mandke and Nayar, 1998] can be seen to define the reliability requirement for the IS as a whole; reliability index being amenable to quantification through techniques such as Analysis of Variance (AOV).

In a narrower sense, however, reliability is a measure denoting the probability of the operational success of an item under consideration. The notion of reliability, in this case, may be applied to a single component (e g., a diode or a light bulb); a complex system (e g., an aircraft, a computer or a network of computers); a computer program; a procedure (e g., conversation between a pilot and an air-traffic controller); an element of an IS; namely, data, i. e., IS input, or IS output which is "data processed", i.e., information; or even a human. Specifically, reliability analysis is concerned with occurrences of undesirable or unanticipated events during the course of operation of a system or an item and the impact of these events on the system's behaviour or the item's use. And the undesirable events may be failures of components (and, for information systems, failures of resulting data and information in the form of data/information errors) caused by deterioration or wearing out of components due to age and usage or even design problems and inadequacies, etc. surfacing in the course of the use of the system [Viswanadham, Sarma and Singh, 1987].

In view of the IS function of delivering information for use, reliability of each output item delivered, i. e., the reliability of "information", which is the element of the IS, becomes significant and, keeping in mind the needs of user domains, the same can be heuristically defined as follows: "Reliability refers to completeness, currency and auditability of data/information. Specifically, data/information is complete when all component elements are present. Information is current when it represents the most recent value. And, information is auditable if there is a record of how it was derived and that record allows one to trace information back to its source."

Within above framework then, using the earlier described concept of percent defective, given the number of records examined (R) and the number of records with at least one defect of incompleteness, lack of currency or inability of auditability (D2), the reliability index can be quantified as follows.

$$\text{Reliability Index (R)} = \left[1 - \left(\frac{D2}{R} \right) \right]$$

3.5 Integrity Profile

Consider an information system designed and developed for an application area. It is appreciated that each application area, consistent with information usage requirements, will have application area specific order of significance for integrity attributes. Let Wa, W_c and W_r represent significant weightages for the integrity attributes accuracy, consistency and reliability, respectively, for the application area under consideration. These weightages may take values between (0-10].

Consider a user using the above information system for the application at hand. Let the Information Integrity attribute indices as observed at the user end in this specific example be: Accuracy (A) = 0.78, Consistency (C) = 0.55 and Reliability (R) = 0.85. Then Information Integrity Profile from the user end can be represented as in Figure (2) [figure not to the scale].

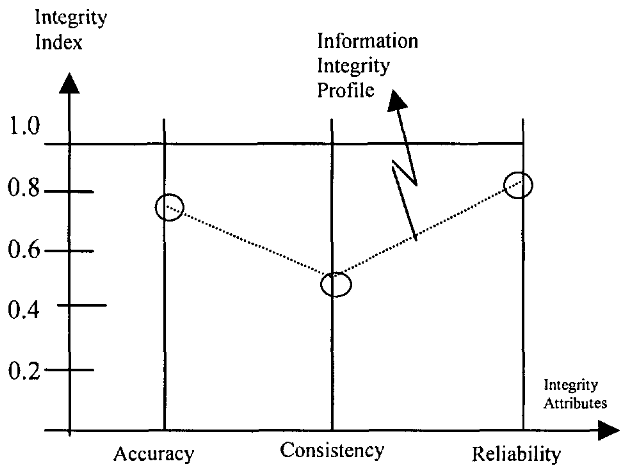


Figure 2. Information Integrity Profile

3.6 Cumulative Information Integrity Index (CIII)

Let Information Integrity attribute, depending on the range in which the attribute index value falls, be assigned a 5-point scale as shown in Table 1.

Attribute Index Value Range	Scale	Points
[1–0.8]	H	5
(0.8–0.6]	G	4
(0.6–0.4]	F	3
(0.4–0.2]	E	2
(0.2–0]	D	1

Table 1: 5 - Point Scale for Integrity Attributes

In the example under consideration, the Information Integrity attributes then have the scales and points as given in Table (2).

Attribute Index Value Range	Scale	Points

Accuracy Index (A)	G	4
Consistency Index (C)	F	3
Reliability Index (R)	H	5

Table 2: Integrity Attributes Scales and Points

Then with a view to quantify the overall Information Integrity Index for the given application by the user, a Cumulative Information Integrity Index (CIII) may be given by:

$$\text{CIII} = \frac{4W_a + 3W_c + 5W_r}{W_a + W_c + W_r}$$

For example, if $W_a = 6$, $W_c = 5$ and $W_r = 8$, then

$$\text{CIII} = \frac{(4 \times 6) + (3 \times 5) + (5 \times 8)}{(6 + 5 + 8)}$$

$$= \frac{79}{19} = 4.158$$

CIII will thus take a value between [1–5]. It could be the situation that this value of CIII may be low from the user point of view (as it is not optimum) and the user may be requiring minimum CIII value of 4.25, which is say closer to the optimum I*I. Further, user may want to improve CIII with additional requirement of Consistency Index having minimum "G " scale. It is to achieve this integrity improvement that the user would then need to incorporate Information Integrity Technology.

It may be mentioned that indices developed here are compatible with analytical requirements of cost-benefit analysis of Information Integrity as outlined in Section (2.3). Before one proceeds with further development of Information Integrity Technology Product platform, a word of caution is warranted here. The quantification of integrity attributes is not a trivial task even when it is possible [Redman, 1992]. Quantifiers suggested above do not bring out the complexity involved. In respect of accuracy quantification, it is already mentioned that there could be a problem of correct value of the identified source (also called standard) being undefined, or being simply unknown. In a situation, an assumed standard itself may be incorrect as is often the case with data gathered some time in the past and with no corroborating evidence. In yet another situation there may be more than one correct value. Then there is a problem of how to quantify accuracy if the value does not lie on a real line, i.e., it is not a numerical.

As regards to consistency quantifiers, it is a relatively simpler concept than accuracy. Even then, it can assume complexities when all real database inconsistencies are to be measured (and which will be the need). Coming to the reliability attribute, it is always a difficult index to develop. There may be more than one way of calculating the reliability index and there will always be a need to develop one based on the nature of available data, form of data and computation aids available for processing.

Finally, exercise undertaken herein considers problem of quantifying integrity attributes when errors made but not corrected are at the level of information item on value (v). But in data/information modelling exercise, errors can be present even at "view" defining level itself. Specifically, data/information modelling exercise begins with modelling facts observed from the real world in the form of "view" which consists of one or more structures called entity types, from where one builds attributes, their domains and, in the end, values; thereby forming the triple <e, a, v>. What if errors are present at the stage of "view" defining stage itself and which invariably is the case. How does one define and quantify integrity attributes in such case? Further, so far what all one has discussed is only in terms of intrinsic integrity attributes. Depending on the context and the nature of use of information, one will also have to similarly develop methods for defining and quantifying extrinsic integrity attributes.

All these areas then constitute further research needs in the context of integrity attribute quantifiers for integrity improvement.

4 Information Integrity Technology Implementation Steps

With a suggestion for Integrity indices as above, within the framework of Information Integrity attributes of Accuracy (A), Consistency (C) and Reliability (R) argued, one can then identify Information Integrity Technology Implementation steps as follows:

i) Understand the user application of the computerized information system under consideration.

ii) Establish organizational standard pertaining to data/information vis-à-vis requirements of: accuracy, consistency, reliability and cumulative integrity, based on application area and study of organizational practices.

iii) Study data/information flow through the Information System and define database(s).
Note: Apart from knowing how the Information System processes the data and apart from understanding more about the "noise" in the system, the study would also necessitate

knowing wherefrom, how and data/information of what integrity flows into the system.

iv) Develop the Information System Model as in Figure (3), based on understanding of data/information flow in the system for the identified database.

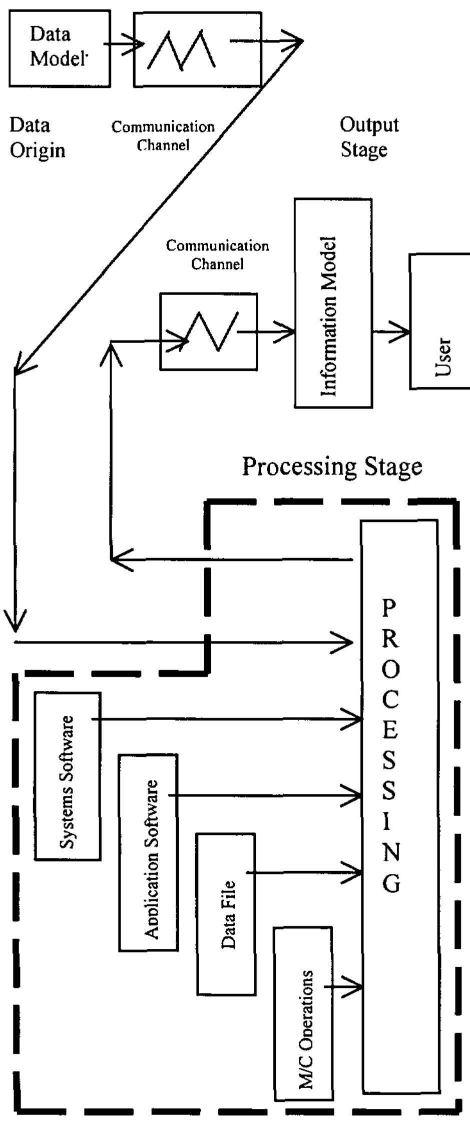


Figure 3: Data/Information Flow Model for an Information System for Implementing Information Integrity Technology

v) Specify and document data rules, also known as edits, to be implemented to study accuracy and consistency of the data/information.

vi) Choose a method for calculating Reliability Index, keeping in view advantages, disadvantages and convenience of application while accounting for factors such as nature of available data, form of data and available computation aids.

vii) Develop Integrity Analysis Software for analyzing intrinsic Information Integrity attributes of accuracy, consistency and reliability.

In addition, Integrity Analysis Software may also undertake statistical analysis (time series analysis and other techniques) of error patterns signifying irregular changes, which contribute to loss of Accuracy and Consistency and of causes, which contribute to loss of Reliability. This in turn leads to the development of:

a) filter to detect error or cause that occurred sometime in the past at time $(t - \tau)$,

b) an estimator to estimate error or cause that occurred in the immediate past at time (t), and

c) a predictor to predict error or cause that may occur sometime in future at time $(t + \tau)$.

viii) For Data/Information Flow Model in Figure (3), select a data sampling point at the output of a subsystem (or at an intermediate point within the subsystem), as close to the beginning of the Information System as possible.

ix) Depending on how data arrives at the sampling point (continuously or in batches), develop a continuous or batch processing sampler (a sampling program) to randomly select a sample of records arriving at the sampling point. Along with sampling records, the sampler program should also select some identifier of the sampling point and record of the data and time of sampling.

x) Following the selection of a sampling point and development of a sampler, select points for maintaining audit trail for sampled records.

xi) These points for maintaining audit trail may be selected at points at the output of subsystems (or at intermediate points within the subsystems) following the sampling point.

xii) Once the points for maintaining audit trail for records sampled are identified, develop a Sampled Records' Audit Trail (SRAT) program to separate or pull out (at the points selected) the audit records.

xiii) Ensure that sampler program and SRAT program are developed in such a way that they can download sampled records and records for audit trail as in (ix) and (xii) into a database to be set up (see (xiv)).

xiv) Accordingly, download the sampled and audit trailed records on mainframe or minicomputer or on personal computer/workstation so as to set up an Error Detection Database, based on hardware and software considerations and number of sampled and audit trailed records.

xv) Using the Integrity Analysis Software developed in (vii), analyze the Error Detection Database to:

a) identify data rule violations in respect of accuracy and consistency attributes,

b) establish degree of integrity of data/information in respect of Information Integrity attributes of accuracy and consistency, based on data rule violation statistics,

c) obtain reliability index for the database along with analysis of factors which contribute to the level of reliability,

d) develop Integrity Profile and Cumulative Information Integrity Index, based on indices of accuracy, consistency, and reliability attributes, and

e) study changes in database not expected, i.e., irregular changes.

xvi) Compare the Integrity profile and indices obtained as in [(xv (b))–(xv (d))] with: standards in (ii) – local, regional, national, international as the case may be – and the user specifications on Integrity, so as to know what is expected of Information Integrity Technology. This would also facilitate ordering or ranking of the Integrity attributes from the point of which attribute needs maximum improvement effort.

xvii) For each of the Integrity attributes of accuracy and consistency, then, further analyze irregular changes either by subsystem or by field (in that order of priority of choice) and locate separate Integrity improvement opportunities at each of appropriately identified pairs of a given field at a given subsystem.

xviii) Similarly locate reliability improvement opportunities at each of the subsystems based on reliability factor analysis in (vii).

xix) Having located pairs of a given field at a given subsystem, each for improvements of accuracy and consistency and having located given subsystems for reliability improvement opportunities, further analyze the Error Detection Database and study irregular changes at each of pairs corresponding to accuracy and consistency attributes and study reliability factors at each of the subsystems, so as to understand over the time error patterns and causes contributing to loss of accuracy, consistency and reliability.

This would then facilitate detection of error or cause that occurred sometime in the past ($t - \tau$), or estimating error or cause at time (t), or predict error or cause that may occur at a future time ($t + \tau$).

xx) Now develop Information Integrity Improvement Action Plan for locations identified in respect of integrity improvement opportunities based on assessment as in (xvi) of integrity improvement target and based on the understanding of error patterns and factors for loss of intrinsic Information Integrity attributes as in (xix). This Integrity Improvement Action Plan may comprise restructuring subsystem(s) previous to the point of occurrence of error, improving integrity of data origin stage, improving communication channels, etc.

xxi) Finally, study performance of the Information System on incorporation of the Information Integrity Technology as outlined above. Accordingly obtain the intrinsic Information Integrity attribute indices, Integrity profile and Cumulative Information Integrity Index and compare them with appropriate reports before implementation of Information Integrity Technology available vide [xv (b)], [xv(c)] and [xv (d)], so as to quantify integrity improvement achieved and to check if it is as per customer expectation.

5 The Information Integrity Technology Product Platform

The Information Integrity Technology product thus emerging would then be a software product developed on a platform having following components:

- the user data rules list for error detection
  Note: Data rule is that which must hold true in an Information System
- the Integrity Analysis Software for:
  - Accuracy
  - Consistency
  - Reliability

  - Integrity profile for the Information System
  - Integrity Indices
- the sampling program
- the Sampled Records' Audit Trail (SRAT) program
- the program for
  - statistical analysis of errors/causes for loss of integrity
  - factor analysis for reliability
  - time series analysis
- the program for:
  - detecting errors/causes (filter program)
  - estimating errors/causes (estimation program)
  - predicting errors/causes (predictor program)
- the generation of Error Detection Data Base
- the reporting based on analysis of Error Detection Data Base in terms of:
  - errors and causes detected; their locations in the Information System and their significance
  - error and cause patterns and trends obtained through statistical techniques such as time-series analysis
  - detection, estimation and prediction of errors and causes
  - identification of Integrity improvement opportunities
  - deciding and implementing Information Integrity Improvement Action Plan for Integrity Improvement opportunities identified (probabilistic action plan as also manual action plan included)
- obtaining improved Integrity Profile and Index
- documentation:
  - data rules list encoding the specifications for the Integrity Analysis
  - software and the reporting facility. This calls for user interaction.
  - the individual program in accordance with the systems and program documentation within the user organization
  - operating instructions for each program
  - program maintenance and test procedures
  - training material for users

6 Conclusion

Computerized information systems contain errors that are made but not corrected by controls built-in at system analysis and design stage of the Information System. Therefore, the confirmation of potential or suspected anomalies in a live database and subsequent integrity improvement becomes an essential facility (beyond application controls) within an Information System. This facility is the Information Integrity Technology.

It is useful to view integrity objective in the context of information Usefulness-Usability-Integrity paradigm that in turn makes it possible to undertake cost benefit analysis of Information Integrity. Specifically what it means is that benefits of increase in information use by achieving integrity are compared with costs of acquiring information as also of evaluating and applying integrity. There is a competitive advantage in the form of an overall benefit, as long as increase in information use value is more than costs. What is interesting is that this overall benefit function representing the overall increase in information use, after accounting for the costs, has a maximal. This then suggests an optimum integrity value that is desirable for maximum competitive advantage. It is this observation that then offers the basis for developing Information Integrity attribute quantifiers leading to the statement of Information Integrity Technology based on a feedback control system approach.

The users of computerized information systems have to undertake computer housekeeping to incorporate Information Integrity Technology in their information systems, so as to avoid serious potential losses occasioned by errors that were made (due to factors external to application control) but not corrected. In concrete terms, Information Integrity Technology will be an application and user - specific software which, for an Information Flow Model as in Figure (3), samples on-line, periodically and systematically, records arriving at an appropriately chosen point, follows or keeps track of sampled records at subsequently identified points through the information system and stores the records so sampled and obtained through follow up (audit trail), to set up error detection database. Information Integrity Technology then analyses this error database to identify errors, i.e., changes not expected, and to quantify resulting loss of integrity therefore, so as to develop Integrity Improvement Action wherein Information Integrity opportunity is identified and implemented.

Understandably, this Information Integrity Technology will have to be developed in a computer language compatible with the information-processing environment of the user organization. This calls for organizational IS planning, devising policies, standards, and guidelines pertaining to data. If this is not ensured, net result is non-compatible, and un-shareable data/information. An important step in the implementation of Information Integrity Technology, therefore, is data rule specification, defining data rule standard, which is also needed for undertaking Information Integrity Analysis.

Yet another area that calls for standards pertains to degree of integrity. As mentioned, the application area would influence the degree of accuracy, consistency and reliability. The application area would also influence values of W_a, W_c, W_r. Further, quantification of Integrity attribute such as accuracy calls for identification of data/information sources and their standards, i.e. correct values. In implementation of Information Integrity Technologies, it would therefore be necessary to establish these application area specific standards representing requirements of degrees of integrity as also of values of Integrity attribute significance factors.

Finally, it is important to appreciate that the development of standards as above would facilitate implementation of Information Integrity Technology products for different subsystems of the information system as also for the total system. This would call for support of reputable software developers and vendors for the purpose. Further, these Information Integrity Technology products would cover data/information in various forms – numerical, alphabetic, alphanumeric, video-images or any other – and that too for different application areas. This would open a new vista in terms of design, development, commissioning, operation and maintenance of data technologies, hitherto not attended, for ensuring on-line integrity of computerized information system.

Acknowledgements

Authors wish to express their sincere appreciation to Professor V. Rajaraman, Hon. Professor, SERC, IISc, Bangalore, India for his valuable inputs in developing research models in Information Integrity.

References

[1] Ameen, D.A. (March 1989) *Systems Performance Evaluation*, Journal of Systems Management, pp. 33–36.

[2] AT&T Publication (1992) *Data Quality Foundations*, Published by AT&T Quality Steering Committee, USA.

[3] Delone M. H., and McLean E. R. (1992) *Information Systems Success: The Quest for the Dependent Variable*, Information Systems Research 3 (1).

[4] Mandke Vijay V. (1996) *Research in Information Integrity: A Survey and Analysis*, Proceedings of the JNCASR and SERC Discussion Meeting at IISc Campus, Bangalore on Information Integrity – Issues and Approaches, Edited by Rajaraman V. and Mandke Vijay V., published by Information Integrity Foundation, New Delhi, India.

[5] Mandke Vijay V., and Nayar M.K. (1997*) Information Integrity – A Structure for its Definition*, Proceedings of the 1997 Conference on Information Quality, Edited by Diane M. Strong and Beverly K. Kahn, MIT, Cambridge, Massachusetts, USA.

[6] Mandke Vijay V., and Nayar M.K. (1998) *Design Basis for Achieving Information Integrity – A Feedback Control System Approach*, IFIP TC 11 Working Group 11.5 Second Working Conference on Integrity and Internal Control in Information Systems, Edited by Sushil Jajodia, William List, Graeme W. McGregor and Leon A.M. Strous, Published by Kluwer Academic Publishers.

[7] Mandke Vijay V., and Nayar M.K. (1999*) Modeling Information Flow for Integrity Analysis,* Proceedings of the 1999 Conference on Information Quality, Edited by Y. V. Lee and G. K. Tayi, MIT, Cambridge, Massachusetts, USA.

[8] Mandke Vijay V., and Nayar M.K. (2001) *Information Envelope and its Information Integrity Implications: For a complex, changing environment, modeling a generic business process as an integral to a closed loop information and control system characterized by uncertainty,* Proceedings of the 2001 Conference on Information Quality, Edited by Elizabeth M. Pierce and Raissa Katz-Hass, MIT, Cambridge, Massachusetts, USA.

[9] Matthews Don Q. (1971) *The Design of the Management Information System*, Auerbach Publishers, NY.

[10] Nayar M.K. (1996) *A Framework for Achieving Information Integrity*, Proceedings of the JNCASR and SERC Discussion Meeting at IISc Campus, Bangalore on Information Integrity – Issues and Approaches, Edited by Rajaraman V. and Mandke Vijay V., published by Information Integrity Foundation, New Delhi, India.

[11] Rajaraman V. (1996) *Information Integrity – An Overview*, Proceedings of the JNCASR and SERC Discussion Meeting at IISc Campus, Bangalore on Information Integrity – Issues and Approaches, Edited by Rajaraman V., and Mandke Vijay V., published by Information Integrity Foundation, New Delhi, India.

[12] Redman T.C. (1992) *Data Quality: Management and Technology*, Bantam Books, NY.

[13] Svanks Maija I. (1984) *Integrity Analysis: A Methodology for EDP Audit and Data Quality Assurance*, EDP Auditors Foundation, Inc.

[14] Tallberg Anders (1999) *An Economic Framework For Information Integrity*, Forskningsrapporter Research Reports, Swedish School of Economics and Business Administration.

[15] Viswanadham N, Sarma V. V. S., and Singh M.G. (1987) *Reliability of Computer and Control Systems*, North-Holland, Amsterdam.

[16] Wang R. and Strong D. (1993) *An empirical Investigation of Data Quality Dimensions: A Data Consumer's Perspective*, TDQM-93-12, The TDQM Research Program, MIT, Sloan School of Management, Cambridge, USA.

Efficient methods for checking integrity: a structured spreadsheet engineering methodology

Kamalasen Rajalingham, David Chadwick and Brian Knight
University of Greenwich
School of Computing and Mathematical Sciences
30 Park Row, Greenwich, London SE10 9LS, United Kingdom
K.Rajalingham@wmin.ac.uk
http://www.kamalasen.com/spreadsheets.html

Keywords: spreadsheets, errors, systems development life cycle, software engineering, auditing

Received: February 10, 2002

This paper describes an approach to the provision of a Structured Spreadsheet Engineering Methodology. The proposed methodology is mainly based on the classical systems development life cycle, structured methods and software engineering principles. It addresses the widespread problem of spreadsheet errors and is an extension to published work by Chadwick-97, Rajalingham-98, Rajalingham-99, Rajalingham-99a, Rajalingham-00, Rajalingham-00a, Rajalingham-00b and Rajalingham-01. This methodology also helps in training users in the process of spreadsheet building. Although there are variations of the life cycle for systems development, they are fundamentally similar to each other. The proposed Structured Spreadsheet Engineering Methodology is primarily based on the systems development life cycle described by Aktas-85, Jackson structures (Jackson-75) and approaches recommended by other authors. Numerous approaches are incorporated into this framework, making it a highly integrated and structured methodology for spreadsheet design and development. Apart from the concepts and principles borrowed from the above methods, the methodology also contains new developments in the research into integrity control of spreadsheet models.

1 Introduction

The problem of data integrity in spreadsheet models has attracted much attention and concern over the years. Although much attention has been devoted to the verification and validation of information processes in general, there has been relatively little research undertaken into developing satisfactory solutions to the problem of spreadsheet errors. As a result, this phenomenon remains prevalent, having a serious effect on businesses and costing them lots of money.

2 Spreadsheet errors

Numerous publications over the years have confirmed and provided sufficient evidence that spreadsheet errors have adversely affected businesses. The phenomenon of spreadsheet errors can be explored or investigated from three different perspectives. They are as follows:

i. the frequency of the errors
ii. the real-life consequences of spreadsheet errors
iii. the different types and classes of specific errors

2.1 Frequency of spreadsheet errors

This aspect of the spreadsheet integrity problem has been given adequate attention. As a result, most of the literature on this issue concern the frequency of the occurrence of spreadsheet errors. Numerous related experiments and studies have been carried out in the past and sufficient information is presently available.

Based on material from a wide variety of publications, Panko and Halverson (Panko-96) have organised the research findings into an excellent compilation statistics on the frequency of spreadsheet errors. Large and well-known auditing firms such as KPMG Management Consulting and Coopers & Lybrand (now known as PricewaterhouseCoopers) have also reported that spreadsheet errors are occurring at appalling rates. These organisations constantly audit very large numbers of spreadsheet models from various clients. KPMG Management Consulting (KPMG-97) and Coopers & Lybrand (Ward-97) have reported that more than 90% of the spreadsheet models they inspected contained errors.

The conclusion that can be reached after examining all these reports is that the rate of occurrence of spreadsheet errors is indeed significantly high.

2.2 The real-life consequences

The second aspect of the spreadsheet integrity phenomenon concern the real-life impact and consequences of spreadsheet errors. The purpose of investigating the negative effect of these errors on businesses is to enable better understanding of the magnitude and seriousness of the problem of spreadsheet errors.

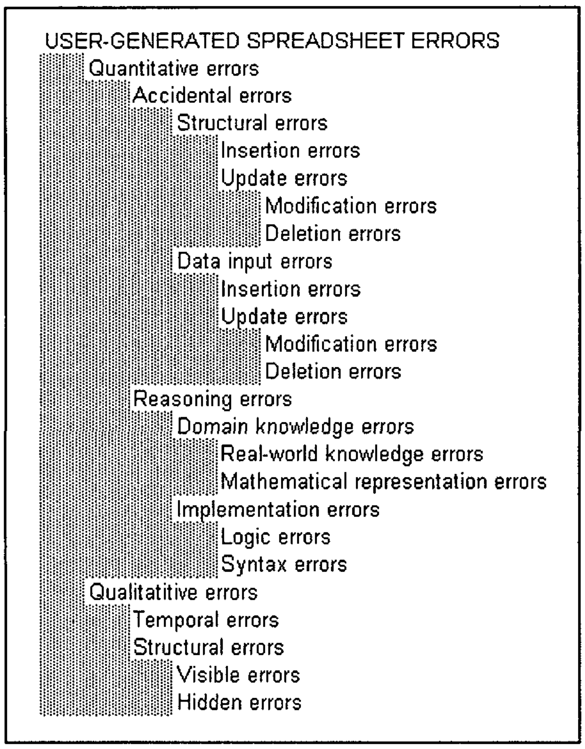
Numerous recent publications have indicated the seriousness of spreadsheet errors and their adverse impact or potential impact on businesses, quoting many relevant cases. It must however be noted that these are just reported cases. There must be many other similar cases that have not been brought to public attention due to fear that they might affect the reputation of the company involved.

Some of the reported cases are given in Chadwick et al (Chadwick-97), Rajalingham & Chadwick (Rajalingham-98) and Rajalingham et al (Rajalingham-99). Although many of these cases are not based on formal research, they do show that spreadsheet errors are deemed serious enough to be reported in the general business and computing press. They also show the extent of the damage that can be caused. If this situation prevails, companies will be losing a lot of money due to poor decisions made based on the unreliable spreadsheet figures.

2.3 Types and classes of specific errors

This is an area that has not at all been adequately explored or discussed in publications on the problem of spreadsheet errors. Four research papers that have addressed this issue and analysed specific types of spreadsheet errors from business and academia are Panko and Halverson (Panko-96), Chadwick et al (Chadwick-1997), Rajalingham & Chadwick (Rajalingham-98), and Rajalingham et al (Rajalingham-99).

The outcome of research into specific types of spreadsheet errors is the provision of a more comprehensive classification or taxonomy of spreadsheet errors. The elements of this classification of spreadsheet errors are presented and elaborately discussed by Rajalingham & Chadwick (Rajalingham-98) and Rajalingham et al (Rajalingham-99,00), supported by relevant examples. *Figure 1* shows an improved version of the model, by Rajalingham et al (Rajalingham-99, 00).



USER-GENERATED SPREADSHEET ERRORS
- Quantitative errors
  - Accidental errors
    - Structural errors
      - Insertion errors
      - Update errors
        - Modification errors
        - Deletion errors
    - Data input errors
      - Insertion errors
      - Update errors
        - Modification errors
        - Deletion errors
  - Reasoning errors
    - Domain knowledge errors
      - Real-world knowledge errors
      - Mathematical representation errors
    - Implementation errors
      - Logic errors
      - Syntax errors
- Qualitatitive errors
  - Temporal errors
  - Structural errors
    - Visible errors
    - Hidden errors

Figure 1: Taxonomy of spreadsheet errors.

3 Rationale for a structured approach or methodology

An investigation carried out into formal development methods used in industry revealed that large scale software developments by professional computing staff were subject to formal development methods and monitored by auditors for errors throughout their life-history. On the other hand, small scale applications, such as spreadsheet models, were not subjected to such methodologies or structured methods (Chadwick-97).

It has been observed for several years at the University of Greenwich, UK that students keep making similar mistakes and errors when developing spreadsheet models. However, presently available literature on spreadsheet errors do not offer sufficient or effective methods, tools, techniques or guidelines that can be used in the process of constructing spreadsheets. It was therefore realised that better methods were needed in order to aid students in making fewer errors whilst building a spreadsheet and to aid the lecturer in assessing the students' spreadsheet for correctness.

Research findings over the last few years clearly show the need for a structured approach or discipline for spreadsheet development. This is mainly due to the absence of recognised methods for spreadsheet design and development. Many authors on the subject have recently called for a disciplined or structured approach to spreadsheet building. Panko and Halverson (Panko-96) distinctly state that there is a need to adopt strict programming disciplines in dealing with complex spreadsheets. The journal paper by Panko (Panko-98) also says that there is an obvious need to begin adopting traditional programming disciplines due to the similarity between spreadsheet errors and programming errors. The paper also states that there is far too little knowledge of spreadsheet errors, which implies that much more research has to be undertaken into spreadsheet errors.

Based on various published reports, we can come to the conclusion that there is a need for the adoption of a structured methodology in spreadsheet development. This will certainly help address the currently major problem of spreadsheet errors.

4 The Structured Spreadsheet Engineering Methodology

This section presents a comprehensive methodology for spreadsheet development based on various approaches, methods, tools and techniques proposed in publications as well as original work done by the authors of this paper. This is in line with the constant call by numerous authors to impose some discipline and structure in spreadsheet development, as a step towards controlling the integrity of spreadsheet models.

The unique feature of this methodology is that it brings together various approaches, most of which have already been discussed in previous publications (Rajalingham-99,99a,01; Knight-00). The basic framework or foundation for this methodology is the classical systems development life cycle given by Aktas (Aktas-87) and Jackson-like tree structures (Jackson-75). The steps and stages within this life cycle are now being proposed for spreadsheet models as well. It should also be noted that the proposed methodology caters for large and complex spreadsheet systems. As such, some of the steps can be omitted when building small or simple models.

The methodology consists of five main stages:

Stage 1: Planning
Stage 2: Analysis
Stage 3: Design
Stage 4: Implementation
Stage 5: Maintenance

Stage 1: Planning

Step 1: Request for construction of the spreadsheet system

The process of building a spreadsheet system or modifying an existing one is initiated through a request by management.

Step 2: Initial investigation of the spreadsheet system

The main requirements for the spreadsheet system are defined. The requirements of the model sponsors are elicited and analysed. The overall objective or purpose of the spreadsheet model is also established. Based on the information gathered, an assessment of the nature, scale and complexity of the model is carried out.

There is also a study of the spreadsheet system's interfaces with other elements such as hardware, people, databases and other software. This encompasses requirements gathering at the system level with a small amount of top-level design and analysis.

Step 3: Feasibility study of the spreadsheet system

This step is important as it establishes whether the proposed system is in fact best developed using spreadsheet software. If certain requirements cannot be met by the spreadsheet software, other alternatives should be considered, such as a database system. The analyst should also determine if expertise to build the spreadsheet system is available.

Stage 2: Analysis

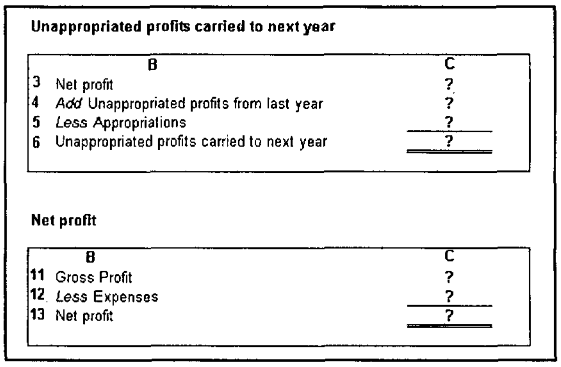
The requirements gathering process is intensified and focused specifically on the spreadsheet model.

Step 1: Redefine/define the spreadsheet problem

The problem to be put on the computer and the nature of the spreadsheet model to be built has to be clearly understood. This includes the information domain for the system as well as the required functions, performance, and interfacing.

This step involves translating the requirements of the model sponsors into a set of spreadsheet model outputs. Each spreadsheet model would normally have one or more associated outputs. The methodology insists on the presentation of outputs on one or more separate worksheets. They should neither appear in the worksheet containing the spreadsheet model schema, nor the worksheet containing the model inputs.

The structure of each output is designed and implemented on the physical spreadsheet. Only the editorial aspects of each desired output are implemented at this stage. These include titles, headings and formula and data labels. An example (based on Wood-96) is shown in *Figure 2.*



Unappropriated profits carried to next year

	B	C
3	Net profit	?
4	*Add* Unappropriated profits from last year	?
5	*Less* Appropriations	?
6	Unappropriated profits carried to next year	?

Net profit

	B	C
11	Gross Profit	?
12	*Less* Expenses	?
13	Net profit	?

Figure 2: Spreadsheet model output.

Chadwick et al (Chadwick-99), Knight et al (Knight-00) and Rajalingham et al (Rajalingham-01) have proposed the use of Jackson structures (Jackson-75) for visually representing the relationships between elements of the spreadsheet model outputs, as well as their precedents and dependants. In the same way that a computer program can be broken down into smaller parts and represented in the form of a tree, elements of a spreadsheet model can also be represented in a similar manner.

Based on the spreadsheet model outputs shown in *Figure 2*, the corresponding dependency diagram based on Jackson structures is shown in *Figure 3*. However, when a top-down approach is adopted without showing duplication of nodes, the structure of the model schema could take the form of a graph instead of the desired tree structure. The purpose of this is to distinctly show instances of multiple dependants of a particular formula of the model schema. This potentially results in a structure as shown in *Figure 4*.

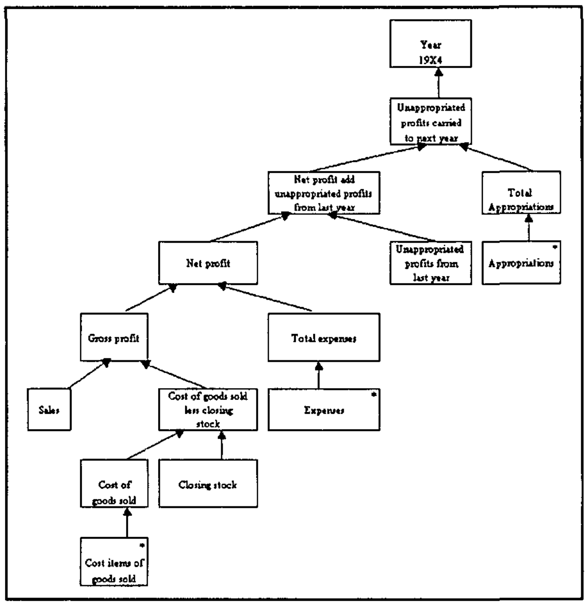

Year 19X4
Unappropriated profits carried to next year
Net profit add unappropriated profits from last year
Total Appropriations
Net profit
Unappropriated profits from last year
Appropriations *
Gross profit
Total expenses
Sales
Cost of goods sold less closing stock
Expenses *
Cost of goods sold
Closing stock
Cost items of goods sold *

Figure 3: Conceptual design.

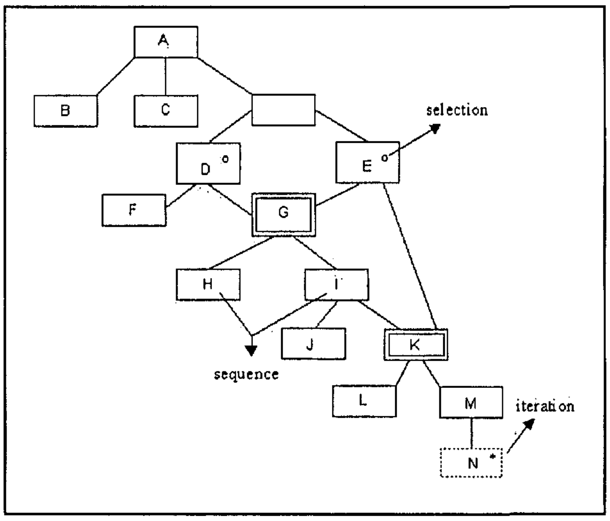


Figure 4: Conceptual design in graph form.

Step 2: Understand existing Spreadsheet System (if there is one)

Often times, especially in large organisations, there is a need to modify or improve an existing spreadsheet system rather than developing one from scratch.

Step 3: User requirements and constraints of the spreadsheet system

A good understanding of the requirements of users and constraints to be imposed within the system is important and should be carefully considered.

Stage 3: Design

The design process translates requirements into a representation of the system that can be assessed for quality before system building begins. This stage consists of two main phases, namely, *logical design* and *physical design*.

Step 1: Logical design

The logical perspective consists of a formal and implementation-free description of the model's logic and data structures (Isakowitz-95). The purpose of this step is to resolve sub-structures with formulae or data with multiple dependants. A formula or data with multiple dependants normally form a graph. Structurally, the aim at this stage is to transform all *graph sub-structures* in the conceptual model to *trees* so that the entire model is in the form of a Jackson-like tree structure. From a more logical perspective, the objective of performing this task is to enable the direct mapping of the Jackson structure to the spreadsheet based on Jackson's method of mapping the data structure diagram to a computer program.

Figure 4 shows an example of a generic conceptual design containing graph sub-structures. For instance, there is a loop in the relationships connecting **E**, **G**, **I** and **K**, so that we no longer have a tree form. In this chart, **K** is a precedent of both **E** and **I**. We can turn the graph into a tree-structure. In order to accomplish this, two important steps prescribing the rules have to be observed:

Step 1: Each node or sub-structure with multiple dependants is duplicated so that each copy is assigned as a direct precedent of every dependant of that node. Nodes with multiple dependants can be easily identified from the conceptual design as they are represented by double-line boxes. This is illustrated in Figure 4.

By performing this task, the graph structure is resolved into a tree-structure. However, in order to prevent multiple occurrence of the entire sub-structure, only the root node of each duplicated sub-structure appears in the logical design of the model at this point. Their precedents are therefore not included in the model.

Based on *Figure 5*, **G** and **K** are duplicated in order to resolve the graph structure, into a tree structure. The precedents of **G** and **K** are not included in the model. **K** is not even shown as a precedent of **G** in order to comply with the rule that precedents of duplicated nodes are not included in the main structure of the logical design.

Step 2: If a duplicated node has precedents, a distinct structured module is created, the logical design of which is represented by a separate Jackson tree structure. The structured module consists of the duplicated node as its root node/formula and the precedents of the particular formula. The structures resulting from the application of this step/rule are illustrated in *Figures 6a* and *6b.*

If the duplicated node is a *leaf* and therefore has no precedents, there is no need to define it as a separate module. As a rule, only a node or formula with precedents, can be defined as a common module.

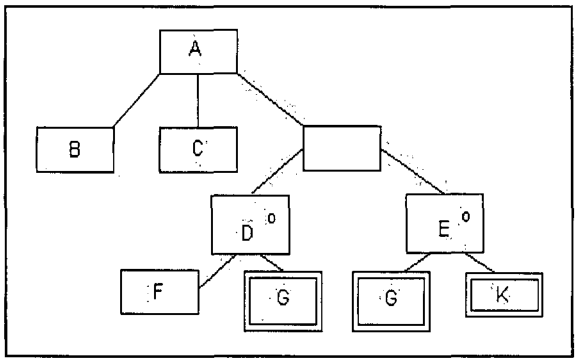


Figure 5: The logical design of the main structure based on *Step 1*

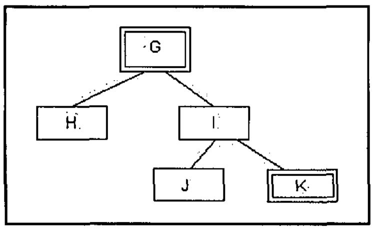


Figure 6a: The logical design of module G (based on *Step 2*)

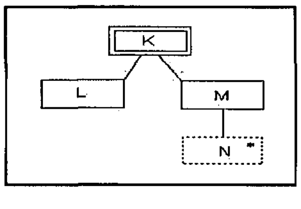


Figure 6b: The logical design of module K (based on *Step 2*)

Based on *Figures 6a* and *6b*, the sub-structures **G** and **K** are defined as separate modules, each of which will occur once in the implemented spreadsheet model.

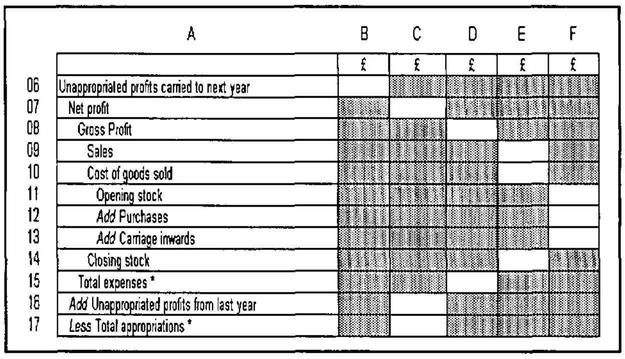
The conceptual design shown in *Figure 4* has now been transformed into a logical design consisting of three modules, represented by three separate Jackson structures. The modules consist of a main or primary module and two secondary modules.

In general, we can always reduce a graph structure to a tree by this method, which conveniently produces a unique modularisation of the spreadsheet model. Referring to our Trading and Profit and Loss Account example, the model does not contain any graph sub-structures. Therefore, this step is not applicable and can be omitted/skipped. In other words, the conceptual design of the model schema also represents its logical design.

Step 2: Physical design

Based on the logical design of the spreadsheet model, the location of the various elements or components of the model on the physical spreadsheet is determined. This is a low-level design of the spreadsheet model. The functions that operate on data values are specified using the right syntax and in terms of cell addresses as well, rather than just labels.

The logical design of the model (represented as Jackson tree-like structures) is systematically mapped onto the physical spreadsheet based on rigorous rules prescribed by the methodology. To maintain the structure modelled in the logical design in the spreadsheet view, the indentation principle is used, both on the row labels and on the corresponding values themselves. The values are indented by assigning a spreadsheet column to each level of indentation. These columns can be referred to as *virtual columns*. Based on the conceptual/logical design shown in *Figure 4*, the corresponding structure of the spreadsheet view at this stage is shown in *Figure 7*.

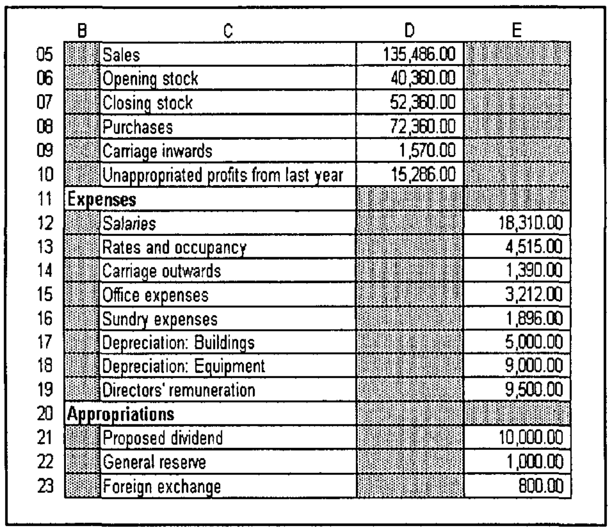


	A	B	C	D	E	F
		£	£	£	£	£
06	Unappropriated profits carried to next year		▨	▨	▨	▨
07	Net profit	▨		▨	▨	▨
08	Gross Profit	▨	▨		▨	▨
09	Sales	▨	▨	▨		▨
10	Cost of goods sold	▨	▨	▨		▨
11	Opening stock	▨	▨	▨	▨	
12	*Add* Purchases	▨	▨	▨	▨	
13	*Add* Carriage inwards	▨	▨	▨	▨	
14	Closing stock	▨	▨	▨		▨
15	Total expenses *	▨	▨		▨	▨
16	*Add* Unappropriated profits from last year	▨		▨	▨	▨
17	*Less* Total appropriations *	▨		▨	▨	▨

Figure 7: Outline of model schema

The input component for the model can now be created and all inputs entered in order to provide the model schema with the values required. This is done on a separate worksheet. The worksheet should be labelled *input*. Based on the logical design for the spreadsheet model, shown in *Figure 7*, the end-leaves can be implemented within an *input* component. This is shown in *Figure 8*.

There are reasons why cells for data input and assumptions should be grouped together in an input section, separate from the structured modules described above. One reason is to do with the utmost importance of obtaining accurate data entry. Kee (Kee-88) also states that using a central data entry area makes data entry easier and helps to prevent input errors. A second reason is that input cells are often referred to by more than one calculated cell. Apart from these reasons, it is also a precaution against any accidental overwriting of formulae. This strategy is similar to the method introduced by DiAntonio (DiAntonio-86). DiAntonio's method advocates the isolation of facts by splitting the spreadsheet into two parts, one for the *facts* and one for the *solution*. DiAntonio's *facts* part corresponds to our input component.



	B	C	D	E
05		Sales	135,486.00	
06		Opening stock	40,360.00	
07		Closing stock	52,360.00	
08		Purchases	72,360.00	
09		Carriage inwards	1,570.00	
10		Unappropriated profits from last year	15,286.00	
11	**Expenses**			
12		*Salaries*		18,310.00
13		Rates and occupancy		4,515.00
14		Carriage outwards		1,390.00
15		Office expenses		3,212.00
16		Sundry expenses		1,896.00
17		Depreciation: Buildings		5,000.00
18		Depreciation: Equipment		9,000.00
19		Directors' remuneration		9,500.00
20	**Appropriations**			
21		Proposed dividend		10,000.00
22		General reserve		1,000.00
23		Foreign exchange		800.00

Figure 8: Input component

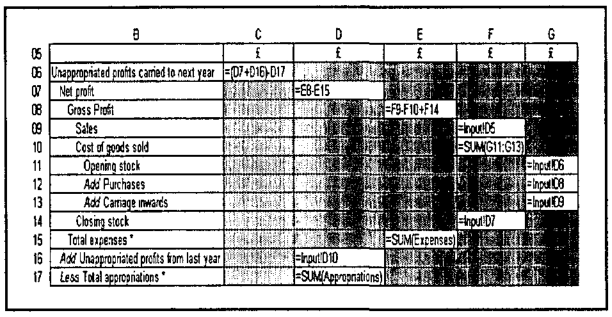
Stage 4: Implementation

Step 1: Spreadsheet system building

The physical design of the spreadsheet model is actually implemented using a particular spreadsheet package e.g. *Microsoft Excel*, *Lotus 123* etc.

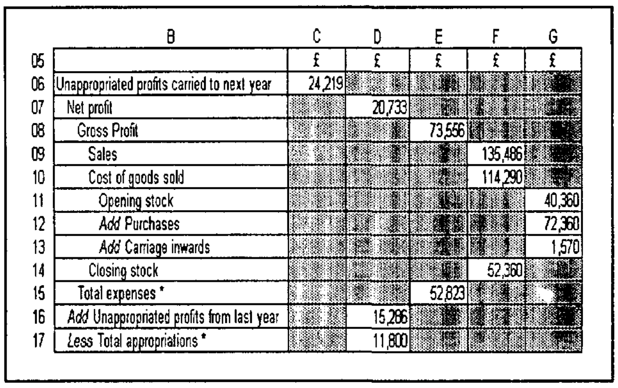
The formulae and binding relationships of the model can now be physically implemented or programmed in the model schema. References to inputs are first entered into the relevant cells in the model schema. This includes functions of input ranges, such as *total expenses* and *total appropriations*.

A bottom-up approach is taken in the implementation of formulae and relationships in the model schema. *Figures 9a (formula view)* and *9b (normal view)* show the final state of the model schema of the *Trading and Profit and Loss* Model.



	B	C	D	E	F	G
05		£	£	£	£	£
06	Unappropriated profits carried to next year	=(D7+D16)-D17				
07	Net profit		=E8-E15			
08	Gross Profit			=F9-F10+F14		
09	Sales				=Input!D5	
10	Cost of goods sold				=SUM(G11:G13)	
11	Opening stock					=Input!D6
12	*Add* Purchases					=Input!D8
13	*Add* Carriage inwards					=Input!D9
14	Closing stock				=Input!D7	
15	Total expenses *			=SUM(Expenses)		
16	*Add* Unappropriated profits from last year		=Input!D10			
17	*Less* Total appropriations *		=SUM(Appropriations)			

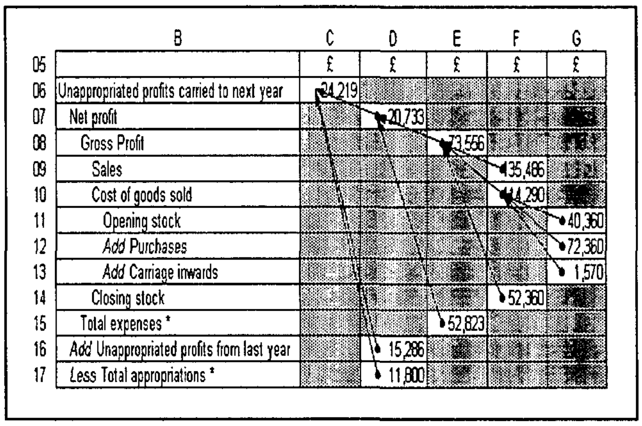
Figure 9a: Model schema (formula view).



	B	C	D	E	F	G
05		£	£	£	£	£
06	Unappropriated profits carried to next year	24,219				
07	Net profit		20,733			
08	Gross Profit			73,556		
09	Sales				135,486	
10	Cost of goods sold				114,290	
11	Opening stock					40,360
12	*Add* Purchases					72,360
13	*Add* Carriage inwards					1,570
14	Closing stock				52,360	
15	Total expenses *			52,823		
16	*Add* Unappropriated profits from last year		15,286			
17	*Less* Total appropriations *		11,800			

Figure 9b: Model schema (normal view).

Figure 10 presents a graphic representation of the dependencies between elements of the model schema. The logic of the model is easily comprehensible as it can be easily seen that each formula is a function of elements in the next virtual column.



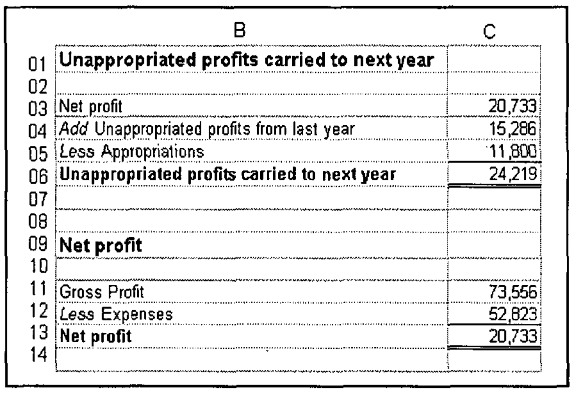
	B	C	D	E	F	G
05		£	£	£	£	£
06	Unappropriated profits carried to next year	24,219				
07	Net profit		20,733			
08	Gross Profit			73,556		
09	Sales				135,486	
10	Cost of goods sold				114,290	
11	Opening stock					40,360
12	*Add* Purchases					72,360
13	*Add* Carriage inwards					1,570
14	Closing stock				52,360	
15	Total expenses *			52,823		
16	*Add* Unappropriated profits from last year		15,286			
17	*Less* Total appropriations *		11,800			

Figure 10: Data dependencies.

It is beyond any doubt that the use of *indentation* and *virtual columns* make it far more straight-forward to make sense of and comprehend the composition of formulae. However, the fact that operands within a particular formula take the form of cell addresses rather than meaningful labels is not entirely desirable.

In order to further enhance the comprehensibility and integrity of the spreadsheet model, each data value and formula in the input component and model schema should be assigned a unique name. These names should then be used as operands within formulae, instead of cell addresses. The exception to this rule applies to a data value which is part of a related set of data that is always treated and operated on as a set, in which case it will be defined as a range along with the other related inputs.

References to corresponding *formulae* in the *model schema* and *data* in the *input component*, can now be entered into the relevant cells of the *output component*. The final state of the output component (based on the *Trading and Profit and Loss* model) is shown in *Figure 11*.



	B	C
01	**Unappropriated profits carried to next year**	
02		
03	Net profit	20,733
04	*Add* Unappropriated profits from last year	15,286
05	*Less* Appropriations	11,800
06	**Unappropriated profits carried to next year**	24,219
07		
08		
09	**Net profit**	
10		
11	Gross Profit	73,556
12	*Less* Expenses	52,823
13	**Net profit**	20,733
14		

Figure 11: Output component.

Step 2: Testing of the spreadsheet system

The testing process focuses on whether or not all the requirements of the system are met. The data values are checked for omitted or redundant items. All data values referenced by some formula must be present in the model. The formulae are checked to make sure that they produce the desired results. The tests uncover errors and ensure that defined input will produce actual results that agree with required results.

Various tools can be used to aid model testing such as the *Microsoft Excel Audit Tool, Spreadsheet Professional for MS Excel by Spreadsheet Innovations, Spreadsheet Auditor, Cambridge Spreadsheet Analyst, Cell/Mate, Microsoft Excel*'s Built In Auditing Functions, *Spreadsheet Detective, The Operis Analysis Kit (OAK)* and *Spreadsheet Auditing for Customs and Excise (SpACE).*

The *audit tool* which is part of the *Microsoft Excel* software enables the user to easily trace the **precedents** or *dependants* of any cell. The precedents of a cell are the cells *referenced by* it while the dependants of a cell are the cells that *reference* it. When tracing the precedents of a cell, an arrowed line is drawn from each precedent cell, pointing to the dependant cell. On the other hand, when tracing the dependants of a cell, an arrowed line is drawn from that cell pointing to each of its dependant cells.

Apart from that, the audit tool also offers a facility for the user to attach a note or description to a cell.

Spreadsheet Professional for Microsoft Excel, which is an add-on to *Microsoft Excel*, has also served as a rather useful auditing tool. It has various functions to help detect errors in the spreadsheet model. Among the significant functions of this tool are the *calculation checker* and the *cell translation* facility.

Chadwick et al (Chadwick-97) have proposed the **3A's** (appropriateness, accuracy, about-right) **Approach** for spreadsheet auditing. This method can be used to test the spreadsheet model. The detailed version of the methodology can be obtained from the source paper. The following are the elements of Chadwick et al's 3A's approach:

Step 1: Checking the appropriateness of the formula applied, from a logical point of view, based on the underlying business model.
Step 2: Checking the accuracy of the formula entered based on a correct interpretation of the data model.
Step 3: Checking if the resulting numeric value of the cell is about right.

Step 3: Document the spreadsheet system/program

Documentation may be included in a special area of the spreadsheet (on-line) or may be prepared in hard-copy form. It includes the program rationale and objectives. The author name and date prepared should also be part of the documentation. In addition to that, there should also be a log of all changes made to the spreadsheet model.

The Jackson structures described in the *analysis* and *design* stages can also be used as a means of documenting the spreadsheet model. These structures facilitate better comprehension of the spreadsheet model.

Step 4: Operation of spreadsheet system

Step 5: Post-implementation review of the spreadsheet system

Stage 5: Maintenance

The delivered spreadsheet system will undoubtedly have to undergo changes, due to advances in spreadsheet software as well as changing user requirements. Changes will also have to be made because of quantitative and qualitative errors that have been encountered. The system may also need to be adapted to accommodate changes in the external environment. System maintenance reapplies each of the preceding life-cycle stages and steps.

5 Conclusion

The Structured Spreadsheet Engineering Methodology is mainly based on the classical systems development life cycle presented by Aktas (Aktas-87), structured techniques and Jackson structures (Jackson-75). Adopting such an approach to spreadsheet design and development imposes great discipline in the spreadsheet building process. This is exactly what has been deemed important and necessary by many authors on the subject of spreadsheet errors and integrity control of spreadsheet models.

The proposed methodology has incorporated various tools, techniques, methods and principles for spreadsheet development, already published as well as new developments in the research into improved methods for integrity control in spreadsheet models. The integrated methodology also has the potential to prevent and reduce most of the errors given in the taxonomy of errors in section 2.3.

The principal objective of a structured and disciplined methodology for the construction of spreadsheet models is to reduce the occurrence of user-generated errors in the models.

In order to assess and establish the quality of the methodology, four different experiments have been carried out. The results of these experiments have been published (Rajalingham-01). They provide adequate evidence of the methodology's potential for controlling the integrity and improving the comprehensibility of spreadsheet models. A more detailed version of the complete set of experiments carried out and a thorough analysis of their results will be published soon.

References

[Aktas-87]
Aktas A Z (1987). *Structured Analysis & Design of Information Systems*, Prentice-Hall.

[Chadwick-97]
Chadwick D, Knight J and Clipsham P (1997). *Information Integrity In End-user Systems*, Proceedings of the IFIP TC-11 Working Group 11.5 First Working Conference on Integrity and Internal Control in Information Systems (December 1997), Zurich, Switzerland.

[Chadwick-99]
Chadwick D, Rajalingham K, Knight B and Edwards D (1999). *A Methodology for Spreadsheet Development Based on Data Structure*, CMS Press (June 1999), No 99/IM/50.

[DiAntonio-86]
DiAntonio A E (1986). *Spreadsheet Applications*, Prentice-Hall.

[Isakowitz-95]
Isakowitz T, Schocken S and Lucas H C J (1995). *Toward a Logical/Physical Theory of Spreadsheet Modeling*, ACM Transactions on Information Systems, Vol 13(1), pp1-37.

[Jackson-75]
Jackson M A (1975). *Principles of Program Design*, Academic Press.

[Kee-88]
Kee R (1988). *Programming Standards for Spreadsheet Software*, CMA Magazine (April 1988), Vol 62, No 3, pp55-60.

[Knight-00]
Knight B, Chadwick D and Rajalingham K (2000). *A Structured Methodology for Spreadsheet Modelling*, Proceedings of the EuSpRIG 2000 Symposium on Spreadsheet Risks, Audit and Development Methods (17-18 July 2000), Greenwich, London: University of Greenwich, pp43-50.

[KPMG-97]
KPMG (1997). *Executive Summary: Financial Model Review Survey*, KPMG (London).

[Kruck-98]
Kruck S (1998). Towards a Theory of Spreadsheet Accuracy: An Empirical Study. Paper delivered at the Decision Sciences Institute, Las Vegas, USA (November, 1998).

[Panko-96]
Panko R R and Halverson R P, Jr. (1996). *Spreadsheets on Trial: A Survey of Research on Spreadsheet Risks*, Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences (2-5 January 1996), Maui, Hawaii.

[Panko-98]
Panko R R (1998). *What We Know About Spreadsheet Errors*, Journal of End User Computing (Spring 1998), Vol 10, No 2, pp15-21.

[Rajalingham-98]
Rajalingham K and Chadwick D (1998). *Integrity Control of Spreadsheets: Organisation & Tools*, Proceedings of the IFIP TC11 WG11.5 Second Working Conference on Integrity and Internal Control in Information Systems (19-20 November 1998), Virginia, USA: Kluwer Academic Publishers, pp147-168.

[Rajalingham-99]
Rajalingham K, Chadwick D, Knight B and Edwards D (1999). *An Approach to Improving the Quality of Spreadsheet Models*, Proceedings of the Seventh International Conference on Software Quality Management SQM'99 (March 1999), Southampton, United Kingdom: British Computer Society, pp117-131.

[Rajalingham-99a]
Rajalingham K, Chadwick D, Knight B and Edwards D (1999). *An Integrated Spreadsheet Engineering Methodology (ISEM)*, Proceedings of the IFIP TC11 WG11.5 Third Working Conference on Integrity and Internal Control in Information Systems (18-19 November 1999), Amsterdam, The Netherlands: Kluwer Academic Publishers, pp41-58.

[Rajalingham-00]
Rajalingham K, Chadwick D, Knight B and Edwards D (2000). *Quality Control in Spreadsheets: A Software Engineering-Based Approach to Spreadsheet Development*, Proceedings of the Thirty-Third Hawaii International Conference on System Sciences (4-7 January 2000), Maui, Hawaii: IEEE Computer Society, CD-ROM.

[Rajalingham-00a]
Rajalingham K, Chadwick D and Knight B (2000). *Classification of Spreadsheet Errors*, Proceedings of the EuSpRIG 2000 Symposium on Spreadsheet Risks, Audit and Development Methods (17-18 July 2000), Greenwich, London: University of Greenwich, pp23-34.

[Rajalingham-00b]
Rajalingham K, Chadwick D and Knight B (2000). *Classification of Spreadsheet Errors*, British Computer Society (BCS) Computer Audit Specialist Group (CASG) Journal (Autumn 2000), Vol 10, No 4, pp5-10.

[Rajalingham-01]
Rajalingham K, Chadwick D and Knight B (2001). *An Evaluation of the Quality of a Structured Spreadsheet Development Methodology*, Proceedings of the EuSpRIG 2001 Symposium on Controlling the Subversive Spreadsheet - Risks, Audit and Development Methods (5-6 July 2001), Amsterdam, The Netherlands: Vrije Universiteit, pp39-59.

[Ward-97]
Ward M (1997). *Fatal Addition*, New Scientist (16 August 1997).

[Wood-96]
Wood F (1996). *Business Accounting 1 (Seventh Edition)*, Pitman Publishing.

Trojan horse attacks on software for electronic signatures

Adrian Spalka, Armin B. Cremers and Hanno Langweg
Department of Computer Science III, University of Bonn
Roemerstrasse 164, D-53117 Bonn, Germany
Fax: +49-228-734 382, Email: adrian@cs.uni-bonn.de

Keywords: Trojan Horses, Electronic Signatures, Security of COTS Software

Received: January 23, 2002

Electronic signatures are introduced by more and more countries as legally binding means for signing electronic documents with the primary hope of boosting e-commerce and e-government. Given that the underlying cryptographic methods are sufficiently strong, attacks by Trojan horse programs on electronic signatures are becoming increasingly popular. Most of the current systems either employ costly or inflexible – yet still inadequate – defence mechanisms or simply ignore the threat. A signatory has to trust the manufacturer of the software that it will work in the intended way. In the past, Trojan horse programs have shown to be of growing concern for end-user computers. Software for electronic signatures must provide protection against Trojan horses attacking the legally relevant signing process. In a survey of commercial of the shelf signature software programs we found severe vulnerabilities that can easily be exploited by an attacker. In this work we propose a secure electronic paper as a counter-measure. It is a collection of preventive and restorative methods that provides, in parallel to traditional signatures on paper, a high degree of protection of the system against untrustworthy programs. We focus our attention on Microsoft Windows NT and Windows 98, two operating systems most likely to be found on the customers' computers. The resulting system is an assembly of a small number of inexpensive building blocks that offers reliable protection against Trojan horse programs attempting to forge electronic signatures.

1 Introduction

Electronic signatures are introduced by more and more countries as legally binding means for signing electronic documents with the primary hope of boosting e-commerce and e-government. While business-to-business (B2B) e-commerce is regarded as an overwhelming success, other areas experience a hard start-up time. Companies involved in business-to-consumer (B2C) e-commerce are hit particularly hard – only but a few are likely to be profitable. One of the many circumstances to which this misery is attributed to is the customer's lack of confidence in the reliability and potential for misuse of business transactions carried out over the Internet. The hope for a turn-around rests to a large extent on electronic signatures, which should provide dependable information on the identity of the parties engaged in a transaction.

Due to its name an electronic signature is supposed to be an electronic counterpart to a human signature, which can be embedded in an electronic identity card. It should be stressed that the envisioned owners of an electronic signature are not only a few specialised users but every ordinary person who has access to a computer. Once it attains the same legal status as a human signature, the consequences of signing an electronic message will be the same as those of signing manually a paper document.

An electronic transaction carried out over the Internet consists of a set or a sequence of messages sent among the parties participating in it. By signing a message the party confirms that it approves this message. The other parties (to whom this message is made available) have then the opportunity to check the genuineness of the signature, ie, the authenticity of the message.

Today's electronic signature schemes use strong cryptographic algorithms, usually based on integer factorisation, the discrete logarithm problem or elliptic curves. With appropriately selected parameters these algorithms are practically unbreakable, which means that an attacker who is not in possession of the private key is not able to compute the signature of a document.

Given that, care has been taken to ensure that an attacker also cannot steal the private key from the user's computer. The idea is to store the private key together with the signing function on a separate signature creation device, eg, a smart card. Supplied with the document this card computes as output its electronic signature – it never returns the private key. Compromising a smart card is, if at all feasible, a technically very challenging process and we can safely assume that an attacker has no command of it.

In view of this setting we can say that an electronic signature cannot be forged by breaking the cryptographic algorithms or by compromising the smart card. Yet we still cannot claim that the signature cannot be forged at all. The reason for this are Trojan horse programs, ie, programs that perform malicious functions invisible to the user. Today, Trojan horse programs come in at least two varieties: as stand-alone programs and as macros embedded in the document. They are not new. But while their creation and distribution required a lot of knowledge and effort in former times, today they can be easily programmed and the Internet is an excellent way for their distribution.

If an attacker knows that he cannot break a component he is most likely to misuse it. The process of creating a signature is a chain of several steps. The user first creates the document; she then sends it to the signature software, which communicates with the smart card in the card terminal; the result is finally passed back to the user. The interfaces between these steps are weak in the sense that a component does not know if it actually receives the input intended by the user. Here, our anticipation goes hand in hand with our experience in that a Trojan horse program will attempt to manipulate or swap a document before it is signed.

One can be tempted to believe that laws concerning the design of the signing process have taken this threat into account by forcing the user to review her document in a 'secure viewer' provided by the signing environment. Our examination, however, reveals that malicious code can easily attach a hook at this point, too. Of course, in the first place one can attribute this weakness to the design of the Windows operating system, the one to be found on most of the prospective customers' computers. But we claim that this shift of responsibility is only too readily hinted at by the manufacturers.

This work commences with the examination of some products and proposals for the processing of electronic signatures on personal computers. It turns out that two extreme assumptions about the trustworthiness of the customer's computer are made. The first one requires the customer to take care that no rogue programs are on her computer (a clearly unreasonable requirement in a home environment, ie, it assumes that all programs are trustworthy. The second one requires the customer to perform tedious calculations on a piece of paper (a similarly unreasonable requirement given that many customers are struggling with their VCR), ie, it assumes that the customer's computer is wholly untrustworthy.

We examine a couple of signature software programs available in Germany on their susceptibility to Trojan horse attacks. The five programs include both major and minor players on the German signature software market. Some software manufacturers claim that they provide higher protection because of their implementation of 'what-you-see-is-what-you-sign'[1] [2]. In our analysis we focus on some of the most likely attacks Trojan horse programs may perform on software for the creation of electronic signatures. Following the examination we give recommendations on how to avert these attacks and present our approach of a secure electronic paper.

In its design, our approach models the steps and properties of manually signing a piece of paper. We assume that some components of the operating system, eg, device drivers and, if present, access controls, are reliable and trustworthy. The central result of our analysis is the use of a write-once-read-multiple device that, supported by some other components, fixes the intermediate results in the process of creating a signature. With a little more effort the user can either ensure that she actually signs the intended document or she can detect a forgery before the document leaves her computer. Admittedly, this point of view does not totally prevent malicious code from doing damage to an electronic signature, but it requires a lot of effort on the attacker's side to bring the Trojan horse program in place and still remain undetected. We describe in detail an implementation for the Windows operating system and give some hints on its use in other operating systems.

2 Previous and related works

Currently there are three different approaches to facilitate the use of digital signatures in insecure environments. We refer to them as 'secure hardware', 'mental arithmetic', and 'secure software'.

While the first two approaches yield a provably high strength against Trojan horse attacks they are expensive, difficult to use, and not flexible regarding the data they are able to sign. The latter 'secure software' approach usually disregards Trojan horse attacks in current implementations. Nearly all surveyed products did not bother to include protective measures against malicious processes on the same computer. On top they are difficult to use and inflexible.

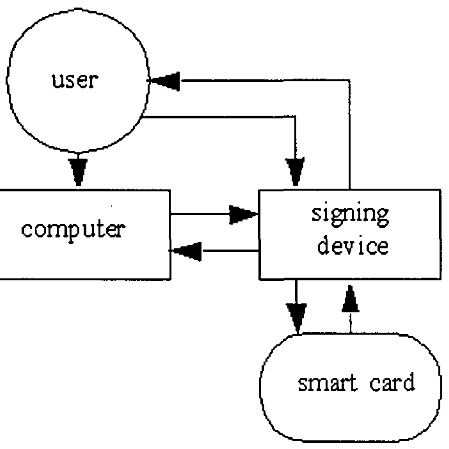
All three existing approaches force the signatory to explicitly review the data that is going to be signed before it is processed. The user must do this even if she just finished working with it in her application software. This sharply reduces the ease-of-use.

2.1 Using secure hardware devices

This approach is favoured by a large number of academic institutions and pursued by Cryptovision GmbH of Germany. They propose a device that comprises a liquid crystal display (LCD), a smart card reader, and a certified circuit board that contains the operating logic for the signing device; costs are estimated to be less than five thousand Euro each. The flow of information is shown in the sketch.

[1] Deutsche Post AG (2002). 'How do I digitally sign data or a message? [...] Your message is then displayed by the integral SIGNTRUST Mail viewer so that you can be sure you are only signing what you have seen and accepted on your screen.' http://www.signtrust.de/service/faq/details.php?id=3&lang=1

[2]Utimaco Safeware AG (2001). 'The viewer is a completely independent component for displaying text data and guarantees that only what is seen on the monitor will be signed with the motto: "What you see is what you sign" and "What you see was signed".' *SafeGuard Sign & Crypt FAQ.*



The computer is used as the provider of the information that shall be signed. Since the computer is assumed as completely insecure the signing device does not get the correct data if this has been manipulated by a Trojan horse on the computer. The signing device and the signature smart card are the only trustworthy components in the eye of the user. So the user has to review the data the signing device received to verify that it is the data she indeed wants to sign.

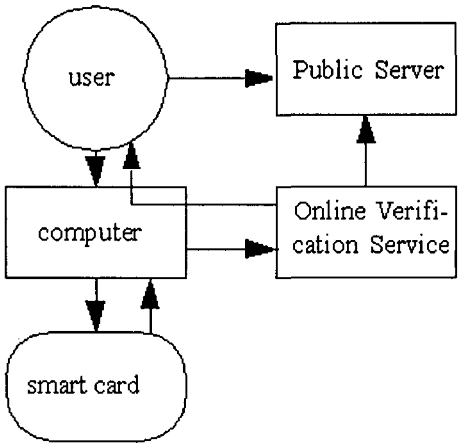
The data is displayed in a standardized way, eg, rich text format or common word processor file formats without advanced options. This implies that the presentation on the signing device does not always match the presentation on the signatory's computer. The receiver of the signed data may need the same device or a software viewer that displays the data like the signing device does. The signatory either confirms or rejects the presentation. After confirmation the data is sent to the signature smart card that actually computes the signature. The signature is then being transmitted through the device to the user's computer.

A Trojan horse that modifies the data before it reaches the signature smart card is detected by the user because she will note any modification of the data. Since the presentation and confirmation take place on a trustworthy device the signature is computed for exactly the data the signatory wants to get signed. The weakest component in this approach is a lazy user who does not review the data for reasons of convenience.

Drawbacks of the system are the small (and fixed) number of accepted file formats, the restriction to data that can be displayed on an LCD (eg, no audio data), the high costs that exceed the price for most personal computers, the high expenditure to roll out updates, and the reduced ease-of-use that diminishes the understanding and the support by the prospective users.

2.2 Neglecting arithmetically challenged users

To avoid the high costs of additional hardware on the signatory's side Stabell-Kulø introduced an approach that forces the signatory to compute simple cryptographic operations by mental arithmetic. The user's computer is regarded as completely insecure and hence can not be trusted. A Trojan horse can alter every communication between the user and components connected to the computer. So it is proposed to introduce an 'Online Verification Service' and a 'Public Server' to the PKI, and a One-Time-Pad and a substitution table for the user.



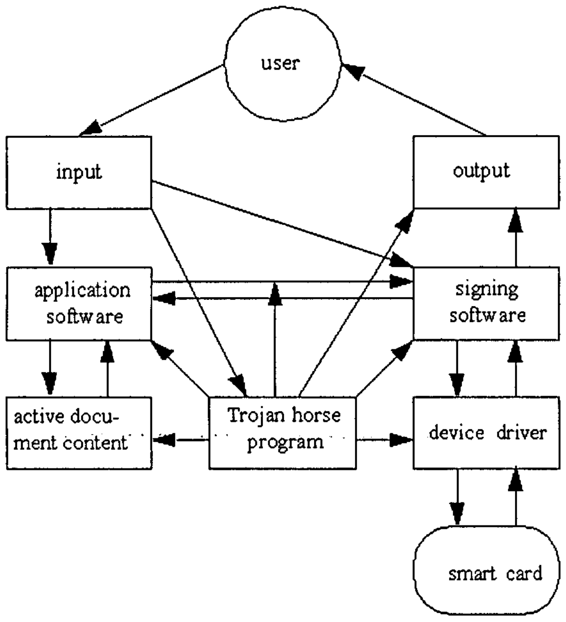
A signature computed by the smart card is sent to the (trusted) Online Verification Service. This service verifies that the signature matches the data for which it has been computed. It then encrypts the data by applying the substitution table and the One-Time-Pad and transmits this encrypted information through the insecure computer to the user; the signature is sent to the Public Server and marked as 'not yet released'. The signatory applies the One-Time-Pad and the substitution table to the encrypted information and retrieves the decrypted data. She compares if the decrypted data matches the data she wanted to sign. Decryption is done without the untrusted computer and takes approximately one minute per 15 characters if the user is a computer science student; otherwise the decryption rate averages 7.5 characters per minute. We did not find a signing method that has a lower ease-of-use. If the user agrees with the signed data she sends a release command to the Public Server that involves the use of a hash value.

While the advantage of obtaining a reliably computed signature in an insecure environment is not bad, the drawbacks are clearly disenchanting: the volume that can be signed is very low, the approach is not suitable for arithmetically challenged people, you have to produce and securely distribute One-Time-Pads and substitution tables, and you have to introduce an Online Verification Service and a Public Server to the Public Key infrastructure.

2.3 Implementing 'secure' software

This is the most common approach found in commercial products that are already being shipped to customers. We found that even leading companies disregard the threats posed by Trojan horses. One of the presumed market leaders was vulnerable to basic attacks. When asked why they did not use protection against Trojan horses they responded that it lies in the responsibility of the user to avoid the execution of untrusted processes on her computer. That is simply not suitable for a product used by inexperienced people.

The user works with an application software and at some point decides to sign the data. The data is then transmitted from the application software to the signature software, displayed for confirmation by the signatory, and finally sent to the signature smart card that computes the signature.



In most implementations a Trojan horse program has access to many interfaces it can attack: inside the application software as active document content, between the application software and the signature software, the signature software itself can be a target as well as the device driver between signature software and smart card.

The signature software displays the data that is going to be signed in a standardized way and prompts the user for confirmation. All but a few manufacturers assume that Trojan horses will not attack their software or explicitly put the responsibility for a Trojan horse-free environment in the signatory's domain.

In contrast to the first two approaches this is a low-cost approach without additional hardware devices or PKI components. However, it still has a low ease-of-use since the data has to be reviewed once more even if the user has worked with it for a long time in the application software. And no company has made efforts to effectively block Trojan horse attacks on a conceptual basis.

3 Secure Software

Most of the previous approaches to secure the creation of digital signatures focus on the cryptographic algorithms and the secure computation in a smart card. While these problems appear to be solved, the smart card still does not communicate directly with the signatory. The smart card relies on a software that is executed on the signatory's computer to transfer the data that is going to be signed by the card.

Approaches that introduced new and expensive hardware have not proved competitive in the market. Hence, many companies developed signature software products that rely on the secure execution of their software.

We asked manufacturers if and how they protect their software against attacks by Trojan horse programs. The answers we got on trade fairs and in personal communication were that protection would be 'surely interesting but expensive if not impossible'. In fact, the protective measures applied seemed more to decrease ease-of-use than to prevent attacks.

In this section we verify the manufacturer's statements.

3.1 Types of likely attacks

In our study we concentrate on four vulnerabilities a Trojan horse program will be likely to exploit. The first is capturing the PIN code for accessing the signatory's private signing key. Second, we observe if the data can be modified between finishing work in the application software and processing it in the signature software. Some products require the user to review the data before it is actually sent to the signature smart card. In that case a malicious program would have to alter the displayed data for review to avoid detection. Hence, we will treat the second and third vulnerability as a single one. The last problem we focus on is a man-in-the-middle attack on the smart card terminal device driver.

We do not present a comprehensive risk analysis and vulnerability report for all signature products available. This is an obligation of the software manufacturers who claim that their products are secure and who keep saying 'what you see is what you sign'. By concentrating on some of the worst problems we show that the vulnerabilites are due to a flawed design.

3.1.1 Capturing the PIN for access to the PSE

For increased security of the storage of the signatory's private signing key most people advocate the use of a smart card. The card is called a *Personal Security Environment* (PSE) because it is in the possession of the user who usually presents a PIN to the card for authentication purposes. Since it is assumed that the signatory will not lose both her smart card and reveal the corresponding PIN this method is regarded reliable.

In current implementations of this protocol the smart card does not know if the PIN is provided by the user or by a third party, eg, an attacker. If an attacker gains access to the PIN and the user inserts her signature smart card into the smart card reader attached to the computer, the attacker could establish communication with the card, provide the PIN and after successful authentication begin signing messages with the signatory's private key.

Most programmers tend to be lazy when coding security features that are not specified sufficiently. In case of a PIN entry on the Microsoft Windows platform, they use a standard edit control and set its property *PasswordChar* to '*'. This partial modification of behaviour allows a user to type in her PIN or password while displaying asterisks for every character typed. However, displaying asterisks does not suffice, since the typed characters are still provided by the edit control. We will show in a later section how the PIN is retrieved in detail.

3.1.2 Modifying the data to be signed

The user works with her application software to create and modify the data she wants to sign. At some point she decides to finish her work and sign the data. The document is then transferred from the application software to the signature software, sometimes displayed again for confirmation, and finally transmitted to the signature smart card that computes the signature.

In most implementations a Trojan horse program has access to many interfaces it can attack:

- inside the application software as active document content
- between the application software and the signature software
- the signature software itself can be a target as well as the device driver between signature software and smart card.

When the manufacturer proposes an additional confirmation step before sending the data to the signature smart card, the signature software displays the data that is going to be signed in a standardised way and prompts the user for acceptance. All but a few manufacturers assume that Trojan horses will not attack their software or explicitly put the responsibility for a Trojan horse-free environment in the signatory's domain.

We will show that modification of the display in a so-called secure viewer component is possible in a commercial off the shelf software product by one of the market leaders for signature software in Germany.

3.1.3 Interfering with the communication between software and PSE

When the smart card receives data for signing with the signatory's private signing key it does not know if this data originated from the signatory. It requires a PIN to be sent to verify that the signatory is present when the data is received. After authentication the smart card usually receives a hash value of the data that the signature is expected to be computed on.

An attacking Trojan horse program that places itself between the signature software and the signature smart card can observe the communication between the two parties. After the PIN has been sent to the card the malicious program could alter the communication and send different data to the card than that the user is expected to getting signed.

We show in a later section that one of the products examined accepted a card terminal driver that was not provided by the operating system.

3.2 Surveyed products

We chose the products by a simple selection process. They had to be easily available for purchase in Germany and should be able to work with a smart card. The first requirement turned out to be not so easy to fulfil as we had thought beforehand. Even the products of the presumed market leaders Deutsche Post and Deutsche Telekom took effort to persuade employees that the software was for sale and they needed some weeks to arrive.

According to surveys we conducted on trade fairs (CeBIT 2000 & 2001, Hanover; Systems 2000, Munich), none of the other software manufacturers perceived Trojan horse programs a threat for their signature software solutions and, hence, did not protect the signatory against such attacks. So while the selection is not representative in the strict sense of the word, it gathers products with typical vulnerabilities. The tests were conducted on the Microsoft Windows 98 and 2000 platforms.

3.3 eTRUST/Signtrust Mail[3]

Deutsche Post AG, the former state-owned postal service, launched it's electronic signature solution on the CeBIT 2000 trade fair. Three months after, we bought a starter kit that consisted of a signature smart card, registration with the manufacturer's trust centre, a smart card reader, and the signature software *eTRUST* 1.01.

eTRUST is designed to be integrated into the Microsoft Outlook email software. The user writes an email, selects 'sign', then 'send'. The email is then presented to the user (again) for confirmation. After confirmation the user is required to provide the PIN for access to the signature smart card. The email message is sent to the card, signed, and the signature is attached to the email.

There are a couple of Trojan horse program-related problems with this product. We are able to obtain the PIN for the card, can modify the data that is going to be signed without knowledge of the signatory, and *eTRUST* accepts our smart card reader driver for communication with the smart card. Hence, we use this product as an example and cover the problems in detail.

3.3.1 Asking for the PIN - and getting it

Obtaining the smart card's PIN is done with standard Trojan horse methods. A similar attack had received broad attention by the media in 1998. Deutsche Telekom's Internet service provider T-Online had been shown to be vulnerable to password retrieval. The access password was stored in a standard edit control and was protected only against visual attacks by displaying asterisks. Nevertheless the password could be made visible or retrieved by another program with virtually no effort. We thought that a newly-developed program two years after the embarrassing T-Online incident would provide at least basic protection against this attack. T-Online had modified their password input and is no longer vulnerable to this kind of attack.

[3] Meanwhile, the product name has been changed to 'Signtrust Mail'.

Our attack takes place when the user has finished entering her PIN and before she clicks 'proceed'. The first step is to obtain a handle to the PIN edit control, then kindly ask for the PIN in it:

```
(relevant code in Delphi)

hWindow:=GetWindow(
            Self.Handle,
            GW_HWNDFIRST);
While (hWindow <> 0) do
Begin
  If (hWindow <> Self.Handle) and
     IsWindowVisible(hWindow)
  Then Begin
     GetClassName(
        hWindow,
        szClass,
        SizeOf(szClass));
     GetWindowText(
        hWindow,
        szText,
        SizeOf(szText));
     If (StrPas(szClass) = '#32770')
         and
        (StrPas(szText) =
           'PIN-Eingabe')
     Then Begin  // Found PIN window
        hWindow:=GetWindow(
                    hWindow,
                    GW_CHILD);
        While (hWindow <> 0) do
        Begin
           GetClassName(
              hWindow,
              szClass,
              SizeOf(szClass));
           If StrPas(szClass) = 'Edit'
           Then Begin
              // Found PIN edit control
              SendMessage(
                 hWindow,
                 WM_GETTEXT,
                WPARAM(SizeOf(szPIN)),
                 LPARAM(@szPIN));
              // Show PIN
              editCapturedPIN.Text:=
                 StrPas(szText);
              hWindow:=0;
           End
           Else hWindow:=
                   GetWindow(
                      hWindow,
                      GW_HWNDNEXT);
        End;
        hWindow:=0;
     End;
  End;
  If hWindow <> 0
  Then hWindow:=
         GetWindow(
            hWindow,
            GW_HWNDNEXT)
End;
```

We iterate through the windows on the Windows desktop. Once we have found the window for PIN entry, we iterate through the controls in that window. Luckily, the PIN entry control is the only control of the *Edit* type. We then send a *WM_GETTEXT* message to the control and capture the PIN into our *szPIN* buffer. For our purposes we just show the PIN in our application. We can as well start communicating with the signature smart card to create signatures with the signatory's private signing key.

In Microsoft Windows 2000 the *WM_GETTEXT* message will only get a response if it is sent from within the same application. This is a step forward. Becoming part of the signature application is nevertheless possible. We put a modified *WINSCARD.DLL* in the *eTRUST* folder. Upon accessing the DLL by *eTRUST* we start a separate thread to capture the PIN.

3.3.2 Modifying the secure viewer's presentation

The next component we targeted was the so-called secure viewer. It is central to the manufacturer's what-you-see-is-what-you-sign concept. Since modification of the data before it is processed by the signature software is not prevented, the data is presented to the user for confirmation. So, if a malicious email software or plug-in has altered the data in a way the signatory does not want it to be, she can decline confirmation and thereby be prevented from signing unwanted documents.

The data presented in the viewer can be easily modified. Here is how:

```
(relevant code in Delphi)

// Get handle of secure viewer's window
// Class '#32770'
// Title 'Visualisierung der Email'

hWindow:=GetWindow(hWindow,GW_CHILD);
While (hWindow <> 0) do
Begin
  GetClassName(
     hWindow,
     szClass,
     SizeOf(szClass));
  If StrPas(szClass) = 'RICHEDIT'
  Then Begin
     // Found viewer's control
     SendMessage(
        hWindow,
        EM_SETREADONLY,
        WPARAM(false),0);
     // insert eg text from clipboard
     // into the secure viewer
     SendMessage(
        hWindow,
        EM_SETSEL,
        WPARAM(0),
        LPARAM(-1));
     SendMessage(
        hWindow,
        EM_PASTESPECIAL,
        WPARAM(CF_TEXT),
        LPARAM(0));
     SendMessage(
        hWindow,
        EM_SETREADONLY,
            WPARAM(true),0);
          hWindow:=0;
        End
        Else hWindow:=GetWindow(
                          hWindow,
                          GW_HWNDNEXT);
      End;
```

Like in the PIN capturing example, we iterate through the handles to obtain the handle for the desired control. The protection applied here by the manufacturer helps against the user typing in the viewer, since the *RICHEDIT* control is set to *read-only* state. We send a message to the control and ask for dropping the read-only restriction. Then we select all the text in the control, and override the selection with the text we earlier copied to the desktop's clipboard. Afterwards we set the control's state back to read-only.

3.3.3 Monitoring communication between software and smart card

After the user has entered the PIN for accessing her signature creation device, ie, the signature smart card, the data to be signed is sent to the card. We place a library in the installation folder of *eTRUST* to monitor the communication with the card.

We provide the file *WINSCARD.DLL* for PC/SC-compliant smart card communication in the folder where the *eTRUST* program files are stored. Obviously, the signature software does not load the DLL from the Windows system folder but examines the search path in standard order. Hence, we are able to observe and modify communication between the *eTRUST* software and the signature smart card.

The smart card provided by Deutsche Post uses some proprietary commands we have not yet identified. But since the PIN is provided with secure messaging and the hash value that is to be signed is not, we assume that we could have altered the hash value to make the card sign the data an attacker wants to be signed contrary to the signatory. The manufacturer did not deny that.

3.3.4 Comments by the manufacturer

When confronted with the results of our analysis, Deutsche Post neither confirmed nor denied any of the vulnerabilities. They retreated to the position that every software running on the Microsoft Windows platform was susceptible to Trojan horse attacks and it is Microsoft's and the user's responsibility to provide protective measures or a Trojan horse-free computer.

In the manual for *eTRUST Mail* the user is advised to check that her personal computer cannot be manipulated by others and that there are no malicious programs on her computer. In case there are, Deutsche Post will not make any statement concerning the integrity of the signing process. On the other hand they justify their 'secure viewer' component with the possibility that some program might try to sign data different from the one the user wants to sign.

We think this is not sufficient for a product that targets a mass market. A company that wants to become market leader in electronic signature software solutions must offer its customers a product they can rely on. Saying that Trojan horse programs may exist but are the signatory's business is not user-friendly.[4]

3.4 Utimaco SafeGuard Sign&Crypt

Sign&Crypt is a product of Utimaco Safeware AG. They were one of the first companies in Germany to offer a signature software with a viewer component. Their homepage named two distributors for their products. One of them was able to ship.

We first tried to retrieve the PIN from the *Edit* control used for input. It displays asterisks to shield the input and it is possible to navigate the cursor with the arrow keys to edit the PIN. Sending a WM_GETTEXT message yields '####', so they prevent this easy attack.

The viewer component gets it input by submitting output to the 'Digital Signature' virtual printer. Thus a Trojan horse knows that this data is going to be presented in the viewer component and signed afterwards. It is possible to alter the display in the viewer component, but the component re-draws the displayed data frequently. Hence, it is difficult to forge a different presentation.

To access the smart card terminal Utimaco provides a proprietary service application. Thus, we did not look at the communication between software and card.

3.5 T-Telesec PKSCrypt

Deutsche Telekom is Germany's formerly state-owned telephone company. They are the market leader in the telecommunications market in Germany and have plenty of top-educated and highly-skilled employees. Their security subsidiary T-Telesec offers an electronic signature software with smart card integration since 1998 and is presumably one of the first companies to do so. Hence, we expected to easily get a first-class product.

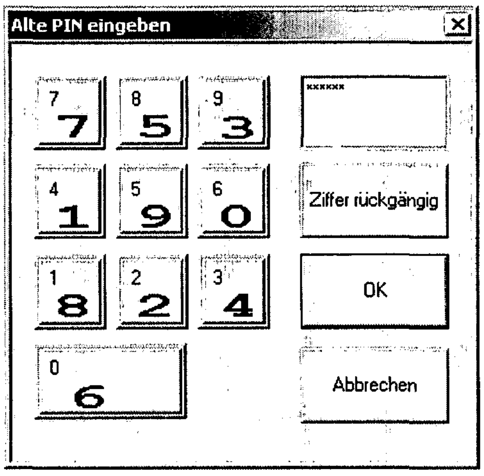
Buying the software was not so simple. We had to persuade some employees that Deutsche Telekom really offers that product for sale and that they surely have the correct forms to fill out for the application for a signature smart card. Some weeks later we actually received the software. An email plug-in is not included, but the software integrates neatly with the Microsoft Explorer to sign files with a click of the right mouse button.

The package comes with a signature smart card that has to be initialised with a user-chosen PIN. The card is initially in a state that it will only accept a first PIN as a command. The first thing we did was to examine the PIN entry. *PKSCrypt* 1.11 offers two ways to enter the PIN. The first is called 'standard input', the second 'secure input'. We checked both.

With 'standard input' the user is supplied with a standard password input window that displays asterisks instead of numbers when she enters her PIN. This sounds familiar, and it takes no effort to kindly ask the software

[4] Deutsche Post has issued a new version of their software which allegedly fixes the problems mentioned here. However, the new version did not arrive in time to be included in this survey.

for the PIN like we explained in detail for *eTRUST.* (Did we mention that Deutsche Telekom's subsidiary T-Online got a lot of embarrassing attention three years ago for a similar lazy implementation in another product?). By the way, 'standard input' is the default setting in *PKSCrypt.*



The alternative option of 'secure input' impresses the user with a numeric keypad on the screen that shows a permutation of the ten number keys. The user is supposed to enter the number next to the number her PIN contains. In our example screen shot we would enter '429380' if our PIN was '123456'. Since the permutation is different each time, a Trojan horse can ask for the PIN but it will never get the correct one (unless, of course, the permutation is the identity function, but the developers will certainly have thought of that).

We thought this would going to be hard, that we would have to extract the permutation with image recognition techniques from a screen shot. But it is simpler than that. The window consists of the edit control with the permuted PIN, some buttons and a lot of controls of the *Static* type. We get the permutation as easy as the PIN. Yes, we have to write some dozen more lines of code, but that is it.

```
Static ( 26, 41)-( 52, 59) "7"
Static ( 23, 28)-( 41, 43) "7"
Static ( 77, 41)-(103, 59) "5"
Static ( 74, 28)-( 92, 43) "8"
...
Static ( x1, y1)-( x2, y2) "n"
```

The *Static* controls contain a number each. By their position, which is also provided, we can determine the correct permutation.

Since the permutation of the input aims at an attacker who can access the PIN edit control, it is not understandable why the attacker is provided with the permutation. In a comment of the manufacturer Deutsche Telekom they said that the intent of the 'secure input' was to protect the PIN from attackers who watch the user entering it or who use a camera aimed at the keyboard.

They use the 'CT' API for communication with the smart card reader. This is a standard fairly common in Germany; it is not compatible with the PC/SC standard. Thus, we had to write another dynamic link library file to intercept messages sent between application and signature smart card – only to find out that Deutsche Telekom had done a good job and had secured the important messages with secure messaging according to ISO 7816.

3.6 Siemens/SSE TrustedMIME

The steps needed to sign an email with *TrustedMIME* are straight-forward. A user chooses a 'sign' icon and is asked for a password after clicking 'send'. After password input the message is signed without verification for integrity by the user. Retrieving the PIN works like in *eTRUST Mail* with modified character sequences to determine the correct window.

Siemens provides signature smart cards to their employees only. We used a card for field tests that cannot be obtained regularly. The communication between *TrustedMIME* and the card could not be monitored and altered with the *WINSCARD.DLL* we employed with *eTRUST.* The application refused to work with our modified library file.

3.7 GDtrust Mail

Manufacturer Giesecke & Devrient was not able to provide a signature smart card, even three months after applying for one. So we excluded smart card communication from our examination.

GDtrust Mail 4.0.2 provides a plug-in for Microsoft Outlook. To sign a message, the user selects a 'sign' button. After she clicks 'send', she is asked for the PIN to access her PSE. There is no way to detect if the message has been modified beforehand by Outlook, an Outlook plug-in, or by another process on the computer.

Of course, it was possible to retrieve the PIN like we did with Deutsche Post's *eTRUST.* We only had to modify the character sequences to look for.

3.8 Comparison

Almost all surveyed products use a PIN entry method that can easily be defeated by Trojan horse programs. Even *PKSCrypt's* 'secure input' provides no barrier for an attacker. The only difference is that it is less user-friendly than 'standard input'.

All surveyed products did not prevent modification of the input data before sending it to the signature smart card or processing it internally. *eTRUST* Mail displays the data in standardised format but does not protect this presentation. *Sign&Crypt* is better at this but still not perfect. *PKSCrypt* shows the file name but offers no way to verify that the file has not been modified. The other two products simply disregard this issue.

Product	Input modified	Display altered	PIN captured	Transfer to card inter-cepted
eTRUST Mail 1.01, 1.11	yes	yes	yes	yes
Sign&Crypt 2.10	yes	yes*	no	(untested)
PKSCrypt	yes	n/a	yes	secured

1.11				
TrustedMIME 2.2.5	yes	n/a	yes	(untested)
GDtrust Mail 4.0.2	yes	n/a	yes	n/a

4 Protection against the presented attacks

We show how the main vulnerabilities that most of the surveyed products revealed can be averted. The recommendations given here aim at removing these special vulnerabilities. However, our concept of a secure electronic paper that we present in the following section is to be preferred since it provides a more thorough approach to the problem.

4.1 Reliable input data

We propose to use a SWORM medium to ensure untampered input to the signature software (*Software-based Write Once Read Multiple*). First of all, the signatory determines the data that she wants to sign in the application software way ahead of the actual computation of the signature. The data is stored on the SWORM medium and cannot be modified after the file has been closed. It is not necessary to perform an additional presentation of the data in the signature software.

4.2 Secure PIN input

You can achieve a secure PIN input by using a piece of specialised hardware for that purpose. For instance, the keyboard manufacturer Cherry offers a smart card reader (G81-8015) that is integrated in a keyboard. After a special command for secure PIN input, key presses on the numerical keypad are transmitted directly to the smart card. The computer operating system receives asterisks instead, so no process, malicious or not, gets to know the secret PIN. During secure PIN input a red LED on the keyboard glows to raise user awareness of the secure PIN input mode. This LED cannot be manipulated by other commands.

It is possible to block a simple retrieval of the PIN – as *Sign&Crypt* shows. To circumvent the risk of keyboard loggers we nevertheless favour the PIN input directly at the card terminal.

4.3 Detecting tampered input to the smart card

The activation of a computation of a signature must ensure that the card will compute the signature for the intended data. Thus, providing the captured verification data for authentication (eg, PIN or biometrical data) has to be linked with the data to be signed.

There are basically two ways to achieve this. You can release the authentication information after the card has proved that it has received the intended data to be signed. On the other hand the signature software can sign the data to be signed with a session key and provide this key together with the verification data. The recipient can recover the session key only if the authentication information is correct.

Both protocols involve some secret knowledge of the signature application to communicate with the signature smart card. We propose to secure a PIN input in the signature application or to use transaction numbers that can be used only once.

5 Secure Electronic Paper

We have developed a set of tools to significantly increase the strength of the signing process against Trojan horse attacks. We call this 'Secure Electronic Paper' because our concept borrows methods that proved to be reliable for centuries to protect contracts signed on paper.

5.1 Requirements

Signatures on paper are considered secure by most people. We wondered how we could transfer to computer systems the properties of the paper signing process that protect it against malicious adversaries. The four main properties that make paper signing reliable are:

- determining the data to be signed before applying the signature
- fixing the data before applying the signature
- detecting modification of the data and of the signature after signing
- defining the semantics of the signed content.

What is different in the electronic world? We do not have paper on which we print or write, we have to use software to enter and modify our data, and then invisibly submit it to a smart card that applies a cryptographic function to the data. We can not sense what is going on and we have to trust the components of our computer system to tell us the truth about the digital data we work with. In our scenario, we can not trust the computer completely. We assume the operating system to be trustworthy, but some applications running on top of it may be Trojan horse programs.

We propose to extend the 'secure software' approach described earlier. While in its original form it puts all the security properties in a single signature application we think it is more promising to split this control and responsibility. Determining the data that is going to be signed takes place in the application software where the user works with it. Fixing is done between application software and signature software. This is the place where our Secure Electronic Paper enters the scene. Detection of modification after applying the signature is achieved by verifying the signature with the cryptographic methods already applied in the field. Determining the semantics of the signed data is supported by incorporating additional information used to deterministically display the data on the receiver's system. In the next paragraphs we will elaborate our concept in detail.

In short, Secure Electronic Paper combines safe storage with a reliable data transfer to the personal security environment, ie, the smart card containing the cryptographic algorithms and the signatory's private key.

5.2 Fixing the data before entering the signing process

In the same way that one usually does not sign a blank piece of paper and let someone fill in the text at a later date, without having control, we do this on a computer system. The document that will be signed is being worked on with some application software, eg, a word processor. This application software can not be assumed to be trustworthy, it may contain malicious code. If it is not a Trojan horse by itself, it could contain a Trojan horse as active document content. And still, it would be possible for a Trojan horse to manipulate the document between the application software and the signature software. A big help for an adversary is labelling a document as 'to be signed'. By this it is identified and becomes an interesting target for modification.

We withhold this information from a potential Trojan horse by not allowing a direct connection of application and signature software. It may seem convenient to just click 'sign' in your application software, but it reveals the purpose of the document you work with. Instead, every application software has a standard 'save work' function that stores a document for further processing without determining the exact purpose. A document could be opened by the same user with the same application or a different one, by another user on another machine, it could be stored on backup media. In order to not risk being detected a Trojan horse program must abstain from altering a document that is just saved. Hence we get an unmodified document that we now will transfer unmodified to the signature software.

Our method to ensure a reliable transfer to the signature software is not complicated. We use a WORM medium (Write Once Read Multiple). This medium does not allow modifications of a file after the file is closed. The file can only be read but not modified or deleted. A WORM can be implemented on a computer in many ways, either in hardware or in software. A CD-R drive, for instance, is also a WORM medium. For reasons of cost-efficiency we prefer a SWORM (=Software WORM), that does not require additional hardware. The access rights for the file are reduced to allow merely read-only access as soon as the export from the application software is completed.

Once we have stored the document to be signed in the SWORM as a trusted source we can use it to apply the signature to it. Some electronic signature laws require that the user must be provided with another possibility to display the data the signature is applied to immediately before the signature creation. This is perfectly possible with our solution. The presentation of the data can be done by opening the read-only file with the application software or by using a different software with the single purpose of displaying documents before signing.

The signature software opens the file stored on the SWORM and sends it to the signature smart card using a cryptographically-secured channel. The smart card is tamper-resistant and computes the signature for the data sent to it with the private signing key of the signatory. To prove that the signatory is present the card usually requires a PIN input from the user. A secure input can be achieved by using a cheap smart card reader with an attached dedicated keyboard.

5.3 Proof of tampering after signature creation

After the signature has been computed the signed document can not be altered without detection. This is ensured by strong cryptography used for the computation of the signature. Nevertheless we do not want to distribute documents with an incorrect signature on behalf of the user. A second situation we want to avoid is a Trojan horse modifying the document between the SWORM and the smart card.

Although we think that the second situation will not arise if the signature software process is implemented properly we propose an additional check. The computed signature is verified whether it matches the document the signature software sent to the card. If the signature is not correct we will have detected that a Trojan horse has manipulated communication between the SWORM and the card. We will then inform the signatory about the compromised signature.

5.4 Reliable presentation of signed data

We stated that the data that is signed has no semantics in itself. A signed paper document can be interpreted by the parties involved and by a third party, eg, a court. You have a fixed presentation that is not altered by the way you look at it. In computer systems we have to use application software to give the data a meaning. The same binary data will be interpreted more or less differently by different software products. While the same binary data has been signed by the signature creation device the interpretation of the signed document can be different at the site of the receiver without modifying the signed document itself.

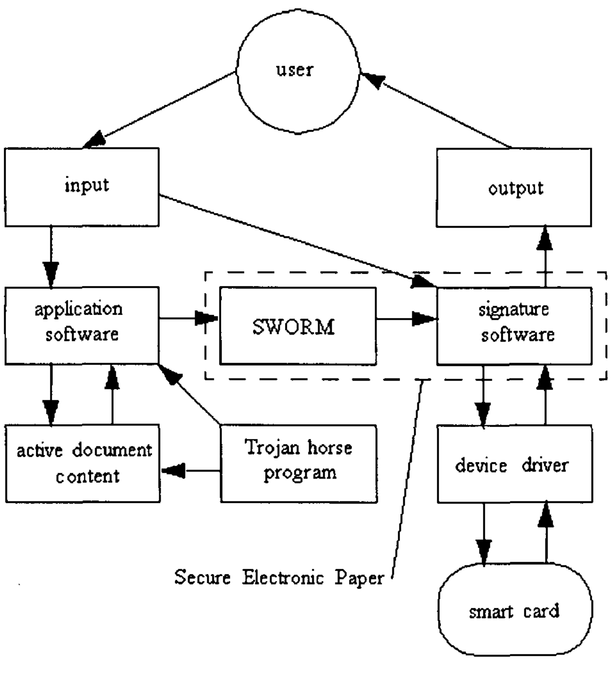
A Trojan horse that is transferred with the signed document as active document content, eg, a macro, can alter the presentation on the receiver's computer. So the receiver may have an otherwise Trojan-free system but will regardless get a presentation not intended by the signatory. The simplest solution would be to disable active document contents at all. But this may reduce flexibility way too much.

There are two ways we propose to tackle this problem. The first way is a softer approach than just disabling active content. It should be possible to restrict the actions of active content. So the receiver of a signed document should be able to determine which actions active content could perform on the document that would not alter the semantics. This requires cooperation of application software manufacturers regarding these options for their products. Today you can usually choose between allowing a macro to perform all actions or none.

The other more promising method utilizes that a computer is a deterministic machine. So it is in principle capable of presenting the signed document in the same way as it has been on the signatory's computer. This can be done without cooperation of the application software manufacturers. The idea is to build a 'sandbox' around the application that presents the signed document. This sandbox sets all environment parameters that can be determined by active document content inside the application software and be used against a deterministic presentation. The environment parameters are collected at the signatory's computer and include user name, computer name, network address, application software parameters etc. All parameters are included in an enhanced signature of the document so they can be evaluated by the sandbox on the receiver's computer. The same parameters will lead to the same presentation. This sandbox approach is especially suitable for the verification step.

6 Implementation of a 'Secure Electronic Paper'

In the previous section we explained which properties of signed paper documents can be used to secure electronic documents against unwanted modification by Trojan horse programs. We will now show how this can be implemented on common computer systems. We will focus on Microsoft Windows here but the results can be easily applied to Linux, Apple Macintosh etc.



Secure Electronic Paper (SEP) consists of a user policy, a SWORM medium, and a focused and robust signature software. The user policy ensures that data is submitted to the SEP without explicit knowledge of a Trojan horse about the transfer. The SWORM medium provides reliable input to the signature software and the signature software opens a secure channel to the signature smart card. Over this communication channel the data to be signed is transmitted to the card and the signature is sent back to the signature software and stored on the SWORM for further processing.

6.1 Reducing valuable information for corrupt participants

We raise the risk of detection for a Trojan horse by withholding information it needs for a successful attack. The document is transferred to the SWORM medium without determining the purpose of the transfer. A Trojan horse that modifies the document anyway when it is saved takes a high risk in being detected. By definition, a Trojan horse program has to keep its existence secret, so it will not risk being detected. Otherwise the origin of it could be traced and the author, possibly, be held liable in court. So even if a Trojan horse resides in the signatory's system it is not able to catch the right time to interfere with the signing process.

The reduction in information for an attacker is enforced by a user policy. A signatory is advised not to use seamlessly-integrated plug-ins in her application to trigger a signing process. Instead it is necessary to direct the output of the application to the SWORM and making it look like a standard and not suspicious action.

6.2 Early unchangeable input for signature creation

The proposed use of a SWORM is central to our concept since it is now possible and feasible to keep the input for the signature fixed at a very early stage. In contrast to other approaches we get the input at the earliest time possible before a Trojan horse even knows that the document will be signed and thereby become interesting.

To achieve a high level of security, the implementation of the SWORM should look like a non-SWORM medium. This prevents active document content from testing if the target of a standard 'save work' command is a SWORM medium.

A cheap implementation of the SWORM medium is a device driver that provides a virtual WORM medium on top of the NT file system. One folder is protected by access rights that allow the system account to freely access the folder but denies modification access for everyone else. In case the access rights are changed this is detected, the driver stops working and issues a warning to the user and the administrator.

The device driver allows the creation of files and read access to them but it does not allow to rename or modify the files after they have been closed. After a specified period of time the files are deleted. Since the files on the WORM are needed for secure signing only this is not a restriction but keeps the hard disk clean of unused files.

It is of course possible to use a hardware device with WORM functions. This could increase security because it is easier to attack a (software) device driver than a hardware device. For most cases we assume that a hardware WORM is too expensive.

6.3 Process communication with access controls

The communication between the components involved in the signing process is secured by the use of access rights. These rights can be configured in a way that make it impossible for a Trojan horse program to interfere with the process. The access control functions are assumed to be provided by the operating system. While this is true for Windows NT/2000 it does not hold for Windows 95/98/Me which are used on the majority of computers in private homes. In the case that the operating system does not provide access controls to protect the signing process we substitute these access rights by using cryptographic methods.

If access rights are not available we store the SWORM files in memory protected by the operating system. A file is immediately sent to the signature smart card for signing and then stored together with the accompanying signature. As long as the operating system is not shut down we achieve a SWORM security level comparable to the Windows NT implementation of the SWORM. In anticipation of a reboot however, the SWORM contents have to be stored on a hard disk where they are no longer protected. While tampering is now possible for a Trojan horse a modification can be detected by verifying the signature.

6.4 Exact labelling of data for later deterministic presentation

The proposed sandbox environment requires that we gather as much input parameters for a deterministic presentation as possible. Since we do not rely on cooperation with the application software manufacturers we have to pick up the parameters at various places.

We think that the following parameters have to be included with the signature to make a possible Trojan horse on the receiver's side believe it is executed on the signatory's machine. These parameters are the document format, the application used for working with the document, the version of the application, the parameters for the application (stored in the Windows system registry or in a configuration file), the size and colour depth of the Windows desktop, available fonts on the system, information about the origin and integrity of the fonts, the user name, the machine name, the network address, number and labels of storage media, serial numbers of the machine and application.

The parameters are collected by the signature software and added to the original data to be signed. This enhanced data to be signed is then signed by the signature smart card, thus protecting the integrity of the environment parameters.

The sandbox is built around the application used to present the signed document. Since not every program is capable of being run reliably in a sandbox it may prove necessary to disable active document content entirely on the system in those situations.

If the verification software on the target machine determines that it cannot display the signed data in the same way it has been displayed on the signatory's machine it issues a warning to the user and rejects the presentation. Usually the signatory and the receiver of the signed data will agree in advance on a common exchange format for the signed data that both are able to display. Including personal information might be viewed as a breach of confidentiality. However, as long as this information can be used by malicious document content, it has to be included to ensure a deterministic presentation.

7 Conclusions

Trojan horse programs, ie, programs with additional hidden, often malicious, functions, are more and more popular forms of attack. On the one hand, high-level macro programming languages in many office applications make it easy even for inexperienced attackers to write, hide and distribute a Trojan horse program. On the other, the emerging electronic signatures are likely to become a favourite target of attacks.

With our analysis of COTS signature software we focused on some attacks that every attacker with no inside knowledge of the software can perform. This does not rule out more sophisticated attacks by advanced attackers with malicious intent. In their current implementations the surveyed products offer almost no protection against attacks by Trojan horse programs. Legislation, especially in the European Union, tends to see the responsibility to prove that a signature has been faked on the signatory. A signatory using one of the products our paper deals with cannot be sure that electronic signatures will only be computed for the data she intends. The only way to employ the current versions of the programs responsibly would be to use them in a Trojan horse-free environment – which cannot be assumed rationally regarding today's personal computers.

We strongly recommend that the responsible software manufacturers act to incorporate protective measures into their products. The threat of Trojan horse programs is real and at the current stage of shipped products an attacker can just walk through open doors.

The proposed Secure Electronic Paper solves important problems regarding the reliable creation of electronic signatures in an insecure environment. It is feasible, cheap and user-friendly.

Firstly, the signatory determines the data that she wants to sign in the application software way ahead of the actual computation of the signature. It is not necessary to perform an additional presentation of the data in the signature software. Secondly, the transfer of the data through the signature software to the signature creation device is secured by access rights or cryptographic methods. And, lastly, the solution can be achieved without additional hardware, thus lowering the financial burden of security.

Active document contents still pose a special problem if a user is not willing or able to disable their execution. To ensure that a document is displayed identically on both the sender's and receiver's display, both parties must run the same document processing program with the same profile, ie, the same command-line parameters, options etc. Thus, in addition to a document and its signature, the sender must include the name, version and used profile of her document processing program, which the receiver must use to view the document. This is supported by using a sandbox environment in the signature verification step.

In conclusion, we have shown that the threat of Trojan horse programs attacking a document's integrity can be averted with only a few measures, which – compared with previous approaches – retain a system's flexibility and incur only minor inconveniences on its usability.

References

[1] Bontchev, V. (1996). 'Possible macro virus attacks and how to prevent them'. *Computers & Security* 15(1996):595-626.

[2] CERT Coordination Center (1999). *CERT Advisory CA-99-02-Trojan-Horses.* http://www.cert.org/advisories/CA-1999-02.html.

[3] Cremers, A.B., Spalka, A., and H. Langweg (2001). 'Vermeidung und Abwehr von Angriffen Trojanischer Pferd Programme auf Digitale Signaturen'. *Proceedings of 7. Deutscher IT-Sicherheitskongress.* Secumedia, Bonn. [German]

[4] Deutsche Post AG, ed. (2002). Signtrust Mail. Encryption and Signature of emails in accordance with the German Digital Signature Act. Manual for the integration into Microsoft Outlook.

[5] Deutsches Institut für Normung (2001). CEN/ISSS Workshop Agreement 14170. Security Requirements for Signature Creation Systems. Berlin.

[6] Docherty, P., and P. Simpson (1999). 'Macro Attacks: What Next After Melissa?'. Computers & Security 18(1999):391-395.

[7] European Parliament and European Council (1999). 'Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures'. Official Journal of the European Communities No. L 13(2000):2. http://europa.eu.int/ISPO/ecommerce/legal/documents/1999_93/1999_93_en.pdf.

[8] Ford, R. (1999). 'Malware: Troy Revisited'. Computers & Security 18(1999):105-108.

[9] Fox, D. (1998). 'Zu einem prinzipiellen Problem digitaler Signaturen'. DuD Datenschutz und Datensicherheit. 22.7 (1998). 386-388. [German]

[10] Gobioff, H., Smith, S., Tygar, J. und B. Yee (1996): 'Smart Cards in Hostile Environments'. USENIX Workshop on Electronic Commerce.

[11] Graf, Deutsche Telekom AG (2001). Personal communication.

[12] Hoffmeister, A., Cryptovision GmbH (2000). Personal communication.

[13] International standard ISO/IEC 7816-4. Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 4: Interindustry commands for interchange.

[14] Janácek, J., and R. Ostertág (2001). 'Problems in Practical Use of Electronic Signatures'. Proceedings of IFIP WG 9.6/11.7 Working Conference on Security and Control of IT in society-II. Ed. Fischer-Hübner, S., Olejar, D., Rannenberg, K., Bratislava.

[15] Kalkreuth, T., Deutsche Post AG (2000). Personal communication.

[16] Lacoste, G. , B. Pfitzmann, M. Steiner and M. Waidner, ed. (2000) SEMPER – Secure Electronic Marketplace for Europe. Springer, Berlin.

[17] Lapid, Y., Ahituv, N., and S. Neumann (1986). 'Approaches to Handling "Trojan Horse" Threats'. Computers & Security 5(1986):251-256.

[18] Mackert, A., Giesecke & Devrient GmbH (2000). Personal communication.

[19] Münchmeier, W., Cherry GmbH (2001). Personal communication.

[20] Okuntseff, N. (1997). Windows NT Security. Miller Freeman, Lawrence.

[21] Olfs, D., Siemens AG (2000). Personal communication.

[22] Popek, G.J., and C.S. Kline (1977). 'Encryption Protocols, Public Key Algorithms and Digital Signatures in Computer Networks'. R.A. DeMillo (1978). Foundations of Secure Computation:133-153.

[23] Pordesch, U. (1993). 'Risiken elektronischer Signaturverfahren'. Datenschutz und Datensicherheit 17.10(1993):561-569. [German]

[24] Pordesch, U. (2000). 'Der fehlende Nachweis der Präsentation signierter Daten'. DuD Datenschutz und Datensicherheit 24.2(2000):89-95. [German]

[25] Potzner, R., Utimaco Safeware AG (2001). Personal communication.

[26] Spalka, A., Cremers, A.B. and H. Langweg (2001). 'Protecting the Creation of Digital Signatures with Trusted Computing Platform Technology Against Attacks by Trojan Horse Programs'. Proceedings of IFIP TC11 16th International Conference on Information Security. Kluwer, Boston.

[27] Spalka, A., Cremers, A.B., und H. Langweg (2001). 'The Fairy Tale of "What You See Is What You Sign" – Trojan Horse Attacks on Software for Digital Signatures'. Proceedings of IFIP WG 9.6/11.7 Working Conference on Security and Control of IT in society-II. Ed. Fischer-Hübner, S., Olejar, D., Rannenberg, K., Bratislava.

[28] Stabell-Kulø, T. (2000). 'Smartcards: how to put them to use in a user-centric system'. *Proceedings of HUC2K The Second International Symposium on Handheld and Ubiquitous Computing.*

[29] Utimaco Safeware AG (2001). SafeGuard Sign & Crypt FAQ. http://www.utimaco.de/internet/uti_know.nsf/51356cb01fbb194b4125666b00482767/3a606701690efe01c12567ae002821ea?OpenDocument

Data protection for outsourced data mining

Boštjan Brumen, Izidor Golob, Tatjana Welzer and Ivan Rozman
University of Maribor, Faculty of Electrical Engineering and Computer Science
Smetanova 17, Si-2000 Maribor, Slovenia
{bostjan.brumen | izidor.golob | welzer | i.rozman}@uni-mb.si
AND
Marjan Družovec
University of Maribor, Faculty of Mechanical Engineering
Smetanova 17, Si-2000 Maribor, Slovenia
marjan.druzovec@uni-mb.si
AND
Hannu Jaakkola
Tampere University of Technology, Pori School of Technology and Economics
PO BOX 300, Fi-28101 Pori, Finland
hj@pori.tut.fi

Keywords: data protection, data mining, outsourcing

Received: January 18, 2002

In the paper, we present data mining from the data protection point of view. In many cases, the companies have a lack of expertise in data mining and are required to get help from outside. In this case the data leave the organization and need to be protected against misuse, both legally and technically. In the paper a formal framework for protecting the data that leave the organization's boundary is presented. The data and the data structure are modified so that data modeling process can still take place and the results can be obtained, but the data content itself is hard to reveal. Once the data mining results are returned, the inverse process discloses the meaning of the model to the data owners. The approach is especially useful for model-based data mining.

1 Introduction

The traditional means for collecting the data were restricted to the use of our natural senses – and so were the means for storing the data. Then, people started to use other, more persistent means for data storage, such as skins, stones and much later, papyrus and paper. But only since the advent of electricity (or more specifically, electronics) the collection of data is no more restricted to human natural senses only.

Electronic equipment today, operative in such diverse fields as geology, business, astronomy, or medicine is capable of gathering vast amounts of data. It is to notice that the storage nowadays is affordable: in 1980, an IBM PC had a 5 MB hard drive and was priced at $3000, and in 1995, for the same price, it was equipped with a 1GB hard drive [Bigus, 1999]. Today (in 2002), an IBM PC at the same price (not adjusted for inflation!) is shipped with an 80 GB hard drive. In the first 15 years, the amount of disk storage available in a PC compatible computer increased over 200 times. From 1995 to today, the increase is almost 80-fold and cumulative factor of increase since 1980 is more than 16000. The situation is equally scaled with larger computers.

In other words, the amount of data storage (and consequently the amount of data actually stored) doubles roughly every 15 months, beating the famous Moore's law[1] by 3 months. The last boost was clearly powered by the wide use of the Internet. In the early 1990s, the computers have definitely moved data in paper form to on-line databases.

For the last 30 years, the data were mainly a by-product of daily operations of a company. In general, not much was used for analytical purposes. But, the data are every company's vital assets [Reinhardt, 1995]. Assets are useless, unless they are used in (strategic) business processes. Data are facts and as such do not contribute to a company's competitive advantage. Thus, data are useless if they only reside in a company – even worse, they are causing costs: maintenance, storage, and security, to mention only a few. For this reason, there is a need that data be processed, analyzed, and converted into information. Upon information, company's decision-makers can act and sustain competitive advantage. Thus, once data are intelligently analyzed and presented, they

[1] Dr. Gordon E. Moore stated in 1965 that the number of transistors per square inch on integrated circuits (and thus the processing power) doubles roughly every 18 months [Moore, 1965].

become a valuable resource to be used for a competitive advantage [Hedberg, 1995]. Data indeed represent a great potential.

Many companies have recognized the value in data and started to systematically collect and store the data. When it came to using the data for other purposes than daily operations, the traditional transactional systems became to fail answering the questions in reasonable time, simply because they were not designed for such queries. In early 1990s a new paradigm was being introduced: the data warehouses.

Today, a modern, efficient and effective information system consists of "two hearts" – a database and a data warehouse (Figure 1).

The transactional databases take care of daily transactions, such as "receive payment" or "ship goods". The data warehouse part is responsible for answering ad-hoc queries, such as "show received payments for shipped goods by month by stores". In between runs the ETL (extract – transform – load) process, which updates the data warehouse content.

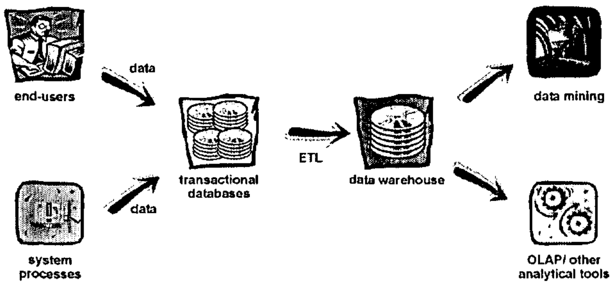


Figure 1: A Modern Information System

Data in the data warehouses, and especially in databases are considered as secure. However, there are many real-life cases where the data are not protected. Furthermore, even if they are protected in the "safe-house" environment of the databases and/or data warehouses, they sometimes leave the environment.

In the next section we describe the data mining process and explain why data mining poses a possible threat to data security.

2 Data Mining

The concept of data mining (DM) has been used since 1991 [Shapiro, 1991], and defined in 1996 [Fayyad, 1996] as a part of process known as knowledge discovery in databases (KDD) which itself is a "nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". Data mining constitutes only a subtask in the overall KDD process and refers to the information extraction itself.

Data mining is an area that connects many different fields, such as machine learning, artificial intelligence, statistics, databases / data warehouses, computer graphics, inductive logic programming, and many others. The results come in two basic forms – in descriptions and models. In the former, we try to describe, "what is going on in data" [Berry, 2000]. In the latter, we use currently available data and build models, which capture the rules or characteristics in data. The data mining results come in various forms, such as neural nets, decision trees and rules, association rules, clusters, connections (graphs, such as trees and networks) and various scores in form of numbers. There are countless techniques available, such as various methods of decision trees/rules induction, neural networks, genetic algorithms, clustering, statistical models, visualization, and many others. The list of possible results and of available techniques is far from being complete; the purpose is to inform the reader that data mining is not a single tool with a single result; it is rather a process.

As discussed in [Brumen, 2001], data mining is not reserved for large companies (with large databases). In the last few years it has become clear that smaller data sets can be mined too. The problem with smaller companies is that they do not posses in-house expertise for doing data mining, but they do have domain knowledge and understand their data structures much better. They have two choices: not doing data mining at all or doing it with help from outside. The former is sometimes not an option due to competitive reasons; the latter poses a potential security threat to data. The larger companies may require some help from outside as well.

Sometimes the company's resources (hardware and software) may not be adequate for mining purposes, thus the data need to leave the organization. Once outside the safe-house environment of organization's databases and data warehouses, they may be used for purposes other than specified. Due to requirements of most of today's mining tools, data need to be prepared in a flat file. The reason is that many mining tools are developed to avoid DBMS overhead, and to assure compatibility across DB and OS platforms. As such, data are much more easily copied and distributed. Even if we somehow enable access to our DB/DW from outside, we have again no control over what happens once the data are out.

3 Protecting data for outsourced data mining

One approach to protect our data is to use techniques developed for statistical databases. Two major systems, namely μ-Argus [Hundepool, 1996] and Datafly [Sweeney, 1997] have been developed. In these cases, the sensitive data values were generalized [Samarati, 2001] or not disclosed, respectively. In the data mining world, we can not have data that have been distorted or changed. For example, a decision rule on generalized and not disclosed data would read `IF marital_status = 'not released' ∧ sex='not released' ∧ AGE='young' ∧ zip_code='9xxxx' THEN class = 7`. Obviously, managers would find such form of discovered knowledge useless.

We propose an approach where no data semantics is lost, the statistics inside the data remains intact, but the data are still protected. In our framework, we transform the (relational) database that is to be exported to the outside world. The transformations are to be performed on both data structure and data values.

In the next paragraph we briefly present the notation and basic definitions for relational data model. We adopt the terminology for the relational data model (RDM) from [Elmasri, 1999] and for the abstract data type (ADT) from [Helman, 1994]. The ADT search table operators are not described in this paper; for formal definitions, which are beyond the scope of this paper, refer to [Helman, 1994].

Suppose we have a ***relational database DB*** [Elmasri, 1999], which is a finite set of ***relational schemas*** *DB* = $\{R_1,...,R_i,...,R_l\}$, where R_i is a relational schema, denoted by $R_i(A_1,...,A_n,...,A_N)$, where $A_1,...,A_n,...,A_N$ is a list of attributes. Each attribute A_n is the name of a role played by some domain D_n in the relation schema R_i. D_n is called the domain of A_n and is denoted by D_n = $dom(A_n)$. A relation r_i of the relation schema R_i (also referred to as a search table), denoted by $r_i(R_i)$, is a set of N-tuples, $r_i=\{t_1,...,t_m,...,t_M\}$. Each N-tuple t_m is an ordered list of N values, $t_m=\langle v_{m1},...,v_{mn},...,v_{mN}\rangle$, where each value v_n is an element of $dom(A_n)$, or is a special null value. The nth value of tuple t_m, which corresponds to the attribute A_n, is referred to as $v_{mn}=t_m[A_n]$.

A relation $r_i(R_i)$ is a mathematical relation of degree n on the domains dom(A_n), which is a subset of the Cartesian product of the domains that define R_i: $r_i(R_i) \subseteq (dom(A_1)\times...\times dom(A_N)\times...\times dom(A_N))$

In the following four definitions we set up a formal framework for reversible database transformation. In lemma 2 we claim that the process is indeed reversible and prove it in proof 2.

Definition 1: Let D_n and D_n^* be sets. A *function* from D_n into D_n^* is a subset F of $D_n \times D_n^*$ such that for each element $a \in D_n$ there is exactly one element $b \in D_n^*$ such that $(a,b) \in F$.

Definition 2: Let D^* be a set of domains, such that $D^* = \{D_n^* \mid |D_n^*| = |D_n|\}$. Let $D_n \to D_n^*$ be a function, and $F^D = \{f_n()\}$ be a set of transformations f_n. f_n is said to transform D_n onto D_n^*, if

$1. \forall b \exists a (f_n(a) = b)$

$2. f_n(a_1) = f_n(a_2) \Leftrightarrow a_1 = a_2$

Definition 3 (*a database schema*). Let DB^* be a set of new relations R_i^*, $DB^* = \{R_1^*,...,R_i^*,...,R_L^*\}$, where each R_i^* is denoted as $R_i^*(A_1^*,...,A_n^*,...,A_N^*)$. Each attribute A_n^* is a role played by some domain D_n^* in the relation schema R_i^*. The relation r_i^* of the relation schema R_i^* is a set of N-tuples, $r_i^* = \{t_1^*,...,t_m^*,...,t_N^*\}$. Each N-tuple t_m^* is an ordered list of N values, $t_m^* = \langle v_{m1}^*,...,v_{mn}^*,...,v_{mN}^*\rangle$, where each value v_n^* is an element of $dom(A_n^*)$.

The relation schemas of database *DB** are essentially the same as of those in the *DB*. That is, the number of the relation schemas is the same and each schema has the same number of attributes as the corresponding table in schema *DB*. Such a database is needed so that the instances are easily transformed from database *DB* into database *DB**. The transformation function is defined in the next definition.

Definition 4: (*table transformation*). Let us define a function *Trans* that transforms a table instance (a relation), r_i, into a transformed table instance r_i^*, such that:

$$Trans(r_i) = r_i^* = \left\{ \begin{matrix} t_1^*,...,t_m^*,...,t_M^* \mid t_m^* = \langle v_{m1}^*,...,v_{mn}^*,...,v_{mN}^*\rangle, \\ v_{mn}^* = f_n(t_m[A_n]), t_m[A_n] \in t_m, t_m \in r_i \end{matrix} \right\}$$

Lemma 1: Function *Trans* is bijection.

Proof 1: Function *Trans* is bijective if it is injective and surjective.

A function *Trans* is said to be injective, if and only if whenever *Trans(p)=Trans(q)* $\Rightarrow$ *p=q*. Suppose we have two tuples $p,q \in r_i$, $p = \langle p_1,...,p_N\rangle$, $q = \langle q_1,...,q_N\rangle$. Since *Trans({p})=Trans({q})* $\Rightarrow$ $\{\langle p_1^*,...,p_n^*,...,p_N^*\rangle \mid p_n^* = f_n(p[A_n])\} =$ $\{\langle q_1^*,...,q_n^*,...,q_N^*\rangle \mid q_n^* = f_n(q[A_n])\} \Rightarrow$ $p_n^* = q_n^* for\ 1 \le n \le N$. Since by Definition 2 $f_n(a_i) = f_n(b_i)$ when $a_i = b_i$ for $\forall i \Rightarrow$ $p_n = q_n \Rightarrow \langle p_1,...,p_N\rangle = \langle q_1,...,q_N\rangle \Rightarrow p=q$.

A function *Trans* is said to be surjective, if and only if for any $q^* \in r_i^*$, there exists an element $p \in r_i$ such that *Trans({p})={q*}*.

Suppose we have $q^* \in r_i^*$, $q^* = \langle q_1^*,\ldots,q_n^*,\ldots,q_N^* \rangle$. Each q_n^* is calculated as $q_n^* = f_n(p[A_n])$. From definition 2 we have that for function f_n, for each b exists an a such that $f_n(a) = b$. By declaring $s_n^* = b$ it is evident that there must exist a *p*, so that *Trans({p})* = *{q*}* $\Rightarrow \forall q^* : \exists p(Trans(\{p\}) = \{q^*\})$.▪

Lemma 2: Function *Trans* has an inverse, $Trans^{-1}$, such that $Trans^{-1}(r_i^*) = r_i$, where $Trans(r_i) = r_i^*$.

Proof 2: Let *Trans* be bijection. Then exists one and only one bijection $Trans^{-1}$ that implies

1. $Trans^{-1}(Trans(\{p\})) = \{p\}$ for $\forall p \in r_i$
2. $Trans(Trans^{-1}(\{q\})) = \{q\}$ for $\forall q \in r_i^*$

That unique bijection $Trans^{-1}$ is called the inverse function of *Trans*.

We first prove that $Trans^{-1}(Trans(\{p\})) = \{p\}$ *for* $\forall p \in r$.

Since *Trans* is injection: $Trans(\{p_1\}) = Trans(\{p_2\}) \Rightarrow \{p_1\} = \{p_2\}$;

Since $Trans^{-1}$ is a function:

$Trans(\{p_1\}) = Trans(\{p_2\}) \Rightarrow$

$Trans^{-1}(Trans(\{p_1\})) = Trans^{-1}(Trans(\{p_2\}))$

i.e. $\{p_1\}=\{p_2\}$.

Next we prove that $Trans(Trans^{-1}(\{q\}))$ = $\{q\}$ *for* $\forall q \in r_i^*$.

Since *Trans* is surjection:

for any $q \in r_i^*$ there exists $p \in r_i$ such that $Trans(\{p\}) = \{q\}$ and since $Trans^{-1}$ is a function: $\{p\} = Trans^{-1}(\{q\})$ is defined for every $q \in r_i^*$.

Thus, $Trans(\{p\}) = Trans(Trans^{-1}(\{q\})) = \{q\}$. ▪

Instead of giving out the original database *DB*, we give out the transformed database *DB**. The set of transformations on domain *D*, F^D, the old names of relations, *R*, and the old names of attributes for each relation are kept secret.

This way the attack on database *DB** is much more difficult since the attacker has no semantic information on the content of the database. If F^D is carefully chosen the attack becomes infeasible. The functions that can be chosen are strong encryption algorithms or other functions that preserve statistical properties of data [Adam, 1989], [Willenborg, 1996]. The advantage is that the F^D can easily be chosen so that it corresponds to the value of data to be protected.

For clarity let us take a closer look at an example where we have three tables. We transform them using the above definitions.

Example 1:

Suppose we have a set of domains *D*:

```
D={Integer, String, Char},
```

a set of relational schemas *DB={R_i}*:

```
DB={STUDENT, COURSE, GRADE},
```

where each relation schema is denoted as

```
STUDENT(S_ID, Fname, Lname, Zip, City, Age);
COURSE(C_ID, Name, Credits);
GRADE(STU_ID, COU_ID, Grade_Value);
```

and each attribute has the following domain:

```
STUDENT:
Dom(S_ID)={1, 2, 3}, Dom(Fname)=String={John,
Martha, Alice}, Dom(Lname)={Smith, Jones},
Dom(Zip)={12345, 12346, 12347},
Dom(City)={London, Helsinki, Berlin},
Dom(Age)={18, 19, 20};

COURSE:
Dom(C_ID)={1, 2, 3}, Dom(Name)={Math, Physics,
Biology}, Dom(Credits)={5, 10};

GRADE:
Dom(STU_ID)={1, 2, 3}, Dom(COU_ID)={1, 2, 3},
Dom(Grade_value)={A, B, C}.
```

We have a set of relations (instances) of relation schemas, presented in a tabular form:

student(STUDENT):

S_ID	*Fname*	*Lname*	*Zip*	*City*	*Age*
1	John	Smith	12345	London	18
2	Martha	Smith	12346	Helsinki	19
3	Alice	Jones	12347	Berlin	20

course(COURSE):

C_ID	*Name*	*Credits*
1	Math	5
2	Physics	5
3	Biology	10

grade(GRADE):

STU_ID	*COU_ID*	*Grade_Value*
1	1	A
1	2	B
2	1	A
2	3	B
3	1	C

We define *D** as

```
D*={Number1, String, Number2}.
```

Further, we define

```
DB*={TABLE1, TABLE2, TABLE3},
```

where each relation schema is denoted as

```
TABLE1(COL1, COL2, COL3, COL4, COL5, COL6);
TABLE2(COL1, COL2, COL3);
TABLE3(COL1, COL2, COL3);
```

and each attribute has the following domain:

```
TABLE1:
Dom(COL1)={10, 13, 16}, Dom(COL2)={Str1, Str2,
Str3}, Dom(COL3)={Str4, Str5}, Dom(COL4)={37042,
37045, 37048}, Dom(COL5)={Str6, Str7, Str8},
Dom(COL6)={61, 64, 67};

TABLE2:
Dom(COL1)={10, 13, 16}, Dom(COL2)={Str9, Str10,
Str11}, Dom(COL3)={22, 37};

TABLE3:
Dom(COL1)={10, 13, 16}, Dom(COL2)={10, 13, 16},
Dom(COL3)={8, 9, 10}.
```

By applying the function *Trans* on each of the relation instances, `student(STUDENT)`, `course(COURSE)` and `grade(GRADE)`, we get the following transformed instances:

Trans(`student(STUDENT)`*)*=`table1(TABLE1)`
Trans(`course(COURSE)`*)*=`table2(TABLE2)`
Trans(`grade(GRADE)`*)*=`table3(TABLE3)`

The instances are a set of tuples. We present each of the instances in a tabular form:

`table1(TABLE1):`

Col1	*Col2*	*Col3*	*Col4*	*Col5*	*Col6*
10	Str1	Str4	37042	Str6	61
13	Str2	Str4	37045	Str7	64
16	Str3	Str5	37048	Str8	67

`table2(TABLE2):`

Col1	*Col2*	*Col3*
10	Str9	22
13	Str10	22
16	Str11	37

`table3(TABLE3):`

Col1	*Col2*	*Col3*
10	10	10
10	13	9
13	10	10
13	16	9
16	10	8

▪

Suppose an outside entity does models-based data mining on the above tables. In models-based data mining, the goal is to make a model based on the underlying data. The models can be neural networks, decision trees, decision lists, association rules, and a set of clusters, to name only a few. Basically, the models can be built using supervised or unsupervised learning. In the former, the algorithm takes as input a vector of values and returns the class as the output. The calculated class is in learning phase compared to the actual value and the correction is made if needed, thus supervised learning. In the latter, the algorithm tries to group vectors together based on some similarity function. Since there is no correct answer, there is nothing to supervise.

For example, when building a decision tree, the algorithm makes a branch based on some statistical properties of data, not on actual values or attribute names. For these reasons the actual data and their structure can be hidden, and the results will still be the same, as long as statistics within data is maintained.

Any mining result that is returned is again in form of a model that includes the transformed data elements. Thus, the result that is returned can be taken as a set of relation schemas $DB^*=\{R_i^*\}$ with a corresponding set of relation instances $\{r_i^*\}$, and connections among them, depending on the structure of the model. Note that even a single data cell can be viewed as an instance of a table with one attribute (column) and one tuple (row).

With inverse transformation, the data owner decodes the model (relation instances) into a readable form. The example 2 depicts the process.

<u>*Example 2:*</u>
Suppose the data mining rule says that `IF table1.Col6 < 67 THEN table3.Col3 > 8.`

We have two table instances, q^* (i.e. `table1`) and s^* (i.e. `table3`). Each table instance has only one tuple, i.e. $q^*=\{t_1^*\}$and $s^*=\{u_1^*\}$

`table1:` q^*

Col6
67

`table3:` s^*

Col3
8

First, by using the inverse, $Trans^{-1}$ on q^*, we get:

$Trans^{-1}(q^*)=q=\{t_l\}=$
$=\{t_l | t_l\,\texttt{[Age]}=f_1^{-1}(t_1^*\,\texttt{[Col6]})\}=$
$=\{t_l | t_l\,\texttt{[Age]}=(t_1^*\,\texttt{[Col6]-7)/3}\}=$
$=\{t_l | t_l\,\texttt{[Age]=(67-7)/3}\}=$
$=\{t_l | t_l\,\texttt{[Age]=20}\}$

Next, by applying $Trans^{-1}$ on s^*, we get:

$Trans^{-1}(s^*)=s=\{u_l\}=$
$=\{u_l | u_l\texttt{[Grade\_Value]}=f_3^{-1}(s_1^*\,\texttt{[Col3]})\}=$
$=\{u_l | u_l\texttt{[Grade\_Value]}=f_3^{-1}\texttt{(8)}\}=$
$=\{u_l | u_l\texttt{[Grade\_Value]= C}\}$

Since instances q^* (`table1`) and s^* (`table3`) correspond to instances `student` and `grade`, respectively, we get the following two relation instances as a result:

`student:` q

Age
20

`grade:` s

Grade_Value
C

Finally, the rule decodes to `IF Student.Age < 20 THEN Grade.Grade_Value > C.` •

4 Conclusion

The information age has caused a shift from a product-centric society to a data-centric one. While the fundamentals for data storage (and protection) have been around for almost 40 years, and have become very solid since the introduction of relational database systems, the new technologies, such as data warehousing and data mining, require special attention and care.

In the paper we presented a new knowledge discovery technology – data mining – from the data security perspective. In data mining process, sometimes the data need to be modeled outside of an organization. In this case they need to be protected in such a way that the modeling process is still possible and the data security is preserved at the same time.

We presented a formal framework for transformation of a schema and content of a relational database. We prove that the transformation is reversible. With careful selection of transformation functions, the attack on data becomes infeasible. The functions can be selected so that the effort to break them exceeds the value of protected data. By using the framework the outsourced data miner is still able to do the data mining, and the data remain secure. The reversibility of the process enables the data owners to decode the model into a readable form.

The presented framework is especially useful for models-based data mining, where the data values and data structure play no role when building a model.

References

[Adam, 1989] Adam, Nabil R.; Wortman, John C.: Security-Control Methods for Statistical Databases: A Comparatice Study, ACM Computing Surveys, Vol. 21, No. 4, pp. 515-556, 1989

[Berry, 2000] Berry, Michael A. J.; Linoff, Gordon: Mastering Data Mining: The Art and Science of Customer Relationship Management, John Wiley & Sons, Inc., New York, USA, 2000

[Bigus, 1999] Bigus, Joseph P: Data Mining with Neural Networks: Solving Business Problems from Application Development to Decision Support, McGraw-Hill, New York, USA, 1999

[Brumen, 2001] Brumen, Boštjan; Welzer, Tatjana; Golob, Izidor; Jaakkola, Hannu: Convergence Detection in Classification Task of Knowledge Discovery Process, In Proceedings of Portland international conference on management of engineering and technology, Portland, Oregon, USA, 2001

[Elmasri, 1999] Elmasri, Ramez A.; Navathe, Shamkant B.: Fundamentals of Database Systems, 3rd Edition, Addison-Wesley Publishing, New York, 1999

[Fayyad, 1996] Fayyad, Usama; Shapiro-Piatetsky, Gregory; Smyth, Padhraic; Uthurusamy, Ramasamy; (Eds.): Advances in Knowledge Discovery and Data Mining, AAAI Press, USA, 1996

[Hedberg, 1995] Hedberg, Sara R.: The Data Gold Rush, Byte Magazine, 10-1995

[Helman, 1994] Helman, Paul: The Science of Database Management. Irwin Inc, 1994

[Hundepool, 1996] Hundepool, Anco J.; Willenborg, Leon C.R.J.: μ- and τ-Argus: Software for Statistical Disclosure Control, Proceedings of 3rd International Seminar on Statistical Confidentiality, Bled 1996

[Moore, 1965] Moore, Gordon E.: Cramming More Components Onto Integrated Circuits, Electronics, Vol. 38, No. 8, April 19, 1965

[Reinhardt, 1995] Reinhardt, Andy: Your Next Mainframe, Byte Magazine, 5-1995

[Samarati, 2001] Samarati, Pierangela: Protecting Respondents' Intentities in Microdata Release, IEEE Trans. On Knowledge and Data Engineering, Vol. 13, No. 6, 2001

[Shapiro, 1991] Piatetsky-Shapiro, Gregory; Frawley, William J. (Eds.): Knowledge Discovery in Databases, AAAI/MIT Press, USA, 1991

[Sweeney, 1997] Sweeney, Latanya: Guaranteeing Anonymity when Sharing Medical Data, the Datafly System, Proceedings, Journal of American Medical Informatics Association, Washington, DC, Hanley & Belfus Inc., 1997

[Willenborg, 1996] Willenborg, Leon C.R.J.; De Waal, Ton: Statistical Disclosure Control in Practice, LNS Vol. 111, Springer-Verlag, 1996

An active networks security architecture

Arso Savanović, Dušan Gabrijelčič, and Borka Jerman Blažič
Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
(arso|dusan|borka)@e5.ijs.si
AND Stamatis Karnouskos
Frauenhofer FOKUS, Kaiserin-Augusta-Alee 31, 10589 Berlin, Germany
karnouskos@fokus.fhg.de

Keywords: active networks, security architecture, FAIN

Received: March 4, 2002

Active networks allow user-controlled network programmability. A security framework has to assure that our infrastructure will behave as expected and will efficiently deal with malicious attacks, unathorized attempts to execute active code etc. We present here a security architecture that is designed within the FAIN project and aims at supporting multiple heterogeneous execution environments. We argue for the pros and cons as well as why we have selected the specific components and also take a look at their interworking in order to provide the security services to the execution environments our active network node hosts.

A Introduction

Basic AN principles have serious consequences have serious consequences for the operation of an Active Network (AN). The possibility of loading and executing active code in Active Network Nodes (ANNs) imposes considerable threats to expected operation of AN/ANN due to flaws in active code, malicious attacks by unauthorized users, and conflicted code execution. Thus, security in AN deals mainly with protecting system (AN infrastructure) from malicious (unauthorized) and erroneous use. The central objective of FAIN [7] security architecture is to guarantee robust/secure operation of AN infrastructure despite the unintentional or intentional misbehaving of users, i.e. their respective active code/active packets.

Fulfilling these objectives is fundamental for the usability of ANs. Clearly, if it is trivial for any user to intentionally degrade the performance of an AN or any single ANN, or to bring down the AN/ANN, then ANs are not really usable. Note the difference between degrading performance of (disabling) an ANN and that of an AN. It is possible that specific network service, i.e. the respective active code, is consistent with local security policy of an ANN; however, due to global, network wide behaviour of the protocol, it can degrade the performance of (part of) network or even completely disable it. Furthermore, if an unintentional error in the design of a new network service, its implementation (active code), or its configuration can degrade performance of an AN/ANN or disable an AN/ANN, then ANs are not really usable.

Finally, if a malicious or unintentional misbehaving of any user can severely degrade or even disable the network services perceived by other user(s) of an AN but with no affect on the ANN/AN, then again, ANs are not really usable. The threat model for active networks covers three broad classes of security issues: protecting AN infrastructure from users and active code, protecting users and active code from other active code, and protecting users and active code from AN infrastructure. However, the scope of initial FAIN security architecture is limited mainly to the first class, i.e. protecting AN infrastructure from users and active code.

B FAIN Active Nodes

The FAIN Reference Architecture consists mainly of AA, VE, EE and Node OS:

- **Active Applications/Services** (AA) are applications executed in Active Nodes. An AA is often referred to also as Active Code (AC).
- **Execution Environments** (EE) are environments where application code is executed. A privileged EE manages and controls the Active node and it provides the environment where network policies are executed. Multiple and different types of EE are envisaged in FAIN. EEs are classified into Virtual Environments (VEs), where services can be found and interact with each other. VEs are interconnected to form a truly virtual network.
- **NodeOS** is an operating system for active node and includes facilities for setting up and management of channels for inter-EE and AA-EE communications, manages the router resources and provides APIs for AA/EEs, isolates EEs from each other. Through its extensions the NodeOS offers:

- **Resource Control Facilities (RCF).** Through resource control resource partitioning is provided. VEs are guaranteed that consumption stays within the agreed contract during an admission control phase static or dynamic.
- **Security Facilities.** Main part about security is authentication and authorisation of using the resources and other objects of the node like interfaces and directories. Based on the policy profile of each VE security is enforced.
- **Application/Service code deployment facilities.** As flexibility is one of the requirements for programmable networks partly realised as service deployment either on the fly or static, the NodeOS must support it.
- **Demultiplexing facilities.** It filters, classify and divert active packets. Flows of packets arrive at the node and they should be delivered to the VE and consequently to the service inside the VE they are destined for.

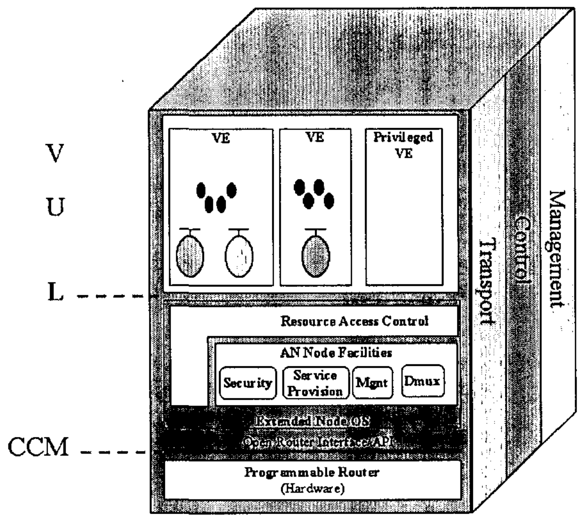


Figure 1: FAIN active node reference architecture

Figure 1 describes the main design features of the FAIN nodes. In FAIN a number of node prototypes are under development as follows:

- A high performance active node, with a target of 150 Mb/s.
- A range of flexible and very functional active nodes/servers, with the objective of supporting multiple VEs and hosting heterogeneous EEs.

The common part of the prototypes (the FAIN middleware) is the NodeOS with the relevant extensions.

C General FAIN Model and Security Requirements

The fundamental property of FAIN AN is the possibility to dynamically inject active code known as active application (AA), that implements new functionality, into the network. This code is executed or interpreted by specific execution environment within ANNs and this way it provides end- user applications with application specific network services. Many different active applications can coexist in an active network. We assume that FAIN AN consists of unlimited number of network nodes and some of them are active (ANN). Active code is injected into the network via active packets, which carry active code itself or its reference, which is used by ANNs to install the code from code repository. Code can be executed in the nodes within the packet path. Execution provides new functionality in the network, which can be temporary or permanent. It can also produce new packets. Each execution uses some of the ANN and AN resources, like CPU, storage and bandwidth, again temporarily or permanently. Specific code in an ANN can be injected, removed or replaced by explicit or implicit request. Additionally, the following properties apply to generalized AN model [1]:

- an AN is a distributed system
- an AN is a packet-switched network, as opposed to circuit-switched
- not all nodes in an AN need to be active
- an AN explicitly provides for computation inside the network, but
- the primary goal of active networks is communication, not computation
- the contents of an active packet can legally change inside ANNs[1]
- not all packets are active
- an AN consists of multiple domains, each controlled by a different administration.

Active networking supplies the users with the ability to install and execute program code within a network node. That by its nature is a security critical activity. In such an infrastructure the security implications are far more complex than in current static environments. In AN the author of the active code, the user who deploys it, the owner of the node hardware, the owner of the execution platform can all be different entities governed by different security policies. In such a heterogeneous environment security becomes an extremely sensitive issue. The possibility of loading and executing active code in ANNs imposes considerable threats to expected operation of AN/ANN due to flaws

[1] In ANNs the payload (data part) can be changed also, not just header fields.

in active code, malicious attacks by unauthorized users, conflicted code execution etc. Thus, security in AN deals mainly with protecting system (AN infrastructure) from malicious (unauthorized) and erroneous use. The central objective of FAIN security architecture is to guarantee robust/secure operation of AN infrastructure despite the unintentional or intentional misbehaving of users, i.e. their respective active code/active packets.

Active Code (AC) is transferred to the node or is itself mobile e.g. in the form of a mobile agent. Therefore the attacks that AC and also the EE are susceptible to are more than those in current passive networks.

In general we can have:

- Misuse of an active network node by the active code
- Misuse of active code by other active code.
- Misuse of active code by an active network node.
- Misuse of active code and/or execution environment by the underlying network infrastructure.
- Misuse of the Active Network as an entity.

Finally a combination of the above categories is possible. These kinds of attacks (the complex and collaborative ones) are very difficult to detect, let alone to prevent or effectively tackle. Classical examples include the co-operation of various hosts and ACs against another EE or AC. Threats can also be analysed from the perspective of a single ANN, and from the network- wide perspective. Of course, threats to a single ANN apply also to the whole AN (domain). However, network-wide threats can be more subtle and harder to combat, since they are based on the global, distributed nature of network protocols, and thus, their respective active codes.

In the initial phase of the FAIN project, only high priority security requirements have been addressed in detail:

- authentication
- authorization
- policy enforcement
- active code/packet integrity
- code verification
- audit.

We have compiled this list in light of the main objective of the FAIN security architecture, which is to provide secure and robust operation of FAIN AN infrastructure in spite of unintentional and malicious misbehaving of AN users, i.e. their respective codes. From this perspective, our criteria in assigning priorities can be summarized as follows:

- How subtle is particular security requirement, i.e. the respective threat "behind" the requirement?
- More subtle yields lower priority.

C.1 Authentication, Authorization, and Policy Enforcement

FAIN ANN is essentially a multi-user computing system. As in any such system, enforcement of access control is a requirement of high significance within every FAIN ANN. On the other hand, FAIN aims at developing a flexible system. In order to achieve the desired level of granularity we decompose access control in authentication, authorization and policy enforcement. These three security requirements have the highest-priority within FAIN security architecture.

C.2 Active code/packet integrity

Active code is executed within an ANN and performs actions on behalf of a user. Therefore, active code is the "carrier of activity" and as such, it is a powerful tool when misused by malicious users, which can potentially tamper with active code while it is in transit over the network. For instance, the whole access control system could be circumvented, if the original active code can be modified or swapped with any other code. Similarly, there are ways to obviate access control system by tampering with active packets, such as cut and paste attacks and replay attacks. This is why protecting integrity of active code and packets deserves a high- priority.

C.3 Code verification

Protecting the active code integrity is a first step to ensure non- modification of the transient code. However this is considered pretty basic and we need to go beyond that in order to achieve a high level of security. The active code has to be somehow marked and tightly coupled with one or more entities, based on which further security decisions can be made. The code carries credentials from these entities, which have to be verified in order to set the security context within which this active code can execute. As code verification is critical into taking further security decisions, this is considered a high-priority requirement for the FAIN security architecture.

C.4 Audit

The Audit Manager component is an integrated part of the security architecture. Via this component

- all events occurring from the usage of the security subsystem are implicitly logged for further future usage.
- It also provides an interface to explicitly log any other events coming from other parts of the FAIN architecture in a clear and homogeneous way.

Modern computer systems do not emphasize enough on the significance of the audit facilities. However audit tools help in realizing possible security leaks (or even preventing some) and make sure that mistakes are not repeated. We feel that within the AN community special care has to be taken with audit activities and therefore it is also considered a high-priority security requirement.

D Technical aspects of active network security

D.1 Authentication

Authentication is a process of verifying an identity claimed by or for a system entity. Symmetric or asymmetric cryptography can be used for authentication. Symmetric cryptography is suitable only for closed systems due to its scalability problems. Thus, we use asymmetric cryptography in FAIN. This requires every AN user to have a public/private key pair and a valid public key certificate. Nevertheless, common remote authentication protocols employ a handshake, i.e. a two way communication in order to perform authentication. In active networks this would require an end-host to perform an authenticating handshake protocol with every ANN en route, which is clearly unacceptable. Thus, we propose the use of "unidirectional" procedure, where authentication is based on digital signatures and one-way communication from end-host to an ANN. Overview of this authentication scenario:

- User employs its private key to digitally sign the static part of an active packet and adds a signature to the packet it transmits
- ANN uses the public key certificate to verify the validity of the user's public key.
- If valid, ANN employs user's public key to verify the digital signature of the packet

A PKI infrastructure is needed to support authentication based on digital signatures.

D.2 Authorization

There will be several enforcement engines in FAIN ANN, each of them residing in a different FAIN ANN subsystem and responsible for mediating access to functions and resources of the respective subsystem. On the other hand, authorization component can be either integrated with policy enforcement or separated from it. In the former case, there would also have to be one authorization engine per ANN subsystem. In the latter case, only one, general-purpose authorization engine can be implemented and used by all policy enforcement engines.

We have adopted the latter approach for FAIN due to the following reasons:

- no duplication of work; this is especially important if we consider that design and implementation of any security component is a difficult and subtle task
- inherent flexibility as a consequence of separation of authorization from enforcement
- possibility of reuse of existing tools.

D.3 Policy enforcement

In the initial phase our discussion is limited to enforcement mechanisms up to and including FAIN Node facilities level, i.e. we currently omit the discussion of policy enforcement within EEs/VEs. Policy enforcement is the active component of security architecture that enforces authorization decisions and thus enforces the use of ANN resources, which is consistent with local security policies. We distinguish two types of resources, hardware and functional resources. Hardware resources include basic low-level ANN resources such as memory, storage capacity, CPU cycles and link bandwidth. Functional resources are high-level resources in the sense that they consume some portion of hardware resources. However, with functional resources it is not important how much memory or storage space they consume but rather what purpose they serve within an ANN, i.e. what function they provide. Examples of functional resources include:

- special purpose files, such as configuration files,
- policy entries in the policy database,
- ANN state,
- ANN API functions themselves, etc.

We note that all resources in an ANN, hardware and functional, are accessible at certain node interface. In order to prevent unauthorized use of ANN resources, policy enforcement has to be scattered across different ANN subsystems that provide specific subsets of ANN API functions. Thus, basic technical approach to policy enforcement is to add an "adaptation" software layer on top of every subsystem API, which then mediates access to node API functions. Whenever an ANN function is called by an "external" entity (such as VE, EE, active code), this software layer:

- intercepts the request (call to node function) and suspends it
- provides call parameters to authorization engine effectively asking for authorization decision; parameters include caller ID, called function name, object(s) name, amount of requested hardware resources, etc.
- when authorization decision is returned
  - if request is authorized, enforcement layer resumes the execution of the request
  - if request is not authorized, enforcement layer discards the request and thus prevents unauthorized actions from taking place

In addition to this "high-level" operation, policy enforcement also has to operate at low-level in order to enforce proper usage of low-level hardware resources. At the "lower level" enforcement is embodied in a more complex policing algorithm(s), which can control the scheduler(s) for specific resource and thus impose limits on resource usage by an entity.

D.4 Active packet/code integrity

In general, protecting integrity of active packet/code while in transit over network involves cryptographic operations. The most common approach is as follows:

- at the sending end—generate integrity protection token (data):
  - calculate a hash of the packet/code
  - encrypt the hash to protect it from modifications
  - send the encrypted hash together with the active packet/code
- at the receiving end—verify the integrity of the packet/code:
  - decrypt the hash that accompanies the received active packet/code
  - calculate a hash of the active packet/code,
  - compare the two hashes; if they differ active packet/code has been modified and should not be processed or allowed execution.

The hash value, which is carried along with active packet/code and is used for integrity, can be protected either by applying asymmetric encryption or symmetric encryption.

If asymmetric encryption is used, integrity protection is provided by digital signatures and there is no need for ANNs to maintain a private/public key pair.[2] ANNs only need to be able to obtain the certificate chain, which verifies the validity of the public key of the party signing the active packet/code. Thus, the advantage of asymmetric encryption is that it eases management of encryption and decryption keys. However, the downside is that asymmetric encryption is on the order of two magnitudes slower than symmetric encryption.

In case symmetric encryption is used, the encrypted hash is known as a MAC[3] value. However, this requires each ANN to maintain a non-compromised private/public key pair and a public key certificate. ANN uses asymmetric encryption to establish a shared secret key with the sending end. Thus, asymmetric encryption in this case is still used, but this time only to set- up a secret key for symmetric encryption. Additional downside of symmetric encryption is that integrity protection requires a negotiation phase before active packet/code can be injected into the AN.

[2]Note that other security requirements may/will impose this.

[3]Message Authentication Code.

In FAIN we have used a combination of asymmetric and symmetric encryption for active packet/code integrity, in order to leverage the advantages of both. The proposed approach is as follows:

- each ANN has a public/private key pair and a public key certificate
- each ANN maintains a shared secret key with every of its direct neighbouring ANNs; neighbouring ANNs employ asymmetric cryptography for establishing and updating shared keys
- the sending end signs active packet/code (using asymmetric encryption) and injects it into the AN
- the ingress ANN fetches the public key of the signer and verifies it against its certificate
- the ingress ANN then uses this key to check integrity of the received active code
- if active code is intact, ingress ANN calculates a MAC value, using a secret key it shares with the next hop ANN
- ingress ANN sends MAC value along with active packet/code and its signature
- every subsequent ANN
  - uses the secret key it shares with previous-hop ANN and checks integrity
  - calculates new MAC values using the secret key it shares with the next hop ANN
  - sends the new MAC value along with the active packet/code

This approach represents a trade-off between FAIN goals of security and performance. On one hand, the described approach is based on the assumption that trust exists between ANNs, which obviously reduces the level of security. However, this is a valid assumption at least in a single domain, which is under the control of a single authority. The trust within domain is applied by per-hop symmetric encryption. On the other hand, this approach is advantageous for ANN performance, since it leverages high speed of symmetric encryption algorithms. Furthermore, because (pre-established) per-hop shared keys are used, it effectively eliminates the symmetric key negotiation phase. Note that per ANN public/private keys and per-hop cryptographic calculations are used. However, since some parts of an active packet are dynamic, i.e. they can change at every hop, they cannot be protected with digital signatures and, thus, per hop integrity will have to be used, anyway.

D.5 Code verification

Verification can enable us to trust to some extent that the active code will behave safely and properly and that we can have some guarantees on its resource usage on the node and in the network. But we shall say in general that verification provides only enhanced trust in proper and safe code execution, which is usually not related to the trust in the user on behalf of which the code is executing. Code verification can help an ANN decide whether to run the newly received code. If the code fails the verification test, it is not trusted and it is dropped or alternatively it can run in an EE with minimal facilities available. In the latter case the EE is the same one that will be used to run anonymous active code. Broadly, code verification techniques can be classified into two groups:

1. Digitally signed code, so we trust the user, organization or repository that has signed the code. Digital signature can be checked at the NodeOS level, immediately after it is available.

2. Various other mechanisms that can enhance the trust in proper and safe execution. These mechanisms mainly operate within EEs, and include techniques like proof carrying code, JAVA bytecode verification, and restricted languages.

If there is resource consumption estimate available, simple resource check is also possible. Since the scope of initial FAIN security architecture is limited to the NodeOS level, we propose the use of first approach, which employs digital signatures for code certification.

D.6 Audit

The information gathered by the audit manager are stored into the audit database and via a policy controlled way are available for further use. Decomposition of auditing activity in this way allows the active node base code to be simpler as it does not have to implement complex handling of audit messages. Audit logs should be securely stored not only locally on the node but also in a distributed scheme as this offers better survivability to attacks against the node. Apart from the node audit, the active code may perform its own auditing and possibly report it via an interface to the node's audit facilities.

E FAIN Security Architecture

Figure 2 depicts a FAIN active network node; all shaded components are part of security architecture. As depicted, FAIN security architecture roughly comprises three parts: security subsystem, other ANN security components, and external security support facilities. Note that the scope of initial FAIN security architecture does not include EE layer of FAIN ANN architecture.

E.1 Security subsystem

Most of security critical decisions are made by security subsystem, which is one of several subsystems within an ANN. The Security subsystem is also responsible for management of security critical data, such as encryption keys, credentials, and policies.

This subsystem is the core of FAIN security architecture and includes the following components:

1. **Crypto Engine:** performs the actual cryptographic operations, such as symmetric encryption/decryption, asymmetric encryption/decryption, and hashing. It implements various cryptographic algorithms, which are used by other components in the security subsystem.

2. **Security Environment (SE):** in a secure fashion stores various encryption keys, which are required by crypto engine. For example, SE stores ANN's public key pair (private and public key) and all secret keys that an ANN shares with its neighbours (one per neighbour).

3. **SE Manager:** is used for managing the keys in SE. SE manager can provide facilities for manual configuration of encryption keys and can also automatically manage keys, e.g. by triggering a key exchange protocol with neighbouring ANN.

4. **Integrity Engine:** checks the integrity of active packets and active code. It depends on integrity protection data contained within an active packet and on crypto engine to do the necessary cryptographic operations.

5. **Verification Engine:** performs code verification (at NodeOS level), if any. It may depend on special data contained within an active packet and on crypto engine to do the necessary cryptographic operations.

6. **Authentication Engine:** verifies the authenticity of active packets. It depends on authentication data contained within an active packet and on crypto engine to do the necessary cryptographic operations.

7. **Authorization Engine:** is responsible for making a decision whether a given user request to execute specific action or to access/manipulate particular object within an ANN is authorized or not. Authorization engine provides this "service" to all policy enforcement engines in an ANN.

8. **Policy database:** stores security policies, which govern who can do what in an ANN.

9. **Policy Manager:** when asked by the authorization engine, searches policy DB and returns all security policies, that are relevant for a particular request, which is currently subject to authorization. It also provides facilities for editing entries in policy DB, either manually by an authorized user, or automatically, i.e. download policies from a centralized policy server.

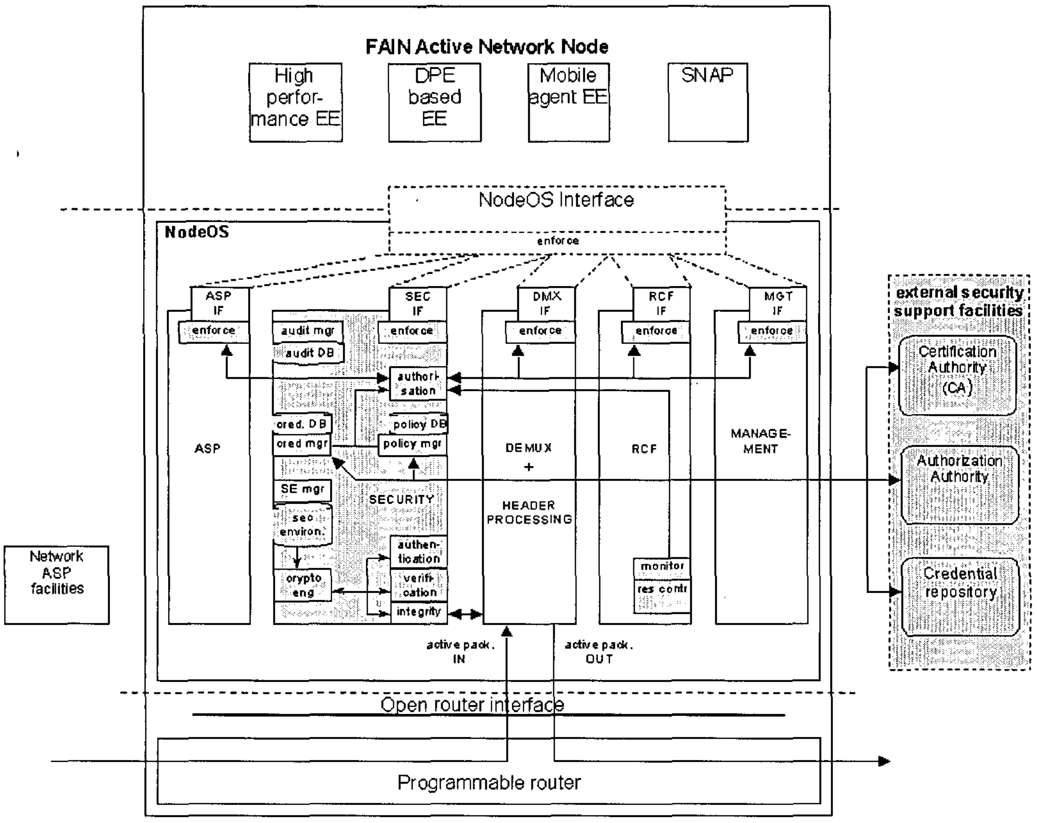


Figure 2: FAIN security architecture.

10. **Credential database:** stores users' credentials, such as public key certificates and attribute certificates.

11. **Credential Manager:** when asked by authorization engine, searches credential DB and returns all credentials, that are relevant for a particular request, which is currently subject to authorization. It also provides facilities for editing credential database, either manually by an authorized user, or automatically, i.e. search and download credentials from an external credential repository.

12. **Audit database:** stores an audit log of security critical events.

13. **Audit Manager:** will be the place where all security architecture's components audit their function in order to be used later in resolution of problems or even to make decisions. E.g. an Intrusion Detection System would use a view of the audit DB in order to recognize attacks against the system. The audit could be also distributed for survivability reasons.

E.2 Other ANN Security Components

The second part of security architecture includes components that are part of ANN but are external to security subsystem. This includes policy enforcement engines and various components providing environment variables, e.g. resource usage monitor. Various subsystems within an ANN offer their services and objects for use by users via their interfaces. Access to these objects and services is governed by security policies. Thus, enforcement of node security policies has to be performed at the point where they can be violated, i.e. at interfaces. At every ANN subsystem interface, a policy enforcement engine acts as an adaptation layer, which is responsible for mediating access to subsystem services and objects based on the authorization decision. While authorization is only a decision making, enforcement is an active process that prevents access to services and objects by unauthorized users. The Enforcement engine suspends the request at interface, asks the authorization engine whether this request is allowed and acts upon authorization decision, i.e. either allows or denies execution of the request.

In addition to these "high-level" enforcement engines, there are also "low- level" enforcement engines, which are tightly coupled with specific hardware resources available within an ANN and therefore they are considered as part of Resource Control Framework (RCF). Finally, there are some components in an ANN, which provide authorization engine with necessary data to make authorization decision. For example, resource usage monitor provides data on current hardware resource consumption by particular user, and a clock provides current time and date.

E.3 External Security Support Facilities

In the initial security architecture, we envisage these security support facilities:

- Certification Authority (CA)
- Authorization Authority
- Credential repository

Authentication based on digital signatures requires a user to have a public key pair and a valid public key certificate. Public key certificate binds a public key and an identity of its owner; these certificates are issued by a trusted third party called Certification Authority (CA). When a user enters an ANN, he must present his public key certificate to authentication engine. Alternatively, he can provide a pointer to his public key certificate in the form of a reference to certificate repository.

In the initial phase of FAIN, a single CA is sufficient for demonstration and testing purposes. This can be later extended with more CAs forming a fully-fledged Public Key Infrastructure (PKI).

Similarly, a scalable approach to authorization requires a user to have one or more attribute certificates. Attribute certificates bind public keys directly to privileges, which can be exercised by the owner of the key. Attribute Certificates are issued by a trusted party called Authorization Authority, which may not necessarily be the same as the Certification Authority. When a user enters an ANN, he must present one or more attribute certificates either directly or by reference to a repository. Later, when a user tries to execute an action, attribute certificates are used by the authorization engine to decide whether he has the necessary privileges.

The Credential repository can store both, public key certificates and attribute certificates. Repository can be implemented in many ways, such as a directory service or a web repository.

F Operation of security architecture

Basically, there are two checkpoints where security functionality from figure 2 is employed to protect an ANN: when a user enters an ANN and when a user tries to execute some action within an ANN. The former is represented by an arrival of an active packet in ANN and we call it *entry-level protection.* The latter occurs when a request for certain operation arrives at NodeOS interface and we call it *execution-level protection.*

In addition to these two types of security protections, one can distinguish two operations, which do not directly provide any security protections. Rather, these two are a sort of "backplane" operations, which support entry- and execution-level security protections. These support operations are: setup of a shared secret key between neighbouring ANNs and obtaining the missing credentials from a node external repository. A secret key, which is shared by a pair of neighbour ANNs, is used for hop-by-hop symmetric encryption of portions of active packet, which is leveraged e.g. for integrity protection. To setup a shared secret key between two ANNs, any key exchange protocol can be used. Key exchange has to be performed when a new ANN is added to/removed from the AN and whenever the key lifetime expires.

On some occasions, a situation may arise, when the credentials needed to make authorization decision are not present in an ANN. In this case, the missing credentials have to be searched for and obtained from somewhere in the network, usually from a repository service.

Finally, there is an audit facility within FAIN ANN, which is responsible for keeping a log of all security critical events within an ANN. This information is required for activities such as intrusion detection and analysis and assessment of security breaches.

F.1 Entry-level security protections

Figure 3 depicts a sequence of security operations that are performed for every packet that arrives at ANN. These security checks are aimed at detecting anything suspicious about this particular packet and, if so, discarding it. A packet is only delivered to appropriate EE if it passes all checks. Upon entering an ANN, an active packet is first processed in order to extract information needed for security checks. This information includes:

- Digital signatures, which are used for authentication, integrity, and verification
- MAC values, which are used for integrity protection
- Public key certificate(s), which are used for checking digital signatures
- Attribute certificates, which are used for authorization

After this information has been provided to security subsystem, entry-level security checks are triggered. The security subsystem verifies credentials, checks integrity of active packet and active code, performs code verification (if any), and performs authentication and returns the result of these operations to the de-multiplexing subsystem. Only if all these checks are successful, the packet is allowed to "enter" an ANN, i.e. it is first processed at NodeOS level (e.g. IP processing) and then forwarded to appropriate EE for further processing. If any of security checks fails, this is reported to de-multiplexing system, which discards the packet.

F.1.1 Active packet integrity

Integrity protection is based on cryptography. Every packet carries along at least one special token, which is used for integrity protection. This token is in the form of a digital signature and/or a MAC (Message Authentication Code). In

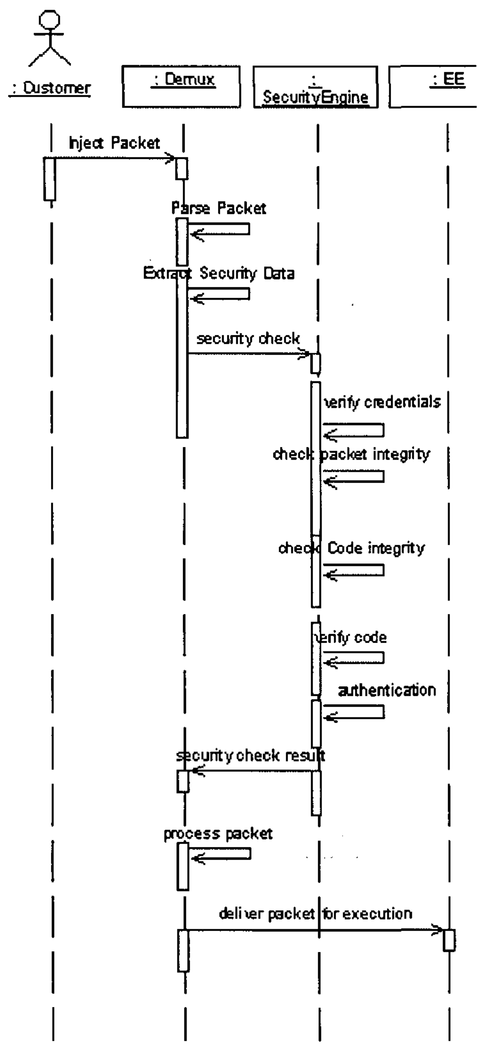


Figure 3: Entry-level Security Checks

FAIN bothe techniques are used for packet integrity. This is due to the fact that digital signatures are required for authentication, so it is sensible to leverage digital signatures for integrity as well. However, this applies only to static parts of an active packet, which do not change en route and can be signed by the source of the packet. For the dynamic parts of an active packet, which can change within an ANN, per hop integrity protections based on MAC must be used.

Integrity engine checks integrity in three steps. Firstly, it asks crypto engine, which performs all cryptographic calculations, to decrypt the integrity token, in this case a MAC value; crypto engine needs to get appropriate decryption key from ANN's security environment. This decryption process returns a hash of the packet as it was seen by its sender. Secondly, it asks crypto engine to calculate hash of the packet. The last step is to compare this hash against the decrypted token. If they are equal, then integrity of the packet can be assumed. If these two values differ, however, then integrity check has failed.

F.1.2 Active code integrity and code verification

Here we check the integrity of the newly received active code. Note that integrity checks for active packet and active code need to be separate because of the fact that these protections are in most cases provided by different encryption keys, i.e. different actors. The reason for this is that active code can be tampered with even before it is included in any active packet. Thus, digital signature generated by packet source at packet creation does not suffice for active code. Every active code is accompanied by at least one special token, which is used for integrity protection. This token has a form of a digital signature and/or a MAC (Message Authentication Code). It is envisaged that both approaches to integrity will be used in FAIN. Digital signatures by code provider/manufacturer can provide integrity protection until code is injected into the active network. From there per-hop MAC protection can be used, which is expected to yield performance gains. Additionally, the advantage of per-hop MAC protection is that it covers both packet and code at the same time. Note that we omit the discussion of multi-domain issues in the initial phase.

When active code integrity is provided with digital signature generated by code provider, the integrity engine must first process code provider's public key certificate in order to extract and validate providers public key. After extracting a valid public key integrity engine checks integrity in three steps, similar to active packets. First, it asks crypto engine to decrypt the integrity token, in this case a digital signature. This decryption process returns a hash of the code as it was seen by the code provider. Second, it asks crypto engine to calculate hash of the code. The last step is to compare this hash against the decrypted token. If they are equal, then integrity of the code can be assumed. If these two values differ, however, then integrity check has failed.

The majority of active code verification techniques are specific to particular EE. Since we have limited our current scope to NodeOS only, we do not address these mechanisms. The only general verification mechanism, which can be placed in the NodeOS is based on digital signatures by trusted parties. For the initial FAIN security architecture we use code providers as these trusted parties. This effectively eliminates the need for distinct code verification process within the NodeOS, since code provider's digital signature is checked as part of code integrity check.

F.2 Execution-level Security Protections

Once an active packet has successfully passed entry-level checks, active code(s) can execute and perform operations within an ANN on behalf of some user. Obviously, some users will have more privileges than others, i.e. security policies define who can do what in an ANN. In order to protect an ANN it is necessary to prevent users from abusing their privileges and violating security policies.

According to the previous paragraph, execution-level protection includes two steps:

- Evaluating every execution request against node security policies, which is performed by authentication engine and
- Allowing or denying execution based on positive or negative authorization decision, respectively; policy enforcement engines are responsible for this.

F.2.1 Policy enforcement

Crucial to policy enforcement is the subsystem specific enforcement engine, which is implemented as an adaptation layer mediating requests at subsystem interface. Every request at subsystem interface is intercepted and suspended by this adaptation layer. Before execution a request has to be evaluated against local security policies. Enforcement engine does not itself evaluate whether the request is compliant with local security policies. Instead it invokes the authorization procedure within the security subsystem and feeds it with request information, such as: requested action, name of target object and requesting caller ID. Only after authorization returns, "request authorized" does an enforcement engine allow execution of the request by the underlying subsystem. Obviously, if authorization returns a negative answer, i.e. "request not authorized", then enforcement engine simply discards the suspended request. This way, it prevents execution of unauthorized requests and essentially enforces users to adhere to local security policies.

F.2.2 Authorization

Here we check everything that is required to authorize the request, i.e. to decide whether to grant it or not. In flexible access control systems, authorization is not integrated with enforcement. Instead it is separated logically and in implementation. In this way, a single authorization engine can be used by multiple policy enforcement engines.

Authorization decision is based on the following set of data:

- request information (action, object name, caller ID)
- local security policies, which govern the way in which particular object can be used
- credentials associated with particular caller ID
- current values of environment variables, such as time of day and amount of resources used by subject

Enforcement engine provides the request information when it asks for authorization decision. This information is used as an "index" by policy and credential managers for fetching appropriate policies and credentials, respectively. Environment variables are provided to authorization engine upon request by facilities, such as system clock and resource monitoring module within Resource Control Framework (RCF) subsystem. Finally, after gathering all the required information, authorization engine processes this data according to its internal rules, which return a simple result, either saying, "request authorized" or " request not authorized." This is returned to the calling policy enforcement engine, which then acts accordingly.

G Related work

FAIN aims to develop a heterogeneous ANN, allowing coexistence of various technologies that enable installation and execution of active code within an ANN. Consequently, FAIN security architecture is aimed at providing a more general solution which provides necessary protections for such an heterogeneous system. This is reflected by the fact that security architecture we have presented does not incorporate details of specific EEs that exist in the FAIN ANN. Its goal is to be as EE independent as possible and provide a common set of basic security services required by all AN enabling technologies.

Some research projects on active networks have already tried to tackle the issue of security, in various ways and at different levels of completeness [3,4,6,8]. In contrast to FAIN, all these security architectures are tied to the specifics of the respective model of active networks and, consequently, reflect the original design decisions and, more importantly, trade-offs chosen by developers of the respective model. Java Security Architecture [5] proved to be useful for AN security, but it has some drawbacks in this context [6]. There has also been some more general work on AN security [2], but this work is still in the early phase.

Some research projects on active networks have already tried to tackle the issue of security [3][4][5][6]. Contrary to FAIN, all these approaches are tied to specifics of particular model of programmability. When designing a more general

AN security architecture, which is the case in FAIN, these specifics can not be assumed. Java Security Architecture [7] proved to be useful for AN security, but again it is technology specific and it also has some drawbacks [5]. There has also been some more general work on AN security [8]. This work is still in the early phase.

H Conclusion

We have presented in this paper a security architecture for future IP active networks as it is done in the context of FAIN project. We try to tackle the high priority security requirements such as authentication, authorization, policy enforcement, active code and active packet integrity and verification and last but not least audit. We have analysed the main design decisions that we have taken and the reasons why we decided to follow them. Subsequently we have presented the components of a security architecture that will be used by multiple heterogeneous execution environments within the same active node. We also provide a look in the interworkings of the architecture and its decision-making logic. A prototype implementation of the presented active network security architecture is currently under development, which will be used for exploring the advantages and drawbacks of our approach.

Acknowledgement

This paper describes work undertaken and in progress in the context of the FAIN - IST 10561, a 3 year project during 2000-2002. The IST program is partially funded by the Commission of the European Union. The FAIN consortium consists of University College London—UK, Jozef Stefan Institute—Slovenia, NTUA—Greece, Universitat Politecnica de Catalunya—Spain, Deutsche Telekom Berkom—Germany, France Telecom/CNET—France, KPN—The Netherlands, Hitachi Europe Ltd.—UK, Hitachi, Ltd.—Japan, Siemens AG—Germany, ETH—Switzerland, Fraunhofer FOKUS—Germany, IKV++ GmbH—Germany, INTERGAsys—Spain, University of Pennsylvania—USA.

References

[1] Active Network Working Group. Architectural Framework for Active Networks, Jul 1999.

[2] Active Networks Security Working Group. Security Architecture for Active Nets, May 2001.

[3] D. Scott Alexander, Wiliam A. Arbough, Angelos D. Keromytis, and Jonathan M. Smith. A Secure Active Network Environment Architecture: Realisation in SwitchWare. *IEEE Network*, Special Issue: Active and Programmable Networks:37–45, May/Jun 1998.

[4] B. Branden, B. Lindell, and S. Bernson. A Proposed ABone Network Security Architecture. ABone-draft, Nov 1999.

[5] Li Gong. Java security architecture (JDK1.2). Technical report, Sun Microsystems, Oct 1998.

[6] Active Networks Working Group. SANTS Security Overview, May 2000.

[7] Future Active IP Networks (FAIN) Project. http://www.ist-fain.org.

[8] Stephen Schwab, Richard Yee, and Rishi Dandekar. AMP Security Overview. Technical report, NAI Labs, May 2000.

Software engineering: configuration languages

Ayaz Isazadeh
Department of Computer Science, Tabriz University, Tabriz, Iran
Phone: +98 411 334 4015, Fax: +98 411 334 2102
E-mail: isazadeh@tabrizu.ac.ir

Keywords: Configurable distributed systems, Configuration languages, Software architecture, Software configuration, Software integration

Received: October 12, 2000

Distributed software systems are playing increasingly important roles in the world of software engineering. Software systems are becoming larger and large-scale systems are naturally distributed. In the spirit of software "re-use" and software "evolution", "configurable distributed systems" are becoming the center of attention. A number of researchers have developed "configuration management systems" and "configuration languages" for describing, constructing, and managing configurable distributed systems. I believe that a pause is in order, at this point, to analyze and review the current state of the research in this area. This paper presents a set of features and requirements expected from a configuration language and reviews some of the existing configuration languages. Specifically, the configuration languages of Conic and its successors, Polylith, LOTOS as a configuration language, Raven, MetaH, Durra, and Argus are reviewed. The paper establishes a basis for the review by proposing a list of requirements, reviews the languages on the basis of these requirements, and concludes with a summary and some final remarks.

A Introduction

Distributed software systems are playing increasingly important roles in the world of software engineering. Software systems are becoming larger and large-scale systems are naturally distributed. In the spirit of software "re-use" and software "evolution", "configurable distributed systems" are becoming the center of attention. Distributed software systems are usually large and often complex to describe, construct, manage, understand, and maintain. Current research approaches attempt to reduce this complexity by separating the description of software structures from the programming of individual components. The components are programmed in a programming language like C, C++, or Ada. Special languages, however, called *configuration languages*, are used for describing the structure of a distributed system. A number of researchers have developed "configuration management systems" and "configuration languages" for describing, constructing, and managing configurable distributed systems. I believe that a pause is in order, at this point, to analyze and review the current state of the research in this area.

This paper presents a set of features and requirements expected from a configuration language and reviews some of the existing configuration languages. Specifically, the configuration languages of Conic and its successors, Polylith, LOTOS as a configuration language, Raven, MetaH, Durra, and Argus are reviewed. The paper establishes a basis for the review by proposing a list of requirements, reviews the languages on the basis of these requirements, and concludes with a summary and some final remarks.

The criterion used for selecting a language for the review is the capability of the language to satisfy a major part of the requirements presented in Section B.2.

A.1 Focus and Approach

The focus of this paper is on the analysis and comparison of the configuration languages selected for the review. Considering that different configuration languages use different terms to express the same or similar concepts, the first step is to present some definitions, introducing a set of basic terms that are used throughout the paper. The second step is to establish a clear basis for the review by providing a set of requirements or features expected from a configuration language. The configuration languages, then, can be reviewed based on these requirements.

The remainder of the paper defines the basic terms, presents the set of requirements expected from a configuration language, reviews the languages on the basis of these requirements, and finally provides a summary and some concluding remarks. But, first, a note on the examples:

A.2 A Note on the Examples

My first thought was to provide a more empirical basis for the analysis and comparison of the languages reviewed. One possibility, in this regard, is to use a single common example to make direct comparison of the languages easier. There are, however, three problems with doing this:

[0]This work has been supported by the Research Institute for Fundamental Sciences, Tabriz, Iran.

1. A single example would show points of strength in some languages and weaknesses in others. Some other example would demonstrate just the opposite. A single example will not be fair to all languages. For example, Darwin supports dynamic reconfiguration and Polylith does not, while Polylith supports post-factum configuration but Darwin does not.

2. An example large enough to illustrate all the points of comparison offered in the paper, and expressed in all eight of the languages considered, would make the paper too long.

3. There are multiple implementations for any example in each of the languages. In the interest of fairness, the example would need to be developed iteratively, ideally with the assistance of the developers of each language. Such a project might offer some revealing insights, and the present paper would be a good starting point, but it is likely that all of the still active research groups will have evolved their ideas in the meantime, so that a comparison at this level needs greater stability in this area than presently exists.

Therefore, I dismissed this possibility and, instead, I included for each language an example from the creators of the language. The examples are intended to show just the look and feel of each language; no further description is given. Interested readers are encouraged to see the corresponding references.

B Basis of Review

This section provides the core of the definitions of basic terms and concepts, followed by my perception of the requirements that configuration languages are collectively and currently attempting to meet. The core definitions given are elaborated in further detail in the following sections in the context of the languages being reviewed.

B.1 Basic Concepts and Definitions

A distributed system is composed of some interconnected components. The base components, also called *computational components*, perform computations and provide the core functional behavior of the system. Each component is active, in the sense of having a separate thread of control.

Computation is the first aspect of a distributed system. The second aspect is *communication* among the components. The third aspect of a distributed system is *configuration*, the way in which the system is structured. *Configuration languages* are formal languages designed to specify (or program) system configurations. A system configuration is a description of which components are connected to each other, and so are permitted to communicate with each other. The communications aspect specifies the nature of this communication in greater detail. The term *computational programming* refers to programming a computational component using programming languages like C++ or Ada. The term *configuration programming* refers to describing the configuration of a distributed system using a configuration language.

Some MILs, (*Module Interconnection Languages*), can also be used for configuration programming, e.g., Polylith MIL. Generally, however, the primitives in an MIL, modules and resources, are different from the primitives in a configuration language, computational components. Computational components are active software components, like UNIX processes or like Ada tasks. An MIL module, on the other hand, as defined by DeRemer and Kron (DeRemer & Kron 1976), is a segment of code that defines one or more resources, where a resource is any nameable entity in the language for programming-in-the-small. Others have slightly different definition of these primitives as described in Dean's survey of MILs (Dean 1993) In general, an MIL primitive does not necessarily form an active component.

In a configuration language supporting hierarchical composition, a *composite component*, also called a *configuration component*, is constructed (or configured) from computational components using the configuration language. Composite components can be recursively configured into higher level composite components, forming a hierarchical structure.

The *interaction structure* of a distributed system is the way in which communication between the system components is provided. Configuration languages normally specify the interaction structure of a system by describing the connections between the components. A connection between two components is *direct* if either component uses knowledge of the inner workings of the other component; otherwise, the connection is *indirect*.

The term *interaction safety*, within the context of a configuration language, refers to the ability of the language to prevent "unwanted" interactions between components. This is discussed more fully in B.2.3 below.

The term *dynamic reconfiguration* refers to changes in a system's configuration while it is running. Dynamic reconfiguration includes adding, removing, or replacing a computational component, changing the connections between components, and moving a component from one machine to another one. Dynamic reconfiguration is either ad-hoc performed interactively, or pre-planned, triggered by the system and performed automatically.

The term *post-factum configuration* refers to a configuration where at least one of the computational components is an existing software system developed independently with no prior plan for using it as a computational component in a distributed system. Configuration languages usually impose certain requirements on the computational components, e.g., using a particular programming language or a particular set of communication primitives. Post-factum configuration, therefore, may require wrappers or packagers, adapting the interfaces of existing software components to meet the requirements of a configuration language.

B.2 Requirements

This section presents a list of requirements or features expected from a configuration language of the class being reviewed. A configuration language that satisfies all of these requirements can reduce the complexity of constructing, understanding, and maintaining dynamically reconfigurable and reliable distributed systems. There are many issues that the requirements as presented do not address. This means that they can be significantly improved, albeit at the expense of increased complexity, and only after further experience has been gained. One of the purposes of this paper is to suggest future directions for work in this area.

1. *Separate computation, communication, and configuration.* The language should separate the three aspects of a distributed system: computation, communication, and configuration. This separation makes it possible to deal with each aspect of the system with minimal effect on the other two aspects. Computational components, for example, can be programmed independently, simplifying both computation and configuration programming; or different communication mechanisms can be used with minimal effect on the system configuration or computational components.

   Configuration independent software development, where the only configuration information embedded in the software components is some standard way of allowing connections to be made to the components, is a well-known and long practiced technique. A good example is the use of pipes and filters in Unix. Most configuration languages separate configuration from computation and communication in this sense. Separation of communication and computation, on the other hand, seems to be more difficult, in the sense that these two aspects of distributed systems are more or less combined in most of the configuration languages reviewed.

2. *Separation of computation and configuration.* The component interaction structure provided by the language should guarantee decoupling of the components, in the sense that a component should not be allowed to directly interact with any other component. Direct connections tightly couple the components, combine the three aspects of distributed systems and, consequently, complicate configuration management, especially dynamic reconfiguration. Configuration languages should use indirect connections.

3. *Component interaction Safety.* The language should provide some measure of interaction safety, reducing the possibility of "unwanted" interactions between components. The major role of a configuration is to provide interaction between components. Configuration languages should provide for complete specification of both the syntax and the semantics of the permitted inter-component interactions, and should support the creation of actual links among components satisfying these specifications. The configuration language implementations should guarantee to provide only the specified interactions - no more, no less. Any other interaction is *unwanted* and must be prevented.

4. *Compositionality.* The language should support compositionality and de-compositionality: it should allow hierarchical composition of computational and composite components. A configuration language satisfying this requirement can express configuration of a complex distributed system in a hierarchical structure, making it easier to understand maintain, and reconfigure.

5. *Dynamic Reconfiguration.* The language should support dynamic reconfiguration. For many large distributed systems, it is simply not acceptable to shut down the entire system for a modification or reconfiguration. These systems must be modified, reconfigured, and extended while they are running without imposing significant performance degradation. It is, therefore, an important feature for a configuration language to be able to express dynamic configuration changes to a running distributed system. Applications of dynamic reconfiguration include adding a new functionality or changing/removing an existing one, employing a new technology, implementing fault tolerance and service availability methodologies, and using load sharing/balancing techniques.

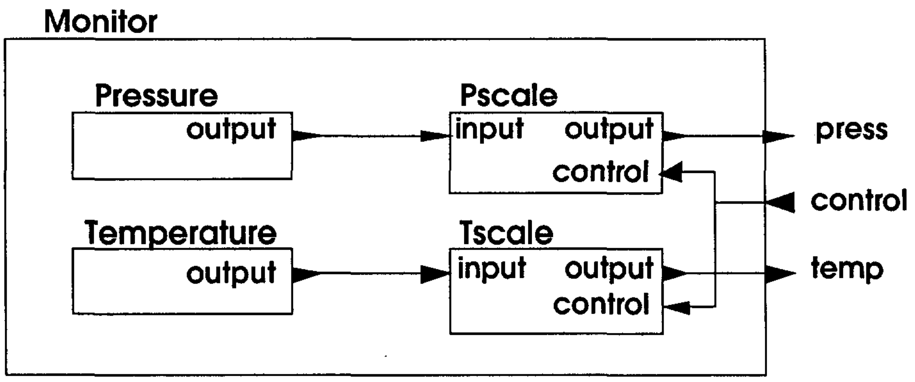
6. The configuration language should make no restriction on the languages used for computational programming. This requirement makes it possible to configure a distributed system using computational components written in diverse programming languages. It is also a prerequisite for the next requirement.

7. The language should support post-factum configuration, since The best argument I can provide in favor of this requirement is that it has many applications. There are many software systems developed separately in the past without any anticipation for their possible use as computational components in the configurations of larger and newer systems.

C Conic and its Successors

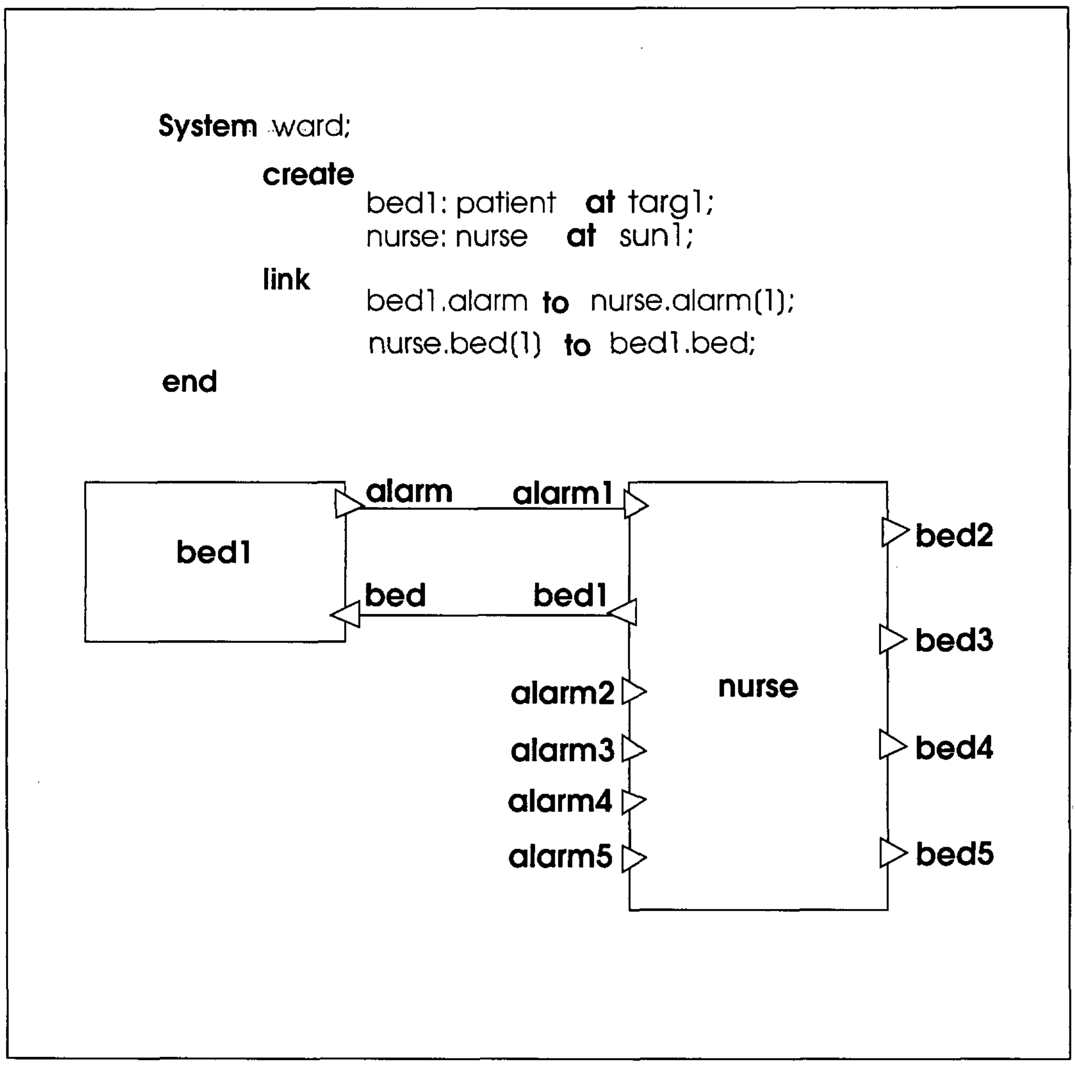
Developed by Jeff Kramer and others of Imperial College, Conic and its successors, Rex and Regis, are environments for constructing and executing distributed systems. The configuration languages used in these environments are reviewed in this section.

```
group module monitor(Tfactor, Pfactor: integer);
        exitport
                press, temp: real    reply    signaltype;
        entryport
                control: boolean;
        use
                scale, sensor;
        create
                temperature: sensor;
                pressure: sensor;
                Tscale: scale(Tfactor);
                Pscale: scale(Pfactor);
        link
                temperature.output to  Tscale.input;
                pressure.output  to Pscale.input;
                Tscale.output  to temp;
                Pscale.output to  press;
                control  to   Tscale.control, pscale.control;
end
```



A group module configuration (From (Magee, Kramer & Sloman 1989))

Figure 1: A sample program segment of Conic



```
System ward;
        create
                bed1: patient  at targ1;
                nurse: nurse   at sun1;
        link
                bed1.alarm to nurse.alarm(1);
                nurse.bed(1) to bed1.bed;
end
```

A patient monitoring system (From (Kramer, Magee & Ng 1989))

Figure 2: A sample display of ConicDraw

C.1 Conic

Distributed systems in Conic (Kramer & Magee 1985, Kramer & Magee 1990, Magee et al. 1989), are constructed from *logical nodes*, which are composite components in our terminology. Logical nodes are hierarchically composed from *group modules*, which are compositions of *task modules* and/or simpler group modules. Group modules are also composite components, and can be nested to any level. A logical node is the unit of distribution and dynamic reconfiguration. Task modules, which are the individual software components (i.e., computational components), are programmed in a version of Pascal extended to include a fixed set of message-passing constructs. Thus the communication and computation aspects of distributed systems in Conic are combined. Configuration and computation, however, are totally separate.

Component interaction structure in Conic is based on indirect connection between modules. All inter-module communications go through *exitports* and *entryports* of the modules (See the sample program segment shown in Figure 1). As mentioned above, Conic totally separates module programming from the configuration. A module, therefore, cannot directly refer to any other module.

Conic provides some component interaction safety, but at a cost of combining communication and computation. Exitports and entryports are strongly typed and connections are unidirectional from exitports to entryports except for the *request-reply* ports, where a transmitted message requires a reply.

The Conic configuration language does not support dynamic reconfiguration. However, dynamic reconfiguration of distributed systems in a Conic environment is still possible using the Conic *configuration manager*, which is a run-time support mechanism external to the configuration program. The configuration manager, therefore, can support only ad-hoc reconfiguration; one cannot program a pre-planned reconfiguration in a Conic environment. Dynamic reconfigurations, specified in terms of the *create, re-*

move, link and *unlink* primitives, take place when the affected nodes are "quiescent". A node is *quiescent* if it is not currently engaged in a communication and no other process has initiated a transaction requiring a service from this node. Kramer and Magee prove that the quiescent state is reachable (Kramer & Magee 1990). Since the units of dynamic reconfigurations are logical nodes, only modules or group modules that form a logical node can be dynamically reconfigured.

Conic does not support different computational programming languages. Computational components, as mentioned above, are programmed in a variant of Pascal.

Conic does not support post-factum configuration.

Another limitation of the Conic configuration language is that it can only express configuration of the modules within a logical node. The configuration manager, which uses an interpretive version of the configuration language, is then used to configure the logical nodes into a distributed application.

C.2 ConicDraw

ConicDraw (Kramer et al. 1989) provides graphic support for part of the Conic configuration language. Using ConicDraw the configuration of an executing distributed system in a Conic environment can be graphically represented. ConicDraw maintains an up-to-date view of the system by communicating with the Conic configuration manager. The user may change the system configuration by editing the configuration diagram displayed by ConicDraw.

Figure 2 provides A sample display of ConicDraw.

C.3 Rex and Regis Configuration Languages

Rex (Crane & Dulay 1992, Kramer 1990, Kramer, Magee, Sloman & Dulay 1992), a distributed systems development environment, is the successor of Conic. The configuration language used in Rex is Darwin, which is reviewed in the next section. Rex is apparently not supported anymore, possibly because its successor, the Regis environment (Crane & Twidle 1994, Magee, Dulay & Kramer 1994), provides more flexibility. The configuration language used in Regis is also Darwin; however, Regis separates configuration, computation, and communication. Computational components in Regis, as discussed below, interact using user specified communication primitives. In Conic and Rex computation and communication are integrated in computational components.

C.4 Darwin

Darwin (Magee, Dulay & Kramer 1993, Magee et al. 1994) like its predecessor, the Conic configuration language, separates configuration programming from computational programming. Darwin also removes some of the restrictions of Conic. Unlike Conic, Darwin can express dynamic reconfiguration of system structures. Furthermore, Darwin is independent of the languages used to program computational components. However, the base language, such as C or Pascal, used for computational programming must still be extended to include specific message-passing constructs which are then used for communications. Darwin, in a Rex environment, requires computational components to use a fixed set of communication primitives, i.e., combining communication and computation, provided by Rex as an extension to the supported programming languages. In a Regis environment, however, Darwin allows computational components to interact using user specified communication primitives, which partially separates communication and computation.

Darwin uses indirect component binding for specifying interaction structures of distributed systems.

To increase component interaction safety, Darwin provides three kinds of constraint checks: directionality, connectivity, and typing constraints. For the directionality constraint, Darwin requires two types of component interfaces: *provides* which identify services provided by the component and *requires* which identify services required by the component (Figure 3).

Bindings are allowed only between interfaces with compatible directionality. For the connectivity constraint, Darwin does not allow *fan-out* bindings, where one provided interface is bound to more than one required interfaces. *Fan-in* bindings, where two or more provided interfaces are bound to one required interface, are legal. Multicasting that can be accomplished by fan-out bindings requires some kind of *interface set* abstraction to define and control one-to-many bindings; Darwin currently does not provide such an abstraction. Interface names are associated with typing constraints. These are checked by the Darwin *interface typing system* to ensure that the bound interfaces are of compatible types.

Darwin supports hierarchical composition by providing the concept of "composite component types". A Darwin configuration structure consists of a hierarchy of interconnected component instances. The leaves of the hierarchy represent instances of computational components (without Darwin sub-configuration) called *primitive component types*. The higher levels of the hierarchy are described by Darwin configuration descriptions called *composite component types*.

Darwin supports dynamic reconfiguration, in a Rex environment, by providing a configuration operation, called *op*, which is a mechanism for describing dynamically evolving structures. Configuration changes are programmed using this operation and triggered at run time by the application (computational components). For dynamic reconfiguration in a Regis environment, Darwin provides facilities for dynamic component instantiation and binding but not for deletion or unbinding; components are expected to die when they complete their tasks. The instantiations are either *direct instantiations*, where a computational or com-

```
process suppervisor;
        provide resultP;
        var
                resultP: port(resultT, @replyT);
                putproblemP: @replyT;
                R: resultT;
        begin
                initialise;
                repeat
                        resultP? (R, putproblemP)
                        if R ≠ null then store(R);
                        putproblemP!(allocate(globalproblem));
                until done;
                finalise;
                halt;
        end

        process worker;
        require resultP;
        var
                resultP: @port(resultT, @replyT);
                getproblemP: replyT;
                subproblem: problemT;
        begin
                resultP? (null, @getproblemP);
                loop
                        getproblemP?(subproblem);
                        resultP!(compute(subproblem), @getproblemP);
                end
        end

        component supervisor_worker {
                use supervisor, worker;
                inst supervisor;
                forall k : 1..Pmax{
                        inst worker[k].@pe = k;
                        bind worker[k].resultP – supervisor.resultP;
                }
        }
```

The supervisor-worker problem(From (Magee et al. 1993))

Figure 3: A sample program of Darwin

posite component is instantiated directly at run-time, *lazy instantiations*, where the component is instantiated at run-time only if and when a binding to that component is triggered, or *recursive instantiations*, where the component is recursively instantiated within its own definition.

Darwin supports, as mentioned above, different computational programming languages.

Darwin does not support post-factum configuration. Darwin in a Regis environment does provide some support for open distributed systems using a name server called *RND*, the Regis Name Daemon. This support, however, does not include post-factum configuration, where the interfaces of a computational component may not meet Darwin's requirements.

Darwin also provides a graphical notation for representation of a program's instance structure for a particular parameterization. In this notation, components are represented by arc-boxes and instances are represented by rectangles. Required and provided services are represented by empty and solid circles, respectively (Pryce & Crane 1998).

Darwin also allows specification of hardware configurations.

D Polylith

Developed by James Purtilo and others of the University of Maryland (Agnew, Hofmeister & Purtilo 1994, Hofmeister, Atlee & Purtilo 1990, Hofmeister, White & Purtilo 1993, Purtilo 1994), Polylith is a *software interconnection system.* Polylith separates configuration, computation, and communication. Configuration is expressed in the Polylith *Module Interconnection Language* (MIL). Communications between components are provided by a Polylith *software bus.* An application may change only the software bus to use a different communication mechanism without affecting the configuration or the computational components.

Polylith uses indirect component binding via a software bus for specifying interaction structures of distributed systems.

To increase component *interaction safety*, Polylith provides two kinds of constraint checks: directionality and typing constraints. For the directionality constraint, component interfaces can be defined as either unidirectional or bidirectional. Two types of unidirectional interfaces are specified: *source* for outgoing messages and *sink* for incoming messages. A source interface must be bound only to a sink interface. Two types of bidirectional interfaces are defined: *client* which initiates communication and accepts a response and *function* which accepts communication and returns a response. A client interface must be bound to a function interface. For the typing constraint, interface names are followed by the typing constraints which are checked by Polylith to ensure that the bound interfaces are of compatible types. (A sample program segment of Polylith is shown in Figure 4.)

Polylith MIL supports hierarchical composition by allowing a module definition to include nesting of additional modules.

The Polylith configuration language does not support dynamic reconfiguration. However, Polylith provides another language called *Clipper* for dynamic reconfiguration of the distributed systems in the Polylith environment. Using Clipper, a programmer can write a reconfiguration script which is basically a set of alternative configurations and certain events that will trigger transformations from one configuration to another. The script is compiled and executed as a special module. When a specified event is detected, the corresponding reconfiguration takes place. The triggering event can be generated by the application, like a signal from a module, by the distributed environment, like the sudden failure of a module, or by the system environment, like the failure of a module's host machine.

Polylith supports different computational programming languages. Software configuration in Polylith is independent of the languages used for programming software components; an application can be configured from software components of mixed programming languages.

Post-factum configuration is possible in Polylith; however, the existing computational components must be packaged for the Polylith software interconnection system, replacing direct communications by Polylith bus calls. This packaging can be accomplished automatically by a Polylith software packager, called *Polygen* (Callahan & Purtilo 1991). Currently, Polygen can package computational components that use point-to-point communication or RPC.

E LOTOS/ODP

LOTOS (Logrippo, Faci & Haj-Hussein 1992, Vogel 1994, Vogel, Bochmann, Dini & Polze 1994) Developed within the framework of Open Systems Interconnections (OSI), LOTOS is the standard Formal Description Technique (FDT) for Open Distributed Processing (ODP). ODP is an ongoing joint standardization effort by ISO and CCITT to provide a reference model for open distributed systems. Based on five levels of abstraction, called *viewpoints*, ODP attempts to reduce the complexity of distributed systems by partitioning concerns. Each viewpoint provides a different abstraction of the system by focusing on a different set of concerns. The five levels of abstractions are *enterprise, information, computational, engineering*, and *technology* viewpoints. The computational viewpoint is concerned with the functional decomposition of the system into objects suitable for distribution, while the engineering and technology viewpoints focus on the distribution infrastructure and choice of technology to support distribution. An ODP system configuration, therefore, is primarily expressed by the computational viewpoint, while the implementation support is provided by the technology viewpoint.

Computational components as well as composite compo-

```
sevice "hello" : {
        implementation : { binary : "./greet.exe" }
        source "send" :{string}
}

service "print" : {
        implementation { binary : "./pr_source_sink.exe"}
        sink "msg1" : {string}
        sink "msg2" : {string}
        sink "msg3" : {string}
}

orchestrate "source_sink" : {
        tool "hello"
        tool "hi" : "hello"
        tool "greedings" : "hello"
        tool "print"
        bind "hello send" "print msg1"
        bind "hi send" "print msg2"
        bind "greedings send" "print msg3"
}
```

A Polylith application description (From (Hofmeister et al. 1990))

Figure 4: A sample program segment of Polylith

nents, in LOTOS, are represented as *processes*. The whole system is also a LOTOS process. Processes interact with each other through *gates*. Process interactions are called *events*. An event, therefore, is associated with the gate at which the event takes place. The *observable behavior* of a LOTOS process is defined by a *behavior expression* as sequences of allowed events.

LOTOS supports hierarchical composition by providing an operator, called *hide*, which hides actions that are internal to a process. Using LOTOS, a system can be composed as a hierarchy of interconnected processes, hiding details of internal structure of a process from the higher levels. LOTOS also supports sequential and parallel composition of behaviors. Sequential compositions are accomplished by passing *exit* parameters of a behavior expression, using the operators *enable* and *accept*, to the next one. Parallel compositions are expressed by the *interleaving* operator, when no synchronization is needed, and by the *selective* parallel operator, when the processes must be synchronized on some common events. Parallelism, in LOTOS, is equivalent to expression of choice provided by the operator *choice* which expresses a nondeterministic choice between behaviors.

Although LOTOS processes appears as black boxes to their environment and all interprocess interactions go through gates, as noted above, the processes are allowed to have common gates. LOTOS, therefore, does not separate the system behavior and structure. This may well be expected from LOTOS as an FDT (Formal Description Technique). From the configuration point of view, however, combining the configuration and behavior of a distributed system is an unnecessary complexity. Vogel's work, as described below, may eliminate this complexity.

Andreas Vogel (Vogel 1994) introduces a LOTOS sublanguage, called *LOTOScomp*, for the ODP computational viewpoint. LOTOScomp describes the ODP architecture in terms of communicating computational objects. LOTOScomp, therefore, can be used as a configuration language for distributed software systems. However, the LOTOS sub-language approach (including LOTOScomp) is still evolving and, to my knowledge, it has not been applied to any large-scale system.

LOTOScomp defines a LOTOS process, called *Object Space*, which includes instantiation and termination of computational objects. In addition, Object Space contains another LOTOS process, called *computational infrastructure* which specifies the communication between computational objects. ODP does not further specify the computational infrastructure. (A sample program segment of LOTOS is shown in Figure 5.)

Andreas Polze (Polze 1993), however, proposes a shared memory approach, using a C++ based programming environment, also called *Object Space*[1] for specifying interaction structures of distributed systems. Computational com-

[1]The concept "Object Space" is overloaded here; the first Object Space, the LOTOS process, is different from the second one, the C++ based programming environment. The second one is considered as a suitable engineering mechanism to implement the computational infrastructure.

```
behaviour

hide announcement, invocation, termination in

object_space[ req, resp, announcement, invocation, termination ]
|[ announcement, invocation, termination ]|
infrastructure[ announcement, invocation, termination ]

where
process object_space[ req, resp, announcement, invocation, termination ]: noexit :=

      hide get, put in
      binder[ put, get, announcement, invocation, termination ]( b )
      |[ get, put ]|
      ( server[ put, invocation, termination ]( p_op_if, md )
      |||
      client[ get, invocation, termination ] ( p_op_if, c_op_if, c_stream_if ))

where
process server[put, invocation, termination] ( op_if: ident, movie_dir: dir ): noexit :=

      hide control in
      stream_interface_manager[ control, put ]
      |[ control ]|
      server_operational_interface[invocation, termination, control, put] (op_if, movie_dir)

where
. . .
```

Part of a video-on-demand system specification (From (Vogel 1993))

Figure 5: A sample program segment of LOTOS

ponents, then, can write to and read from Object Space. This method of communication decouples the components.

To increase component interaction safety, Polze's approach provides a constraint: each component attempting to read an object from the Object Space presents a template; the object then can only be read by the component if the object matches the template.

LOTOS (and LOTOScomp) does not support dynamic reconfiguration. There is an unpublished paper (Schoo, Roth & Vogel n.d.), however, which proposes an extension to LOTOS, called $LOTOS_R$, demonstrating the potential of the language to support dynamic reconfiguration.

LOTOS does not support different computational programming languages. LOTOS specification of a distributed system (including computational components) can be transformed, theoretically, into any programming language. The feasibility of such a transformation into C code has been shown (Vogel 1994). Computational components (and configuration), however, must originally be written in LOTOS.

LOTOS does not support post-factum configuration.

F Raven

Developed by Terry Coatta and others of the University of British Colombia (Acton, Coatta & Neufeld 1992, Coatta 1994, Coatta & Neufeld 1994), Raven Configuration Management System (RCMS) is designed for constructing and managing object-oriented distributed software systems. Applications in RCMS are based on "composite objects" and "constraints", which are expressed by Raven Configuration Language (RCL). Computational components are expressed in RCL as objects, also called *elementary objects*. A *composite object* is a composition of some elementary objects and previously defined composite objects. The *constraints* are expressions in first order predicate logic. A group of related constraints form a *configuration specification*, which defines the relationships between the components of a composite object. Constraints includes an RCL operation, called *link*, which provides communication between objects. Although RCMS handles configuration independent of computation, all three aspects of a distributed system are expressed by an RCL program and the separation of concerns is not clear.

Connections between components, as mentioned above, is provided by the link operation. Link uses the network approach of indirect connection; the resulting component interaction structure decouples the components.

Component interaction safety can be increased by the constraints.

RCL supports hierarchical composition, as noted above, by means of composite objects.

RCL supports a pre-planned dynamic reconfiguration by providing a mechanism called *repair*. This mechanism reconfigures the system to a pre-specified configuration under certain conditions. These conditions are specified by the constraints.

Currently, RCL does not support different computational programming languages. Basically, RCL does not restrict computational components to a particular programming language. Currently, however, the only computational programming language supported by the RCMS run-time environment, the Raven system, is the Raven language. Computational components in RCMS, therefore, are programmed using the Raven language. An RCL program describing a distributed system is also translated into the Raven language. The entire system then is translated into the C programming language. Currently, however, like LOTOS, computational components must originally be written in one specific language, in this case, the Raven language.

RCL does not support post-factum configuration. Basically, RCL can make use of any Raven object, including an object that is not explicitly designed for RCMS. However, as noted above, the RCMS run-time environment, the Raven system, imposes certain restrictions that does not allow RCL to support post-factum configuration.

Figure 6 provides a sample program segment of Raven.

G MetaH

Developed by Steve Vestal and others of Honeywell (Vestal 1994*b*, Vestal 1994*c*), MetaH is a configuration language designed for real-time avionic systems. A distributed system in MetaH consists of a collection of interconnected "entities". An *entity* is either a *low level entity*, describing a computational component, or a *high level entity*, describing how previously defined entities are combined to form a new one. The basic low level entity is a process which is the unit of scheduling, distribution, and fault and security containment. High level entities include *macros*, which are collections of processes and connections, and *modes*, which are also collections of processes and connections; a mode further specifies the collection of processes and connections that are in effect during that mode of operation. A system configuration, therefore, can be defined in terms of modes; and a reconfiguration can be accomplished by changing the mode of operation. Considering that the modes are all predefined, dynamic reconfigurations must be pre-planned.

Designers of MetaH support the methodology of using the language for describing generic configurations, where once a generic configuration is developed, it can be easily tailored to the requirements of a specific product. Each specific configuration then can draw its types and source code of computational components from a growing common pool.

MetaH is supported by a set of tools, the MetaH *toolset*. Using a MetaH specification, these tools can perform realtime schedulability analysis, reliability and security analysis, and automatic assembly of the specified configuration.

MetaH is primarily designed for describing dependable and efficient systems. The language, therefore, utilizes static checking, analysis, and code generation, while mini-

```
class SymbolTable {
        behave add(key : String key, item : Object);
        behave lookup(key : String) : Object;
        behave containsItem(item : Object) : Int;
        }

composite DirTree {
        constructor(symbolTable : Root);
        root : SymbolTable;
        nodes : Set[SymbolTable] indirect;
        }

specification {
        define Path(X, Y) < – Y in X |
                there exists Z [Z in X]
                        Z.instanceOf() == SymbolTable & Path(Z, Y);
        1: root = Root;
        2: nodes := all x : Path(root, X);
        3: for all X ~Path(X, X);
        }

repair {
        atomic;
        }
```

A composite object declaration for a binary tree (From (Coatta & Neufeld 1994))

Figure 6: A sample program of Raven

mizing dynamic run-time decision-makings.

In most part, MetaH keeps configuration programming separate from the computation. However, a process is allowed to share certain components, like a monitor or a package, with other processes. Furthermore, the I/O ports of a process are typed according to the I/O buffer variables of the corresponding computational component and using the corresponding computational language typing system. This coupling of computation and configuration is not a desirable features for a configuration language. There is, however, a mechanism in the language that allows automatic selection of the actual type declarations from the source code of computational components. In addition, MetaH has another mechanism that allows every implementation of a process that shares a component to automatically include the shared component. Although these mechanisms do not eliminate the tight coupling between computation and configuration, but they do reduce the complexity of it.

For component interaction structure, MetaH uses a network approach of indirect connection between components. All inter-process communications go through input and output *ports* of the processes (Figure 7).

For interaction safety, MetaH provides two kinds of constraints: directionality and typing. For the directionality constraint, MetaH allows unidirectional connections between I/O ports of the processes. For the typing constraints, input and output ports of a process, as mentioned above, are typed using the corresponding computational language typing system. This provides a good measure of interaction safety, however, at a cost of coupling configuration and computation.

MetaH, as noted above, supports hierarchical composition by allowing an entity definition to include previously defined entities.

MetaH supports a pre-planned dynamic reconfiguration, as noted above, by changing the mode of operation, where different *modes* specify different (alternative) configurations. At run time, the mode change and consequently the reconfiguration can be triggered by any authorized component.

Currently, MetaH does not support different computational programming languages. Computational components, in MetaH, are programmed in Ada. Creators of MetaH are planning, in the future versions of MetaH, to also support C modules and ultimately a mixture of components in different programming languages. They have already demonstrated the feasibility of support for different computational programming languages by a hand-configured system of mixed Ada and C components (Vestal 1994*a*).

MetaH does not support post-factum configuration. Cre-

```
with port type PTYPES;
          process FOO is
                    A: in port PTYPES.TYPE_1;
                    B: out port PTYPES.TYPE_1;
          end FOO;

          process implementation FOO.BAR is
                    attributes
                              self'Period := 25 ms;
                              self'Sourcefile := "foo_bar.ada";
          end FOO.BAR

          process implementation FOO.RAB is
                    attributes
                              self'Period := 50 ms;
                              self'Sourcefile := "foo_rab.ada";
          end FOO.RAB

with port type PTYPES;
          macro WIDGET is
                    A: in port PTYPES.TYPE_1;
                    B: out port PTYPES.TYPE_1;
          end WIDGET;

          macro implementation WIDGET.BAR is
                    P1: periodic process FOO.BAR;
                    P2: periodic process FOO;
                    connections
                              P1.A < – A;
                              P2.A < – P1.B;
                              B < – P2.B;
          end WIDGET.BAR;
```

A MetaH example (From (Vestal 1994*c*))

Figure 7: A sample program of MetaH

ators of MetaH recognize the importance of post-factum configuration. They have tried, successfully, using some existing commercial navigation products as computational components within MetaH (Vestal 1994*a*). There are certain coding guidelines and restrictions, however, imposed by MetaH on the computational components. These guidelines must be relaxed, as the creators of MetaH have done so to a certain extent, in order to be able to write a MetaH description of an existing component. Otherwise, either a packager must be used, like Polylith does, or the source code must be edited, adapting the existing component to the requirements of MetaH.

MetaH also allows specification of hardware configurations.

H Durra

Developed by Mario Barbacci and others of Carnegie Mellon University (Barbacci, Doubleday, Gardner, Lichota & Weinstock 1991, Barbacci, Weinstock, Doubleday, Gardner & Lichota 1993), Durra separates configuration, computation, and communication by recognizing two kinds of system components: *tasks* and *channels*. Tasks are active components (computational components); they implement the functionality of the system and initiate all message-passing operations. Channels are passive; they implement communication facilities and react to requests from tasks. The language is based on *task description* and *channel description*. Groups of tasks and channels are compiled into executable programs, called *clusters*, which are units of distribution. The run-time support portion of a cluster, the *cluster manager* is a piece of code generated by the Durra compiler.

Component interaction structure, as noted above, is based on channels which provide the connections between components. To use a different kind of communication mechanism, one may change the channel without affecting the computation or configuration aspects of the system; using channels for communication decouples the components.

For component interaction safety, the I/O ports of tasks and channels are typed and connections are unidirectional. These typing and directionality constraints limit component interactions only to the messages of compatible types and directionality, reducing the possibility of unintended interactions. Notice that the Durra configuration programming, separate from the computational programming, includes these constraints, while in MetaH, for example, although the measure of interaction safety is even stronger, the configuration language uses the typing information and system of computational components, combining computation and configuration.

Durra supports hierarchical composition by providing the concept of *compound task description*, which is a composition of simpler task and channel descriptions. An application description is in fact a compound task description.

Durra supports a pre-planned dynamic reconfiguration by allowing an application description to include a set of alternative configurations and a set of conditional configuration transitions. At run time, then, a reconfiguration takes place if the associated condition is met. Like the Conic configuration manager, Durra also makes sure that the affected processes are quiescent[2] before the reconfiguration.

Considering the structure of the Durra configuration language, Durra seems capable of supporting different computational programming languages. Currently, however Durra supports only one computational programming language, Ada. The Durra compiler, as mentioned above, generates a piece of code, the cluster manager, which acts as a communication/reconfiguration manager for a cluster. Historically, considering that the cluster manager is in Ada, it was less effort for the creators of Durra to develop the application examples in Ada. In the past Durra also supported C, but it was dropped in favor of Ada to save effort. The choice of Ada as the computational programming language, therefore, is not because some Ada feature, e.g., synchronization, is central to Durra. Durra can support other computational programming languages; currently, however, it does not.

Durra does not support post-factum configuration.

A sample program segment of Durra is shown in Figure 8.

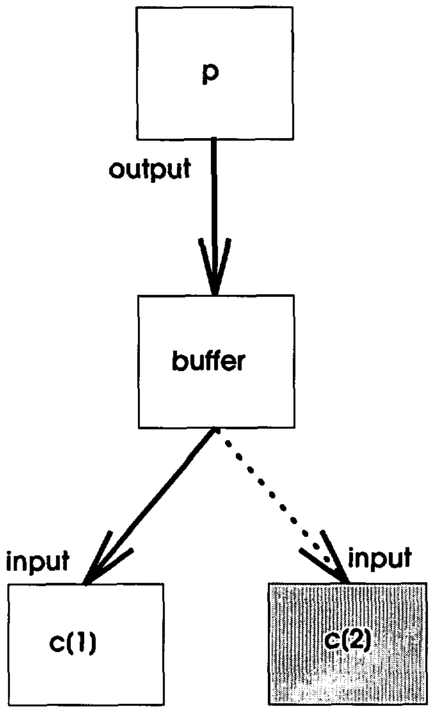
I Argus

Developed by Barbara Liskov and others of MIT (Bloom & Day 1993, Liskov 1988, Liskov, Day, Herlihy, Johnson, Leavens(ed.), Scheifler & Weihl 1987), Argus is a language for constructing and maintaining distributed systems and an environment for their execution. Distributed systems in Argus consist of "guardians". A *guardian* is an abstract object that encapsulates some resources. A guardian can interact with other guardians, accessing their resources, by means of special procedures, called *handlers*, that can be remotely invoked. Argus uses a message-passing communication mechanism for implementing handler calls. A guardian can dynamically create other guardians; it can also destroy itself. Guardians are units of distribution and reconfiguration; they are, in fact, the computational components of distributed systems in Argus.

The Argus programming language is used for both programming the computational components (guardians) and describing the system configuration. Argus does not separate the three aspects of distributed systems, computation, configuration, and communication. Argus is designed primarily for applications that require concurrent on-line transactions, where data must be kept consistent in spite of concurrent access and possible hardware failures. For these applications Argus may work well; however, from a configuration point of view, combining computation and configuration is a weakness.

[2]For a definition of quiescence see Section C.1, Page 228

```
task dynamic_producer_consumer
    components
        p: task producer;
        c(1...2): task consumer;
        bufer: channel fifo(message, 10);
    structures
        L1: begin
            baseline p, c(1), buffer;
            buffer: p.output >> c(1).input;
        end L1;
        L2: begin
            baseline p, c(2), buffer;
            buffer: p.output >> c(2).input;
        end L2;
    reconfigurations
        enter L1;
        L1 => L2 when signal(c(1), 1);
    clusters
        cl1: p, buffer;
        cl2: c(1), c(2);
end dynamic_producer_consumer;
```



A producer-consumer application with dynamic reconfiguration (From (Barbacci et al. 1993))

Figure 8: A sample program segment of Durra

Inter-guardian communication, as noted above, is provided by handler calls of the guardians. Handlers make direct connections between guardians; and the resulting component interaction structure tightly couples the components. This is another weakness of Argus as a configuration language.

On the positive side, as a point of strength, Argus provides a good measure of interaction safety, however, at a cost of the tight coupling of components. Furthermore, Argus transactions, also called *actions*, are atomic; they either *commit* or *abort*. If an action aborts, the state of affected objects is restored to the pre-transaction state, as if the transaction was never attempted. An action can include other actions, making a group of transactions be performed atomically.

From a configuration point of view, considering that internal structures of guardians are computational issues, Argus does not support compositionality. There is a concept called *subsystem* which describes a group of guardians cooperating to provide a single service. However, a subsystem is an abstract structure and is not represented as an entity in the Argus language.

Argus provides strong support for dynamic reconfiguration. As noted above, guardians and their handlers can be created and destroyed dynamically. Furthermore, the atomicity of an action which may include some sub-actions, expressing a reconfiguration, makes either the reconfiguration successfully take place or the system state remain unchanged.

Argus does not support different computational programming languages. Computational components, guardians, as well as configuration of a distributed system in Argus environment are programmed in the Argus language.

Argus does not support post-factum configuration.

Figure 9 provides a sample program segment of Argus.

J Summary and Conclusion

This paper has provided a list of features and requirements as a set of criteria for reviewing some configuration lan-

```
brach = guardian is create handlers total, open, close, deposit, withdraw

          % type definitions . . .

          create = creator (c: string, size: int) returns (branch)
                    code := c
                    seed.value := 0
                    ht := htable$new()
                    for i: int in int$from_to(1, size) do
                              htable$addh(ht, bucket$new())
                              end
                    return(self)
                    end create

          total = handler() returns(int)
                    sum: int := 0
                    for b: bucket in htable$elements(ht) do
                              for p: pair in bucket$elements(b) do
                                        sum := sum + p.acct.bal
                                        end
                              end
                    return(sum)
                    end total

          open = handler() returns(account_number)
                    intcell$write_lock(seed)
                    a: account_number = account_number${code: code, num: seed.val}
                    seed.val := seed.val + 1
                    bucke$addh(ht[hash(a.num)], pair${num: a, acct: acct_info${bal: 0}})
                    return(a)
                    end open

          close = handler(a: account_number) signals(no_such_acct, positive_amount)
          . . . .

          end branch
```

Part of A branch guardian for a banking application (From (Liskov 1988))

Figure 9: A sample program segment of Argus

guages. On the basis of these criteria, the paper has presented a review of some configuration languages. Table 1 provides a summary of this review.

As the table shows, none of the reviewed languages satisfies all the requirements. The closest language to satisfying the requirements is Polylith. Polylith, however, is not a configuration language; it is a software interconnection system with a collection of tools. In Polylith, for example, static configuration is expressed by Polylith MIL, dynamic reconfiguration is described by Polylith Clipper, and Post-factum configuration can be accomplished by using the Polylith software packager, Polygen. One may rightfully argue that it is a point of strength for Polylith to provide a platform for constructing, executing, and maintaining dynamically evolving distributed software systems, where a collection of different tools are available for different tasks. Strictly from a configuration point of view, however, a more coherent approach, a language satisfying all the requirements, is preferable. After Polylith, the configuration language that comes closest to meeting the requirements is Darwin, whose major weakness is its inability to support post-factum configuration. The remaining reviewed languages, as the table shows, have even more weaknesses with respect to the requirements.

Other related work not reviewed here includes:

- CORBA, the Common Object Request Broker Architecture (OMG 1995), which is a standardized system integration framework based on distributed object technology. CORBA can be used for integration of heterogeneous hardware and software components (Polze, Wallnau & Plakosh 1998). Dynamic reconfiguration in CORBA framework is also possible (Bidan, Issarny, Saridakis & Zarras 1998).

- Aster (Issarny & Bidan 1996, Issarny, Bidan & Saridakis 1998), which is an interconnection language aimed at building customized distributed runtime systems.

- Gandiva (Wheater & Little 1996), which is an object-oriented software development system, aimed at separating software components into their interface and implementation components. It provides a set of C++ classes to support the construction of interface classes from implementation classes.

- Warren and Sommerville (Warren & Sommerville 1996) also present a model for dynamic reconfiguration of distributed systems. Their model is aimed at evolving system architecture, causing low execution disruption, but requires system specific features and is application dependent.

In conclusion, design and creation of a configuration language for distributed software systems, or improvement and extension of an existing one, satisfying the requirements presented in this paper is still an active area of research. Specifically, the following topics of research are identified in this area:

- Some configuration languages restrict the computational components to specific programming languages, e.g., MetaH, Durra. An ongoing research is to provide support for computational components of mixed programming languages.

- Some configuration languages are used for formally describing the entire distributed system, e.g., Raven, LOTOS. An ongoing research is to translate these formal descriptions into a suitable programming language.

- In the configuration languages supporting dynamic reconfiguration, capturing and restoring the state of the affected computational components without programmer intervention is still an ongoing topic of research.

- Dynamic reconfigurations expressed by the current configuration languages require computational component participation. The programmer has to manually adapt a computational component for this participation. Polylith Surgeon (Hofmeister et al. 1993) automates the component participation for a class of components. Generally, however, dynamic reconfiguration without component participation is an open problem.

- Finally, post-factum configuration, expressed by a configuration language, is another topic of research. Current configuration languages are not capable of expressing post-factum configuration. Polylith software packager, Polygen (Callahan & Purtilo 1991), is a great accomplishment in this area. However, it has to be integrated into a configuration language, providing a unified approach for expressing configurations, where computational components are packaged as required transparent to the programmer.

Acknowledgements

Many thanks to Professor David Lamb for his comments. I would like to also thank Jeff Kramer, Andreas Vogel, Terry Coatta, Steve Vestal, Mario Barbacci, and Mark Day for reviewing parts of this paper and providing me with their comments.

References

[1] Acton D., Coatta T. & Neufeld G. (1992), The Raven system, Technical Report TR 92-15, University of British Columbia, Department of Computer Science.

[2] Agnew B., Hofmeister C. & Purtilo J. (1994), Planning for change: A reconfiguration language for distributed systems, *in* 'Proceedings of the Second International Workshop on Configurable Distributed Systems', Pittsburgh, Pennsylvania, p. 15–22.

LANGUAGE	REQUIREMENTS AND FEATURES						
	Component interaction	*Support for compositionality?*	*Support for dynamic reconfiguration?*	*Support for post-factum configuration?*	*Different computational programming languages?*	*Support for separation of computation and configuration?*	*Separation of computation, configuration, and communication?*
Argus	direct	no	yes	no	no	no	no
Conic	indirect/network	yes	no[a]	no	no	yes	no
Darwin	indirect/network	yes	yes	no	yes	yes	yes[b]
Durra	indirect/network	yes	yes	no	no	yes	yes
LOTOS	indirect/shared memory	yes	no	no	no	partially	no
MetaH	indirect/network	yes	yes	no	no	partially	no
Polylith	indirect/network	yes	no[c]	yes	yes	yes	yes
Raven	indirect/network	yes	yes	no	no	partially	no

[a]Reconfigurations in Conic are accomplished by Conic configuration manager
[b]Only in a Regis environment.
[c]Reconfigurations in Polylith are expressed by Polylith reconfiguration language, Clipper.

Table 1: Comparison of Configuration Languages

[3] Barbacci M. R., Doubleday D. L., Gardner M. J., Lichota R. W. & Weinstock C. B. (1991), Durra: A task level description language reference manual, Technical Report CMU/SEI-91-TR-18, Carnegie Mellon University, Software Engineering Institute.

[4] Barbacci M. R., Weinstock C. B., Doubleday D. L., Gardner M. J. & Lichota R. W. (1993), 'Durra: A structure description language for developing distributed applications', *IEE Software Engineering Journal* **8**(2), 83–94.

[5] Bidan C., Issarny V., Saridakis T. & Zarras A. (1998), A dynamic reconfiguration service for corba, *in* 'Proceedings of 4th International Conference on Configurable Distributed Systems', Annapolis, Maryland, USA, p. 35–42.

[6] Bloom T. & Day M. (1993), 'Reconfiguration and module replacement in Argus: Theory and practice', *IEE Software Engineering Journal* **8**(2), 102–108.

[7] Callahan J. R. & Purtilo J. (1991), A packaging system for heterogeneous execution environment, Technical Report UMCP TR 2542, University of Maryland, Computer Science Department.

[8] Coatta T. (1994), Configuration Management Using Objects and Constraints, PhD thesis, University of British Columbia, Department of Computer Science.

[9] Coatta T. & Neufeld G. (1994), Distributed configuration management using composite objects and constraints, *in* 'Proceedings of the Second International Workshop on Configurable Distributed Systems', Pittsburgh, Pennsylvania, p. 112–122.

[10] Crane S. & Dulay N. (1992), Constructing multi-user applications in Rex, *in* 'Proceedings of the IEEE International Conference CompEuro 92', Den Haag, p. 365–370.

[11] Crane S. & Twidle K. (1994), Constructing distributed Unix utilities in Regis, *in* 'Proceedings of the Second International Workshop on Configurable Distributed Systems', Pittsburgh, Pennsylvania, p. 183–189.

[12] Dean T. R. (1993), Characterizing Software Structure Using Connectivity, PhD thesis, Department of Computing and Information Science, Queen's University, Kingston, Canada.

[13] DeRemer F. & Kron H. (1976), 'Programming-in-the-large versus programming-in-the-small', *IEEE Transactions on Software Engineering* **2**(2), 114–121.

[14] Hofmeister C., Atlee J. & Purtilo J. (1990), Writing distributed programs in Polylith, Technical Report UMCP TR 2575, University of Maryland, Computer Science Department.

[15] Hofmeister C., White E. & Purtilo J. (1993), 'Surgeon: A packager for dynamically reconfigurable distributed applications', *IEE Software Engineering Journal* **8**(2), 95–101.

[16] Issarny V. & Bidan C. (1996), Aster: A framework for sound customization of distributed runtime systems, *in* 'Proceedings of the 16th International Conference on Distributed Computing Systems', Hong-Kong, p. 586–593.

[17] Issarny V., Bidan C. & Saridakis T. (1998), Achieving middleware customization in a configuration-based development environment: Experience with the aster prototype, *in* 'Proceedings of 4th International Conference on Configurable Distributed Systems', Annapolis, Maryland, USA, p. 207–214.

[18] Kramer J. (1990), Configuration programming – a framework for the development of distributable systems, *in* 'Proceedings of the IEEE International Conference CompEuro 90', Israel, p. 374–386.

[19] Kramer J. & Magee J. (1985), 'Dynamic configuration of distributed systems', *IEEE Transactions on Software Engineering* **11**(4), 424–436.

[20] Kramer J. & Magee J. (1990), 'The evolving philosophers problem: Dynamic change management', *IEEE Transactions on Software Engineering* **16**(11), 1293–1306.

[21] Kramer J., Magee J. & Ng K. (1989), 'Graphical configuration programming', *IEEE Computer* **22**(10), 53–65.

[22] Kramer J., Magee J., Sloman M. & Dulay N. (1992), 'Configuring object-based distributed programs in Rex', *IEE Software Engineering Journal, Special Issue on Object-Oriented Systems* **7**(2), 139–149.

[23] Liskov B. (1988), 'Distributed programming in Argus', *Communications of the ACM* **31**(3), 300–312.

[24] Liskov B., Day M., Herlihy M., Johnson P., Leavens(ed.) G., Scheifler R. & Weihl W. (1987), Argus reference manual, Technical Report MIT/LCS/TR-400, Massachusetts Institute of Technology, Laboratory for Computer Science.

[25] Logrippo L., Faci M. & Haj-Hussein M. (1992), 'An introduction to LOTOS: Learning by examples', *Computer Networks and ISDN Systems* **23**(5), 325–342.

[26] Magee J., Dulay N. & Kramer J. (1993), 'Structuring parallel and distributed programs', *IEE Software Engineering Journal* **8**(2), 73–82.

[27] Magee J., Dulay N. & Kramer J. (1994), A constructive development environment for parallel and distributed programs, *in* 'Proceedings of the Second International Workshop on Configurable Distributed Systems', Pittsburgh, Pennsylvania, p. 4–14.

[28] Magee J., Kramer J. & Sloman M. (1989), 'Constructing distributed systems in Conic', *IEEE Transactions on Software Engineering* **15**(6), 663–675.

[29] OMG (1995), The common object request broker architecture and specification (revision 2.0), Technical report, Object Management Group, Inc., Framingham, MA, USA.

[30] Polze A. (1993), The object space approach: Decoupled communication in C++, *in* 'Proceedings of TOOLS USA 93', Santa Barbara, p. 195–204.

[31] Polze A., Wallnau K. & Plakosh D. (1998), Real-time computing on off-the-shelf components – a case for corba, *in* 'Proceedings of the Third Biennial World Conference on Integrated Design and Process Technology', Vol. 6, Berlin, Germany, p. 70–77.

[32] Pryce N. & Crane S. (1998), Component interaction in distributed systems, *in* 'Proceedings of 4th International Conference on Configurable Distributed Systems', Annapolis, Maryland, USA, p. 71–79.

[33] Purtilo J. (1994), 'The Polylith software bus', *IEEE Transactions on Software Engineering* **16**(1), 151–174.

[34] Schoo P., Roth R. & Vogel A. (n.d.), On reflective extension of LOTOS. Gesellschaft für Mathematik und Datenverarbeitung, FOKUS, Berlin, Germany, Unpublished.

[35] Vestal S. (1994*a*), Private communication. Honeywell Technology Center, Minneapolis, MN.

[36] Vestal S. (1994*b*), *MetaH Reference Manual*, Honeywell Technology Center, Minneapolis, MN. Draft.

[37] Vestal S. (1994*c*), Mode changes in a real-time architecture description language, *in* 'Proceedings of the Second International Workshop on Configurable Distributed Systems', Pittsburgh, Pennsylvania, p. 136–146.

[38] Vogel A. (1993), LOTOS specification (computational viewpoint) of a simplified video-on-demand system. Université de Montréal, Département de'IRO, Unpublished.

[39] Vogel A. (1994), A formal approach to an architecture for Open Distributed Processing, Technical Report 902, Université de Montréal, Département de'IRO.

[40] Vogel A., Bochmann G. V., Dini P. & Polze A. (1994), Configuration in the framework of open distributed processing, *in* 'Proceedings of the Second International Workshop on Configurable Distributed Systems', Pittsburgh, Pennsylvania, p. 106–111.

[41] Warren I. & Sommerville I. (1996), A model for dynamic configuration which preserves application integrity, *in* 'Proceedings of the 3rd International Conference on Configurable Distributed Systems', Annapolis, Maryland, USA, p. 81–88.

[42] Wheater S. M. & Little M. C. (1996), The design and implementation of a framework for configurable software, *in* 'Proceedings of the 3rd International Conference on Configurable Distributed Systems', Annapolis, Maryland, USA, p. 136–143.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of **Slovenia** (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 219 385
Tlx.:31 296 JOSTIN SI
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Natalija Polenec

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of e-mails with the text in Informatica LaTeX format and figures in .eps format. The original figures can also be sent on separate sheets. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Rudi Murn, Informatica, Institut Jožef Stefan, Jamova 39, 1111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than five years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: ....

Title and Profession (optional): ....

....

Home Address and Telephone (optional): ....

....

Office Address and Telephone (optional): ....

....

E-mail Address (optional): ....

Signature and Date: ....

Informatica WWW:

http://ai.ijs.si/informatica/
http://orca.st.usm.edu/informatica/

Referees:

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Volume 26 Number 2 July 2002 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics