RC 21736 (97580) 12 April 2000
Computer Science/Mathematics

IBM Research Report

Using a Mandatory Secrecy and Integrity Policy on Smart Cards and Mobile Devices

Paul A. Karger, Vernon R. Austel, and David C. Toll
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - T.J. Watson - Tokyo - Zurich

Using a Mandatory Secrecy and Integrity Policy on Smart Cards and Mobile Devices

Paul A. Karger, Vernon R. Austel, and David C. Toll
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

Accepted for publication at the EUROSMART Security Conference
13-15 June 2000, Marseilles, France

Using a Mandatory Secrecy and Integrity Policy on Smart Cards and Mobile Devices

Paul A. Karger, Vernon R. Austel, and David C. Toll
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

1 Introduction

IBM®[1] has developed a new mandatory security model, combining both secrecy and commercial data integrity requirements [16]. This model was developed as part of an effort to design a high assurance operating system [6] for the Philips SmartXA chip – an operating system that could be evaluated at the highest security levels of the ITSEC [5] or the Common Criteria [2-4].

This paper shows how using the new security model can permit applications developers to solve security-related problems that could never be addressed before, either in smart cards or in larger computer systems. In particular, the security model and the operating system are designed to permit in-the-field downloading of applications written either in native languages (such as C or assembler) or in interpreted languages (such as Java Card™[2]). These downloaded applications could be mutually hostile, yet the operating system will prevent unauthorized interference between the applications, yet still allow controlled sharing of information between selected applications, subject to the constraints of the new security model.

The paper will cover three hypothetical applications – an electronic purse, an airline loyalty scheme, and cell phone – personal digital assistant (PDA) that handles classified message traffic in a military scenario. None of the examples are intended to be a precise guide to implementation. The examples are somewhat contrived to show the use of the security model. Applying the model in real applications would take significant application-specific design work. The paper will only summarize the aspects of the protection model itself. For full details of how the model was developed, see [16].

[1] IBM is a trademark of the International Business Machines Corporation in the United States, other countries, or both.

[2] Java Card is a trademark of Sun Microsystems, Inc. in the United States, other countries, or both.

2 What are Mandatory Access Controls?

There are two primary classes of access controls in computer systems – *discretionary access controls* and *mandatory access controls*.

Discretionary access controls are the commonly available security controls based on the fully general Lampson access matrix. They are called discretionary, because the access rights to an object may be determined at the discretion of the owner or controller of the object. Both access-control-list and capability systems are examples of discretionary access controls. The presence of Trojan horses in the system can cause great difficulties with discretionary controls. The Trojan horse could surreptitiously change the access rights on an object or could make a copy of protected information and give that copy to some unauthorized user. All forms of discretionary controls are vulnerable to this type of Trojan-horse attack.

Mandatory access controls have been developed to deal with the Trojan horse problems of discretionary access controls. The distinguishing feature of mandatory access controls is that the system manager or security officer may constrain the owner of an object in determining who may have access rights to that object. Mandatory access controls were developed to solve what Lampson has called the *confinement* problem [17] to control the leaking of information by Trojan horses.[3] Lipner [19] and Denning [12]

[3] A full treatment of these issues is beyond the scope of this paper. The interested reader

have shown that for *lattice security models*, unlike for fully general access matrices, it is possible to solve the confinement problem. All mandatory controls, to date, have been based on lattice security models. The use of mandatory access controls in smart cards has been proposed by Girard of GEMPLUS [14].

A lattice security model consists of a set of *access classes* that form a partial ordering. Any two access classes may be less than, greater than, equal to, or not ordered with respect to one another.

2.1 Secrecy Lattices

A very simple secrecy lattice might consist of two access classes: LOW and HIGH. LOW is less than HIGH. LOW is system low, and HIGH is system high. A slightly more complex example might be a list of secrecy levels, such as UNCLASSIFIED, CONFIDENTIAL, SECRET, and TOP SECRET. Each level in the list represents data of increasing secrecy.

There is no requirement for strict hierarchical relationships between access classes. The U.S. military services use a set of access classes that have two parts: a secrecy level and a set of categories. Categories represent compartments of information for which an individual must be specially cleared. To gain access to information in a category, an individual must be cleared, not only for the secrecy level of the information, but also for the specific category. For example, if there were a category NUCLEAR, and some information classified SECRET-NUCLEAR, then an individual with a TOP SECRET clearance would not be allowed to see that information, unless the individual were specifically authorized for the NUCLEAR category.

Lattice models were first developed at the MITRE Corporation by Bell and LaPadula [8] and at Case Western Reserve University by Walter [24] to formalize the military security model and to develop techniques for dealing with Trojan horses that attempt to leak information. At the time, no one knew how to deal with Trojan horses at all, and it came as quite a surprise that two quite simple properties could prevent a Trojan horse from compromising sensitive information.

First, the *simple security property* says that if a subject wishes to gain read access to an object, the access class of the object must be less than or equal to the access of the subject. This is just a formalization of military-security-clearance procedures that one may not read a document unless one is properly cleared.
Second, the *confinement property or *-property*[4] requires that if a subject wishes to gain write access to an object, the access class of the subject must be less than or equal to the access class of the object. The net effect of enforcing the confinement property is that any Trojan horse that attempts to steal information from a particular access class cannot store that information anywhere except in objects that are classified at an access class at least as high as the source of the information. Thus, the Trojan horse could tamper with the information, but it could not disclose the information to any unauthorized individual. A more detailed discussion of the confinement property and its interpretation in the context of a practical time-sharing system can be found in [8]. A survey on formal security models in general can be found in [18].

2.2 Integrity Lattices

Secrecy lattices, while useful for protecting against unauthorized information disclosure, do not deal with unauthorized tampering or sabotage of information. The early military models focused only on secrecy, and even Girard's proposal [14] for mandatory access control on smart cards is only a secrecy model. A commercial system, however, cannot be limited to only protecting the secrecy of information. Assuring that information is not tampered with is often much more important in a commercial setting.
Whether a smart card is used as a cash card or as

should consult Denning [11, chapters 4 and 5] for a more complete treatment.

[4] The confinement property was called the **-property* in [9]. It was so named as a place holder until a better name could be found. No better name was found prior to publication, so *-property was used, and much of the literature on non-discretionary controls continues to use the name *-property (pronounced *star property*). In 1977, Jerry Saltzer [21] urged that a more meaningful name be found. Thus, some of the literature has since used the term confinement property.

a loyalty card, ensuring that the correct amount of money or loyalty points are transferred may be much more important than keeping secret how much money or how many loyalty points were transferred.

Biba [10] developed a model of mandatory integrity that is a mathematical dual of the Bell and LaPadula mandatory-security model. Biba defines a set of *integrity access classes* that are analogous to security access classes and defines *simple-integrity* and *integrity-confinement properties* that are analogous to the simple-security and confinement properties. The difference between integrity and security is that the direction of the less-than signs are all reversed, so that a program of high integrity is prevented from reading or executing low integrity objects that could be the source of tampering or sabotage. The principal difficulty with the Biba integrity model is that it does not model any practical system. Unlike the security models that developed from existing military security systems, the Biba integrity model developed from a mathematical analysis of the security models. However, Biba did not suggest how to actually decide which programs deserved a high integrity access class and which did not. This has made practical application of the Biba model very difficult.

Lipner developed a commercial integrity model [20] that used both the mandatory security and mandatory integrity models to represent a software development environment in a bank. It tied the integrity modeling closer to reality than the Biba model did, but it was still quite complex and did not provide for how to assign integrity levels to programs, either. To our knowledge, no effort has been made to actually implement the Lipner commercial integrity model.

2.3 IBM Combined Secrecy and Integrity Lattice

How do we actually decide which programs are worthy of a higher integrity level? Since smart card issuers will be particularly worried about the security of applications on their cards (since they might be held liable in a court), we need to improve on the Biba model.

The Biba model also prevents high integrity applications from reading low-integrity data, in fear that the application might be compromised in some form. This makes it difficult to describe applications that have been designed with high integrity to specifically process low integrity data input and to rule on its appropriateness. This processing of low integrity data is called *sanitization*. How do we modify the model to support sanitization (both for integrity and secrecy)?

Our new model solves the problem of assigning integrity levels by using third-party evaluation. Just as for the ActiveX and Java policies, developers digitally sign their applications. However, we go beyond this. If an application has been independently reviewed and digitally signed by the certifying body, then we can grant it a higher level of integrity. For example, we could define integrity levels for ITSEC [5] or Common Criteria [2-4] evaluated applications. The Commercially Licensed Evaluation Facility (CLEF) would evaluate the application and the certifying body would digitally sign the application and its ITSEC E-level. A card issuer (such as a bank) might lay a requirement on vendors who want to download applications onto their cards. Your application must have received at ITSEC evaluation of some level to be acceptable.

The approach we have defined for assigning ITSEC E-levels as integrity levels does not address integrity categories. Biba defined integrity categories, and Lipner proposed use of them in his commercial data integrity model [20]. Interestingly, the Shirley and Schell program integrity model [23] also does not use integrity categories. However, we have not yet identified a use for integrity categories in this new model. We continue to include them for mathematical completeness and because someone may develop a use for integrity categories in the future, but all examples in this paper will only have integrity levels.

To solve the sanitization requirements, we will mark a process with TWO separate integrity access classes – one for read permission and one for write and execute permission. The write/execute integrity access class will be the level determined by the independent evaluator. If the read integrity access class equals the write/execute class, then we have the standard Biba model. If the read integrity access class is lower than the write/execute integrity access class, then we have a process permitted to sanitize and upgrade input that is initially marked at a low integrity access class. Not just any program will be trusted for sanitization, but rather

only programs explicitly evaluated for the purpose. Identifying the range of levels across which a program is allowed to sanitize will be specified as part of the digitally signed information from the CLEF. The operating system kernel will read that information when starting a program into execution to know what range of integrity classes to assign the process. Separating the execute permission from the read permission originated in the program integrity model of Shirley and Schell [23]. The policy was further developed in the GEMSOS security model [22] that specified a range of levels within which integrity downgrading could occur.

We do essentially the same for secrecy sanitization or downgrading. We define a pair of secrecy access classes, rather than integrity levels, and the read and write rules are reversed. There would be two secrecy access classes – one for read/execute and one for write. Note that for secrecy, we keep execute tied to read permission, because a process at a low secrecy level should not be permitted to execute high secrecy program code. This is not for anti-piracy purposes, but rather to maintain the secrecy of algorithms or constants in the code that must not be revealed, even by mere use of the program. Software piracy protection is outside the scope of this security model.

The combined access rules are shown in Figure 1. Note that we separate execute permission into two subclasses – normal transfers and a special CHAIN operation. A normal transfer is the execution of a branch instruction or a subroutine call instruction. CHAIN is a way to start a separate process executing at some other integrity and secrecy access class. Due to the limited memory of a smart card, the process executing the CHAIN operation is immediately terminated. The intended use of CHAIN is to start a guard or sanitization process or for a guard process to start a recipient of sanitized information.

Read permission

Secrecy read/execute access class (process) >= secrecy access class (object)
Integrity read access class (process) <= integrity access class (object)

Write permission

Secrecy write access class (process) <= secrecy access class (object)
Integrity write/execute access class (process) >= integrity access class (object)

Execute permission

Transfer

Secrecy read/execute access class (process) >= secrecy access class (object)
Integrity write/execute access class (process) <= integrity access class (object)

The target program of a transfer runs at the integrity level of the caller. A high integrity program cannot call or transfer to lower integrity code.

Chain

Secrecy read/execute access class (process) >= secrecy access class (object)
Secrecy write access class (process) <=
runtime read/execute secrecy class (new process)
Integrity write/execute access class (process) >=
integrity read access class (new process)

The first rule ensures that chain is possible only to files to which the caller has secrecy read permission. Integrity read permission is NOT required, because a high integrity process is always allowed to start a low integrity process. Contrast this with the rule that a high integrity program is not allowed to transfer directly to a low integrity program in the same process. The second rule ensures that the target process must have secrecy read permission to any passed arguments. The third rule ensures that the target process is not contaminated by a low integrity argument. The target program runs in a new process at the integrity access class specified in the digitally signed certificate of the program. This is a very important distinction. The process that issues the CHAIN operation does not determine the access class at which the target process executes. Rather, it is determined by the third party evaluator and certifying body who have digitally signed the code file.

Figure 1. Access Control Rules

3 Electronic Purse Example

The simplest example is a smart-card based electronic purse. The new security goal is to assure to the merchant terminal that the electronic purse application is not just genuine, but has been evaluated to an appropriate level of assurance under the ITSEC or Common Criteria. Simultaneously, the same kind of assurance can be provided to the purse application on the smart card that the merchant application has been similarly evaluated. Such a scenario is shown in Figure 2.

This example has only slightly stronger requirements than current conventional smart cards can meet. The difference is that the new security model can cryptographically identify the purse application, and also can identify the level of evaluation that has been applied to the application. This is accomplished by using the commercial data integrity part of the model.

The purse application on the smart card must be evaluated under the ITSEC scheme, and digitally signed by the certifying body to have met the requirements of a particular E level. For purposes of this example, let us assume that purses need to be evaluated E4. Whenever the smart card is to be used, mutual authentication must be performed between the card and either the merchant's terminal or some higher-level server application. The digital certificates exchanged during that mutual authentication must include the mandatory access class information about both the smart card application and either the merchant's terminal or the higher-level server, as appropriate. The secure operating systems on both ends must check the mandatory access rights before allowing the communications to proceed. If the merchant terminal is set up to require an E4-evaluated purse, then unless there is an appropriate E4 purse on the card, the communications would not even be allowed to start.

The details of the cryptographic protocols to authenticate mandatory access controls cannot be specified here, due to limitations of space. However, they would be based on standard public-key authentication protocols extended by the MISSI program [7] to include mandatory access control checks. The MISSI protocols only check secrecy access classes, but the commercial version would include integrity access classes. A commercial version would also need support for multi-organizational access classes that are not supported in MISSI. See [15] for more details. These authentication protocols would be built into the smart card operating system and independently evaluated for correctness as part of the ITSEC evaluation of the operating system.

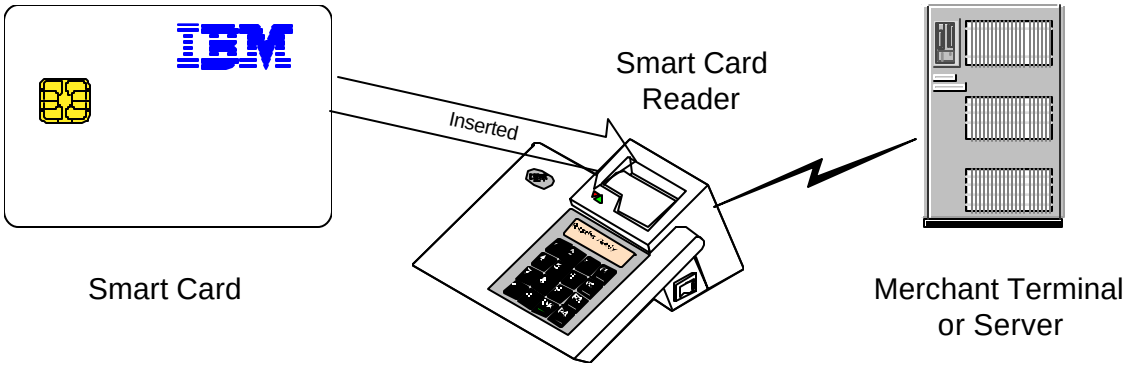

Figure 2. Electronic Purse Example

4 Airline Loyalty Example

This example shows off more of the features of the model. Assume that we have an airline loyalty program that partners with two different car rental companies and two different hotel chains. Assume that those two car rental companies are mutually hostile (and highly unethical) competitors and would be willing even to plant Trojan horses into their competitor's code. This example shows how the new security model can permit each car rental company to download applications to the same smart card.

The two applications can selectively share information with the airline applications on the smart card, yet no information can leak from one car-rental company's application to the other car-rental company's application. Even if the applications of one car rental company contain deliberately planted Trojan horse code, written by a mole in the first company's software development shop who really works for the second company.

The example is significantly different from Lipner's proposals for using non-discretionary controls for commercial applications [20], and we believe it is more easily understood and implemented. Assume we have an airline **A** with ties to hotel chains **H** and **M** and rental car chains **B** and **D**. Staying at the hotel chains earns airline loyalty points. Hotel H gives hotel loyalty points in addition to airline points, while hotel M gives hotel points or airline points, but not both. Hotel H loyalty points and Hotel M loyalty points are completely separate systems. Furthermore, the hotel chains consider the information about where and when the customer has stayed to be valuable marketing information, since the competing hotel chain could use this information to do target marketing. However, the customer and the airline would like all three loyalty systems to be managed from a single smart card, so that the customer need only carry one card, and that card is branded by the airline. Hotel chains H and M do not trust one another, but are both willing to cooperate with the airline A.

Based on these assumptions, the software that manages hotel H loyalty points must behave differently from the software that manages hotel M loyalty points. This is because the hotels have different policies on *double dipping* in which the customer earns points in both hotel and airline plans. Furthermore, the software for both hotel chains and for the airline may need periodic updating to reflect limited time special offers (e.g.: stay five times in one month and earn 500 bonus points), to reflect newly contracted partners, or other significant changes. Another type of special program that would depend on code on the smart card itself would be bonuses if you fly the airline, stay at hotel H, and rent a car from B, all on the same day. Tracking combined bonuses like that could be more easily done on the smart card itself, rather than requiring the central servers for the airline, the hotel, and the rental car companies to all communicate with one another.

We define the following secrecy and integrity levels and categories for the loyalty application:

Integrity Levels: E6 > E5 > E4 > E3 > E2 > E1
Secrecy Levels: System-Low
Secrecy Categories: A, H, M, B, D

There will be data files storing loyalty information for each company. Each company will have a Bell and LaPadula secrecy category. Initially, the customer goes to the airport to fly on airline A. The airline's application with secrecy clearance A will run and grant some airline loyalty points. These are recorded in a file classified A. The airline must also make today's flight information available to all partners. It wants to indicate that the customer has flown today, but it might not want to give full flight details, due to either company confidentiality concerns and/or customer privacy concerns. Therefore, it writes into a different file that is classified system-low that the customer flew today, but with no further details. Now any partner application can read that information. (This assumes that there is only one airline on the card.) The partner applications do not run at this time. They are only invoked when the customer later rents a car or checks into a hotel.

Now the customer rents a car from company B. B's application code runs with secrecy clearance B and computes how many airline loyalty points to grant. It must communicate this information to the airline application, but it does not want the information to be known by rental car company D. Furthermore, B may not want A to know everything about its customer, either. Therefore, B's application writes the number of points earned into a communications file. It then upgrades the classification of that file to require both secrecy category A and secrecy category B. It can do this, because that is a write-up operation from secrecy level B to secrecy level A and B. Next, the B application code must start a B-downgrader process into operation that runs code at a higher integrity level. The B-downgrader will hold a secrecy clearance for both categories A and B. Its program code will be evaluated to a higher ITSEC level. The B-downgrader will inspect the communications file contents and ensure that the data being transferred to the airline does not compromise any of B's confidential information. It then downgrades the information by removing the B secrecy category and passes the information to the airline's software which

runs in another process with only the A secrecy-category.

Figure 3 shows the procedure for communication between rental car company B and airline A through a guard or downgrader process. Ovals are processes, and boxes are data files. Arrows show the direction of information flow, and lightning bolts show Chain operations.

The net result is that only the program that actually inspects the information to be passed from B to A has to undergo a higher-level ITSEC evaluation. The bulk of the applications code that runs on behalf of B and on behalf of A can be either unevaluated or only evaluated to a lower level.

Now let us consider a more complex example. Assume that A, B, and H have created a special bonus program in which if your fly A, rent from B, and stay at H, all on the same day, then you get extra bonus points. To implement this, all three partners must share information, since the flight, car rental, and hotel stay could happen in any order.

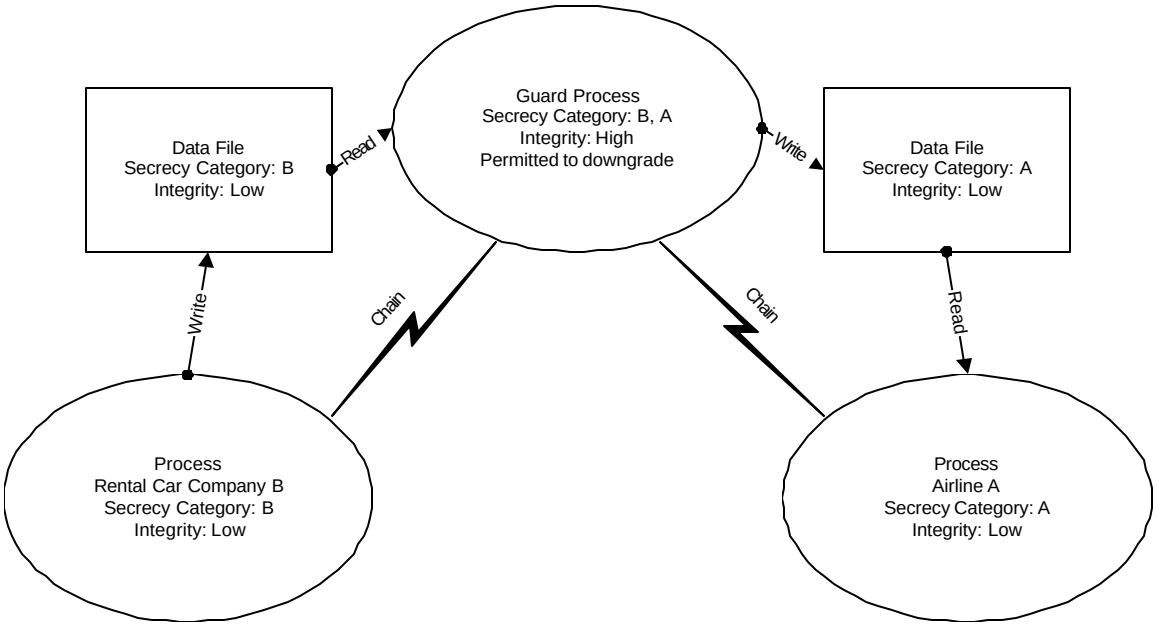

Figure 3. Information flow through a downgrading guard.

Each partner, A, B, and H would need to record in an individual file that a rental, flight, or hotel stay had occurred that day. Each of these files would be marked with all three secrecy categories A, B, and H. Writing these files is a write-up operation and does not require special privileges. They would then each start up a high integrity application that was cleared to read A, B, and H category information. That application would check to see if all three transactions had occurred, and only if all three had occurred, then and only then would the high integrity application downgrade some information to category A only to award the bonus points. Note that most of the code for implementing the bonus operation can be written by each of the partners and only has access to that partners data. Only the downgrader needs access to data from all three partners, and it can be formally evaluated to the higher integrity level to ensure that it does not compromise data.

5 Cell-Phone - PDA for Military Messaging Example

The most complex example will show how the model could be used in a cell-phone-based PDA to be used by a military service. Imagine a cell phone messaging service issued to a military officer. That messaging service should allow the officer to send and receive unclassified messages from his or her family, confidential messages concerning routine military operations, and top secret messages alerting the officer to immediate war alerts. The messaging software might come from an Internet-based company that views its source code as highly proprietary. Therefore, the code of the messaging system cannot be checked

for security. Furthermore, the messages might be permitted to contain embedded active content (such as the macros that can be embedded in word processing packages, such as Microsoft® [5] Word or Lotus® Word Pro™ [6]). Such embedded active content could easily contain viruses or Trojan horses, as could the messaging software itself. The conventional approach is to ban such active content in high security environments. The paper will show how the new security model can permit such active content to execute, yet still ensure that the top secret messages aren't compromised.

There are several possible design approaches for this messaging scenario. Which is best would depend on the details of the messaging software itself. The first approach (shown in Figure 4) would be to maintain several instantiations of the messaging software, one for each secrecy access class involved. The messaging software itself would be stored as a shared read-only program file in ROM or EEPROM, mapped into the address space of each instantiation. There would be three databases of messages, one at each of the three access classes – unclassified, confidential, and top secret. When an incoming message arrives, a trusted demultiplexer would first mutually authenticate the sender and determine the appropriate access class. It would then start a process at that access class, running the read-only code of the messaging system. The process would only have write access to the database at the proper access class, but it could also have read access to the databases at lower access classes. This approach minimizes the amount of trusted, evaluated code required, but when the user is reading a confidential message, he or she cannot have access to top secret messages, even to know whether any such messages exist. This approach of a totally untrusted messaging system was first developed for a US Air Force messaging system [1]. That approach ultimately evolved into the SACDIN network [13], the first high assurance messaging system ever developed.

The second approach (shown in Figure 5) would be to store the messages in a single protected database that was inaccessible to the messaging software. A trusted demultiplexer is still required, but all the messages can be stored in a single database. Access to the database would have to be mediated through a high-integrity program that was evaluated to a sufficiently high ITSEC E-level to adequately separate the messages of different classifications. Such an approach was taken in the design of message segments to support multi-level electronic mail in the Multics system [25], although the message segments were implemented as part of the operating system, rather than as a separate high-integrity application. However, the message segments were implemented in ring 1, rather than in the most privileged ring 0, thus making use of the Shirley and Schell [23] integrity model.

Both the first and second approaches retain much of the messaging system software as unevaluated code, which will significantly reduce the cost of software development. The third approach (shown in Figure 6) would be to fully evaluate the entire messaging system software to a high ITSEC E-level. Then, the de-multiplexing, message processing, and message storage could all be done in a single monolithic system. The principal drawback is that this approach requires all code undergo a high level ITSEC evaluation, which would result in the highest development and evaluation costs. Furthermore, this would require either that the macro-extension language be shown totally secure, or require avoiding the use of such macros entirely. The history of security flaws in such macro languages makes it unlikely that such an extension language could pass evaluation easily. The only interpreted language to successfully pass a high-level ITSEC evaluation is MEL, the MULTOS Executable Language, and that evaluation put extreme constraints on the language, such as no sharing of data between applications. MEL is also a much more constrained language than Visual Basic or LotusScript.

6 Conclusions

We have developed a new model for mandatory access controls that solves a number of problems in the existing Bell and LaPadula model for secrecy and the Biba model for integrity. We have shown several examples of how this model could be used in smart card and PDA applications. Will this new model actually prove successful in the commercial world? It is much too early to

[5] Microsoft is a trademark of Microsoft Corporation in the United States, other countries, or both.
[6] Lotus and Word Pro are trademarks of Lotus Development Corporation in the United States, other countries or both.

tell. None of the previous commercial data integrity models have truly succeeded in the commercial world, so it would be presumptuous to claim that this new model will succeed where previous ones have not. We will not know whether the new model proposed in this paper will be commercially successful until the model has been exposed both to security experts and to real commercial developers. This paper is a step in that process.

7 Acknowledgements

We must thank a number of people for their contributions to the development of this model. At IBM, our work has benefited from the comments and suggestions of Elaine Palmer (the project manager), Jonathan Edwards, Helmut Scherzer, Klaus Gungl, James Riordan, and Günter Karjoth. We must also thank Helmut Kurth of iABG, Stefan Wittmann of BSI, Gerhard Schellhorn and Wolfgang Reif of the University of Ulm, and Axel Schairer of the DFKI for their inputs. Finally, we must thank the anonymous referees of our earlier paper for pointing out the relationship of this work to the GEMSOS system.

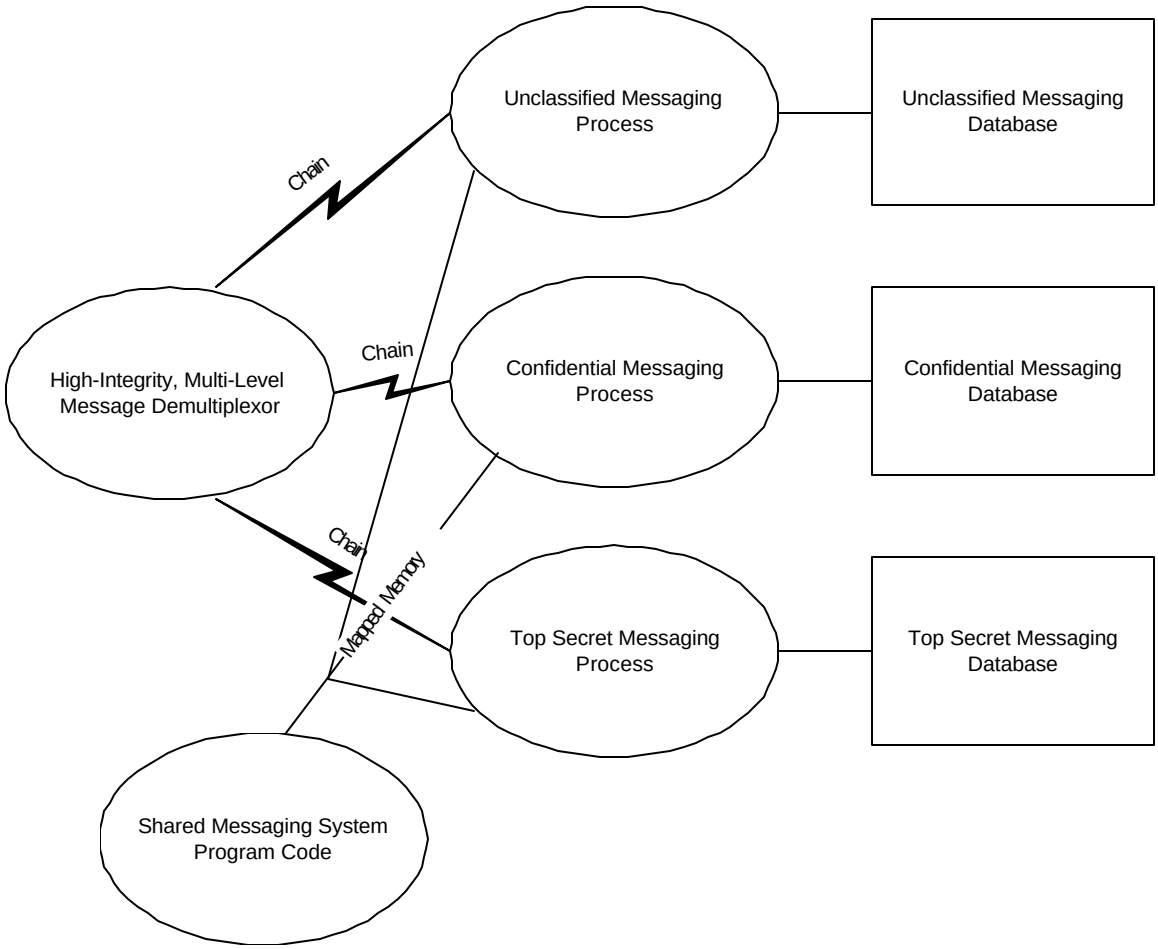

Figure 4. Multiple-Instantiation Messaging Approach

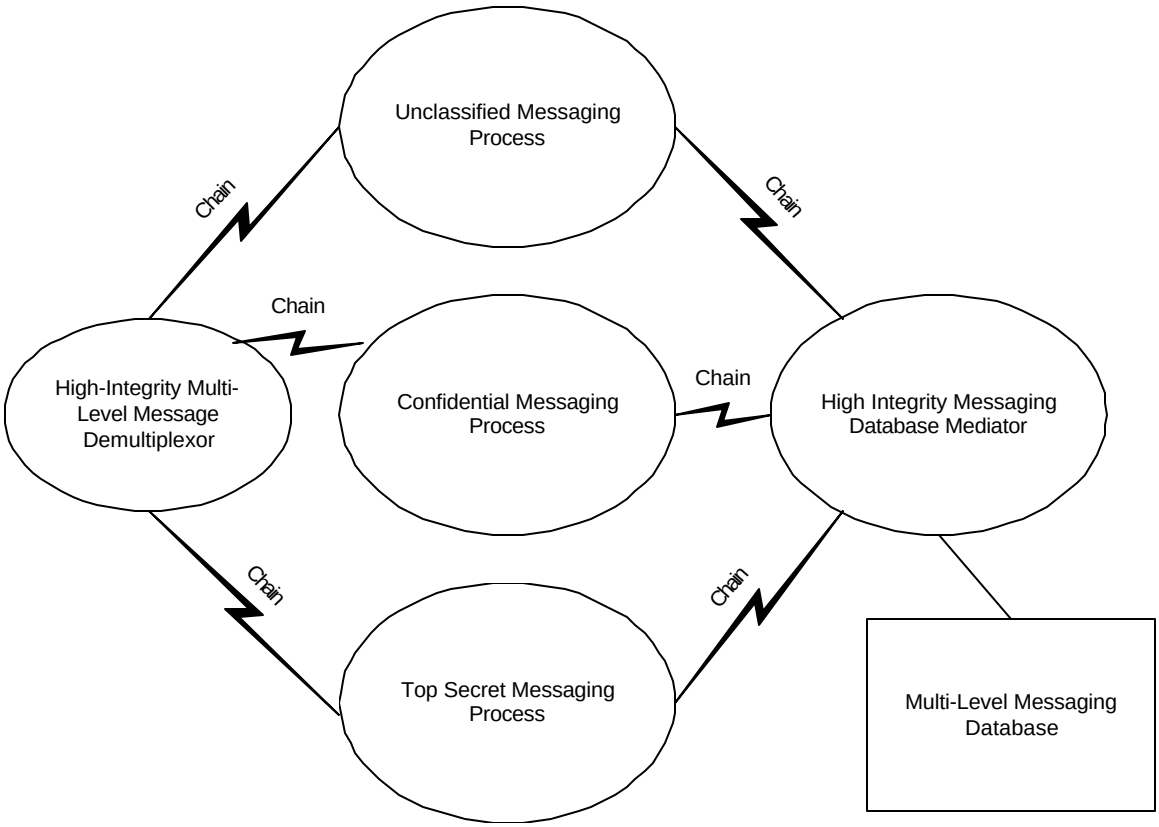

Figure 5. Multi-Level Database Mediator Approach

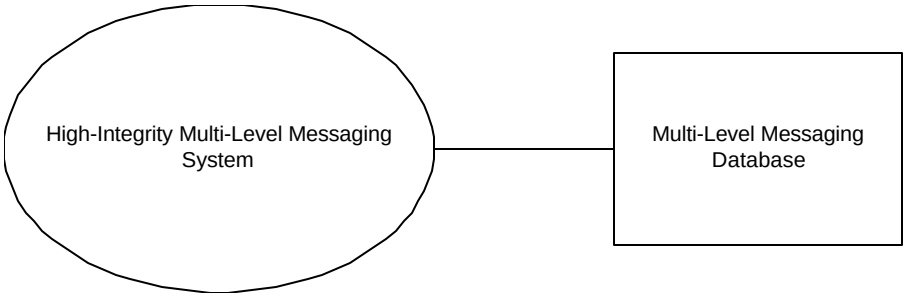

Figure 6. Monolithic Messaging Approach

References

1. *The Feasibility of a Secure Communications Executive for a Communications System,* MCI-75-10, August 1974, Electronic Systems Division: Hanscom AFB, MA.

2. *Information technology - Security techniques -- Evaluation criteria for IT security -- Part 1: Introduction and general model*, ISO/IEC 15408-1, 1999, International Standards Organization.

3. *Information technology - Security techniques -- Evaluation criteria for IT security -- Part 2: Security functional requirements*, ISO/IEC

15408-2, 1999, International Standards Organization.

4. *Information technology - Security techniques -- Evaluation criteria for IT security -- Part 3: Security assurance requirements*, ISO/IEC 15408-3, 1999, International Standards Organization.

5. *Information Technology Security Evaluation Criteria (ITSEC)*, June 1991, Commission of the European Communities: Brussels, Belgium.

6. *Philips Semiconductors and IBM Research to co-develop secure smart cards: Highly secure operating system and processor, suitable for multiple applications*, 4 February 1999. URL: http://www.semiconductors.philips.com/news/content/file_384.html

7. *SDN.801: MISSI Access Control Concept and Mechanisms*, MCCB-04.02.029, ON636216, Revision C, 12 May 1999, National Security Agency: Ft. Meade, MD. URL: http://www.armadillo.huntsville.al.us/Fortezza_docs/sdn801c.pdf

8. Bell, D.E. and L.J. LaPadula, *Computer Security Model: Unified Exposition and Multics Interpretation*, ESD-TR-75-306, June 1975, The MITRE Corporation, Bedford, MA: HQ Electronic Systems Division, Hanscom AFB, MA.

9. Bell, D.E. and L.J. LaPadula, *Secure Computer Systems: A Mathematical Model*, ESD-TR-73-278, Vol. II, November 1973, The MITRE Corporation, Bedford, MA: HQ Electronic Systems Division, Hanscom AFB, MA.

10. Biba, K.J., *Integrity Considerations for Secure Computer Systems*, ESD-TR-76-732, April 1977, The MITRE Corporation, Bedford, MA: HQ Electronic Systems Division, Hanscom AFB, MA.

11. Denning, D.E., *Cryptography and Data Security*. 1982, Reading, MA: Addison-Wesley.

12. Denning, D.E., *A lattice model of secure information flow*. **Communications of the ACM**, 1976. **19**(5): p. 236-243.

13. Ferdman, M. *SAC Digital Network (SACDIN) Security Methodology*. in **Proceedings of the Fourth Seminar on the DoD Computer Security Initiative**. 10-12 August 1981. Gaithersburg, MD: National Bureau of Standards. p. G-1 - G-9.

14. Girard, P. *Which Security Policy for Multiapplication Smart Cards?* in **Proceedings of the USENIX Workshop on Smartcard Technology**. 10-11 May 1999. Chicago, IL: The USENIX Association. p. 21-28.

15. Karger, P.A., *Multi-Organizational Mandatory Access Controls for Commercial Applications*, RC 21673 (97655), 22 February 2000, IBM Research Division, Thomas J. Watson Research Center: Yorktown Heights, NY. URL: http://domino.watson.ibm.com/library/CyberDig.nsf/home

16. Karger, P.A., V.R. Austel, and D.C. Toll, *A New Mandatory Security Policy Combining Secrecy and Integrity*, RC 21717 (97406), 15 March 2000, IBM Research Division, Thomas J. Watson Research Center: Yorktown Heights, NY. URL: http://domino.watson.ibm.com/library/CyberDig.nsf/home

17. Lampson, B.W., *A note on the confinement problem*. **Communications of the ACM**, 1973. **16**(10): p. 613-615.

18. Landwehr, C.E., *Formal Models for Computer Security*. **Communications of the ACM**, 1981. **13**(3): p. 247-278.

19. Lipner, S.B., *A comment on the confinement problem*. **Operating Systems Review**, 1975. **9**(5): p. 192-196. Proceedings of the Fifth Symposium on Operating Systems Principles.

20. Lipner, S.B. *Non-Discretionary Controls for Commercial Applications*. in **Proceedings of the 1982 Symposium on Security and Privacy**. 26-28 April 1982. Oakland, CA: IEEE Computer Society. p. 2-10.

21. Saltzer, J., *Personal communication on the name of the *-property*, 1977, Massachusetts Institute of Technology.

22. Schell, R.R., T.F. Tao, and M. Heckman. *Designing the GEMSOS Security Kernel for Security and Performance*. in **Proceedings of the 8th National Computer Security Conference**. 30 September - 3 October 1985. Gaithersburg, MD: DoD Computer Security Center and National Bureau of Standards. p. 108-119.

23. Shirley, L.J. and R.R. Schell. *Mechanism Sufficiency Validation by Assignment*. in **Proceedings of the 1981 Symposium on Security and Privacy**. 27-29 April 1981. Oakland, CA: IEEE Computer Society. p. 26-32.

24. Walter, K.G., W.F. Ogden, W.C. Rounds, F.T. Bradshaw, S.R. Ames, and D.G. Shumway, *Primitive Models for Computer Security*, ESD-TR-74-117, 23 January 1974, Case Western Reserve University, Cleveland, OH: HQ Electronic Systems Division, Hanscom AFB, MA.

25. Whitmore, J., A. Bensoussan, P. Green, D. Hunt, A. Kobziar, and J. Stern, *Design for Multics Security Enhancements*, ESD-TR-74-176, December 1973, Honeywell Information Systems, Inc., HQ Electronic Systems Division: Hanscom AFB, MA. URL: http://csrc.nist.gov/publications/history/whit74.pdf