Quality in Object Based Schemas

Richard Wagner

rwagner@usc.edu

or

76427.2611@compuserve.com

O: 5/5 C: 5/5

Abstract

The idea that the quality of an object based schema is a function of its utility is developed in this paper. Then based on the defined uses of the schema, and with the aim of maximizing schema quality, the extension of computer aided software engineering (CASE) modeling tools to three dimensions is proposed.

HIPPIAS: Oh, well, Socrates, if *that's* what he's after, there's no difficulty at all in telling him what fineness is...

Plato [2]

Introduction

The quality of a database schema can be judged by its suitability for the use to which it is put. As an extreme example of this, consider a schema for a relational database that is not in any normal form at all: It is completely useless for the purpose of providing correct information over a varied range of queries. Further, the entity-relationship (ER) model is useful as a design tool[1] to aid in the development of relational schemas: those relational schemas derived from the ER model will be in third normal form (3NF) [3], and will therefore be not only useful for a variety of data storage and retrieval tasks, but will provide flexibility for future schema growth. Hence, an important aspect of schema quality is utility.

To examine the subject of the utility of an object based schema, we need to address the uses to which the schema will be put. These uses, in turn, can best be examined in the context of the problems facing a database designer (DBD) when he sets out to devise a system for a complex application environment. Now that almost all information in significant business enterprises is automated, and that various units of these enterprises are joining in computer networks to share their data, the "real world" application environments today are complicated indeed. In short, the database designer needs all the help he can get.

[1] A conceptual schema developed according to the ER model can be considered an adjunct for design in this application.

HIPPIAS: I'll tell you. What you're after, I think, for your answer, is fineness which is such that it will never under any circumstances seem contemptible to anyone. [2]

The Utility of Schemas

An object based schema can be considered to be useful in two different ways. The first use is in providing a description (in graphical form) of an abstraction of the application environment (mini-world). The second is in aiding the database designer (or team of designers) in the implementation process of mapping the abstraction (of the mini-world) to the database management system (DBMS). These two uses can be considered to apply to both the creation of a new database system and the modification (evolution) of an existing one.

For the purpose of this paper, I assume that the DBMS being used by the DBD was chosen for its ability to easily map an object based schema to a DBMS implementation (data model mapping [4]), and focus on the design aspects of schema use.

Design

The design of a nontrivial database system is inherently iterative, i.e., nobody gets it right the first time. In addition, no significant mini-world is static. Requirements evolve during the initial design phase. Database design in challenging applications therefore is never "completed." To be relevant in today's world, a useful schema must be supportive of change in all phases of database design and implementation.

Good database design comes (in part) from knowledge of the mini-world. Today's application environments are so complex that very few people can be intimately familiar with all relevant aspects of them, particularly those who specialize in database systems. Therefore, the DBD is dependent upon the system users for his knowledge of the mini-world. The DBD's primary tool for communicating with the users is his schema. Hence, a good schema is one that is useful for communicating with users.

SOCRATES: But we mustn't give up the chase yet, my friend. Uncovering fineness isn't quite hopeless yet.
HIPPIAS: Of course not, Socrates. It's really quite easy. I'm sure that after I'd thought it over for a bit, in peace and quiet, I could give you the most precise of precise definitions. [2]

Communication

Because users therefore will interface with the schema in all phases of the DBMS design and implementation, we should consider what aspects of a schema will make it most useful in that regard. That is, what things will make one schema better than another for communicating with users? First, the naive users will need to understand the schema. This means that of two schemas that effectively capture the same level of abstraction of the mini-world, the simpler one will be better. A corollary of this is the strategy of "confusion avoidance" (CA) which is generally applicable to all things that have anything to do with computers. In addition to simplicity, CA is also achieved with orderly presentation and the eschewing of cryptic notation.

Simplicity

The recent database design for the National Gallery of Art [1] is an example of the application of simplicity to the use of a schema:

> The design process used a Bachman style diagram to describe the database entities and relationships. We found this simpler than the Entity Relationship Model (ERM). Sometimes ERM users get hung up on details, such as classifying entity types and relationship types, that do not bear on the final design. Bypassing the ERM saved time, and the simpler model was more easily understood by both technical and non technical members of the design team. [1]

In addition to providing a rich mini-world abstraction capability, the Generic Object Based Database Model (GOBDBM) also features a relatively simple and non cryptic schema notation, making the GOBDBM very attractive for challenging database applications. Simplicity, while an important consideration in the quality of a schema, is not the only factor. Because the ability of users to understand the schema also depends on their total perception of, we also need to consider some sensual (aesthetic) aspects of schema.

SOCRATES: The same goes for the whole body: we call it 'fine for running' or 'for wrestling'. Or again, take any living creature--a fine horse, cock or quail; take any artifact; take any vehicles for land and boats and ships on the sea; yes, and take any instrument, whether it is musical or used in some other skill; and, if you like take practices and laws--all of these we invariably call 'fine' if it is useful in some way or for some purpose or at some time; but if it is useless in all these respects, we call it 'contemptible'. Don't you agree Hippias? [2]

Aesthetics

In the iterative design process, a wide range of users will be called upon to interact with the DBD and help to develop requirements and to comment on the adequacy of interim designs. For instance, people from the budget office, from engineering, from sales, or from manufacturing might be involved on an occasional or regular basis. The way people perceive and understand the schema will be affected by aesthetic considerations.

Take for example one set of design tools that produces hard copy of a schema by printing ASCII characters on multiple sheets of paper (boxology). Perhaps the DBD will pin the sheets up on the wall in some connected order for communicating with the users. Now consider a similar set of design tools, but which has the capability of rendering a graphic output to a J-size (standard aerospace mechanical design) plotter. Obviously the system with the finer output capability will produce the better schema graphic.

Included in the category of aesthetics are some notions from the world of art. Concepts like balance, proportion, line, color, grouping, and the like will affect the way people relate to the schema. Defining criteria for these subjective things is difficult, but some obvious principles can be stated. For example, classes that are more centrally involved in the mini-world should be more centrally located in the schema. There should be some effort to minimize the number of crossed lines (see CA, above), and the use of white space can be effective in denoting implied groupings.

HIPPIAS: Listen Socrates. This is all picking and whittling at words, as I said before--just splitting hairs. What else do you suppose it is?[2] [2]

The Third Dimension

While a J-size drawing may be adequate for displaying the schema of an ambitious database in today's world, the limits of a two dimensional (2D) drawing surface may by breached in the next century. Chemists describe molecules on paper when they can, but for very complex structures they resort to solid models, either physical, or lately, using three dimensional (3D) design software. Architects use 3D solids software to view their buildings from any perspective, and with "virtual reality" equipment, can join their clients for cyberspace journeys through their edifices. A schema that can be viewed and manipulated in cyberspace would be more useful (and therefore be of higher quality) than one that can only be viewed on screen or paper.

The GOBDBM in Cyberspace

The GOBDBM has a fairly simple schema notation, consisting primarily of ellipses (for classes) connected with directed lines (for relationships), with provisions for subclasses, grouping classes, and methods. Extending this notation to 3D is straightforward: a class would be shown as an oblate sphere, with a label in front of it for its name. The label would be "aware" of the user's viewpoint, and always present itself on the side of the class that the viewer is on. An analogous scheme has been implemented in the game *Cyberspace Crossword*[3], in which the player enters letters in a 3D wire frame puzzle structure. As the player moves through cyberspace to change his viewpoint, the puzzle letters rotate about a vertical axis so as to always present their front sides.

---

[2]Socrates and Hippias never do find out what fineness itself is although both agree that one knows a fine thing when he sees it. That may not be the case with an object based schema, where the proof is in the process. A fine schema will be praised by all who use it, while a lesser schema will be found contemptible.

[3]*Cyberspace Crossword* from Ivory Tower Software is a game for the Microsoft Windows graphical environment.

Arrows will have to have some thickness so as to show perspective, with arrows denoting subclasses being especially thick. A 3D arrow head is rendered as a cone. Color could be used to good advantage too. For instance, objects of interest to one group of users could be of a similar color.

To be maximally useful, the cyberspace schema environment should also provide effortless maneuvering so that the user can change his vantage point as easily and rapidly as possible. He should be able to zoom backwards to grasp the whole structure, jump to remote points by simply indicating them, and move in and around volumes of interest with intuitive ease. Permanently modifying the schema should be done only by the DBD, but if a user wanted to move a group of objects, or to add or change connections, he should be able to do that so as to demonstrate his intent to the DBD. To enhance communication among the viewers, they should have knowledge of their respective virtual locations by the display of appropriate body representations[4].

Conclusion

The more useful a schema, the better it is. Schemas that are readily grasped by users are more useful than unnecessarily complicated ones. Other aesthetic qualities also affect the way a schema assists the DBD in communicating with the users. Moving the schema paradigm into cyberspace can improve accessibility and may be necessary for ambitious future applications.

---

[4]In the *Cyberspace Crossword* puzzle, other users are shown as different colored spheres. A more complex solid which represents a body with head, arms, and legs, would be better because it gives a better impression of relative size.

References

1. Brand, E. G. and Rerritsen, R., "Database Design for the National Gallery of Art," *DBMS* magazine, Volume 6 Number 3, March 1993.

2. Plato, "Hippias Major," *Early Socratic Dialogs*, beginning on p. 231, Penguin Books, 1987.

3. Elmasri, R., and Navathe, S., *Fundamentals of Database Systems*, p. 376, The Benjamin/Cummings Publishing Company, Inc., 1989.

4. Elmasri, R., and Navathe, S., *Fundamentals of Database Systems*, p. 39, The Benjamin/Cummings Publishing Company, Inc., 1989.

Security in a Spacecraft Assembly and Test Information System

Richard J. Wagner

76427.2611@compuserve.com

December 1, 1992

Abstract

Efforts are underway by contractors to implement distributed information systems for spacecraft programs. Protection of data is important for the unclassified programs as well as for the restricted access programs that have special requirements for secrecy. This paper proposes a security model to provide a baseline protection level for all programs and extends the concept to meet customer requirements for classified programs. The proposed security model is based on encrypted authentication services with smartcard-maintained client process authorization to foil both intruder programs and evil administrators.

1 Background

Various degrees of automation of information preparation, storage, and access have been implemented for the several disciplines involved in spacecraft design and production. Two-dimensional computer aided design (CAD) is standard, and most new programs are moving to three dimensional solid modeling systems for spacecraft mechanical design. Word processing technology has been in routine use for many years for the preparation of interoffice correspondence, requirements documents, proposals, reports, and so on. Program business administrators were among the first to adopt personal computers (PCs) in their work, using spreadsheet application software for cost estimating and tracking. Many project managers now use PC project planning and scheduling software to help them manage their areas of responsibility.

Many of these isolated systems are becoming connected to high performance (10+ Mb/s) local area networks (LANs). One LAN system in a classified facility allows automated assembly procedure preparation and execution, with database and file servers maintaining the spacecraft as-built configuration and history.

1.1 Contemporary Examples

LANs in unclassified areas of many aerospace companies are being connected into internetworks. For instance, TRW has a rather extensive internetwork called TIENET (see figure 1), which is itself connected to the Internet. Security in this system is based primarily on physical security. The facility buildings require ID picture badges for entry, inter-building cabling is routed underground and manhole covers are locked. The Space

Park LAN routers limit access to services so that only machines connected to Space Park LANs may download information from Space Park servers.

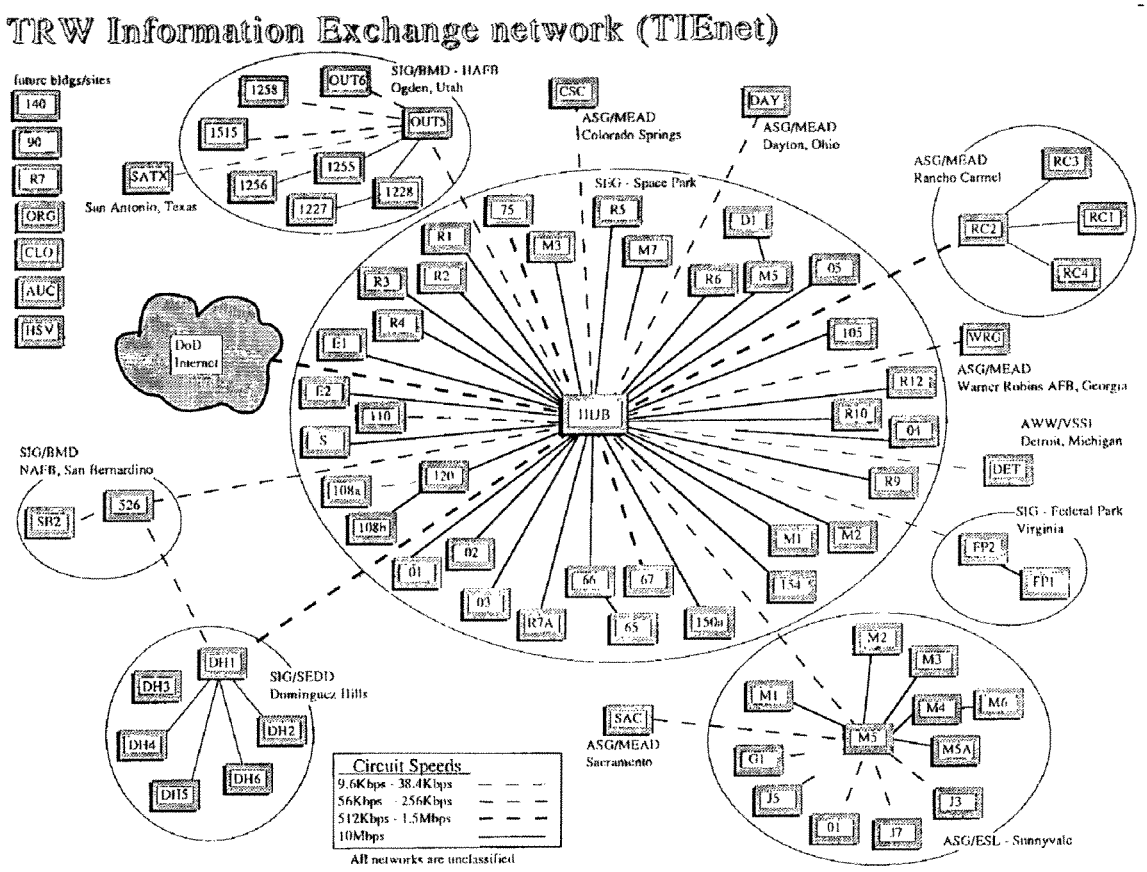


Figure 1

An example of an internetwork supporting a classified project is shown in figure 2. In this system, LANs are contained within secure facilities, with an encrypted link between the main facility and the remote site. Classified projects deal with information related to national security, and as such must satisfy stringent requirements imposed by customer communities. These additional requirements include limits on radio emission levels (which are controlled by screening facility buildings and suppressing signals at their sources) and restrictions on data communication across facility boundaries.

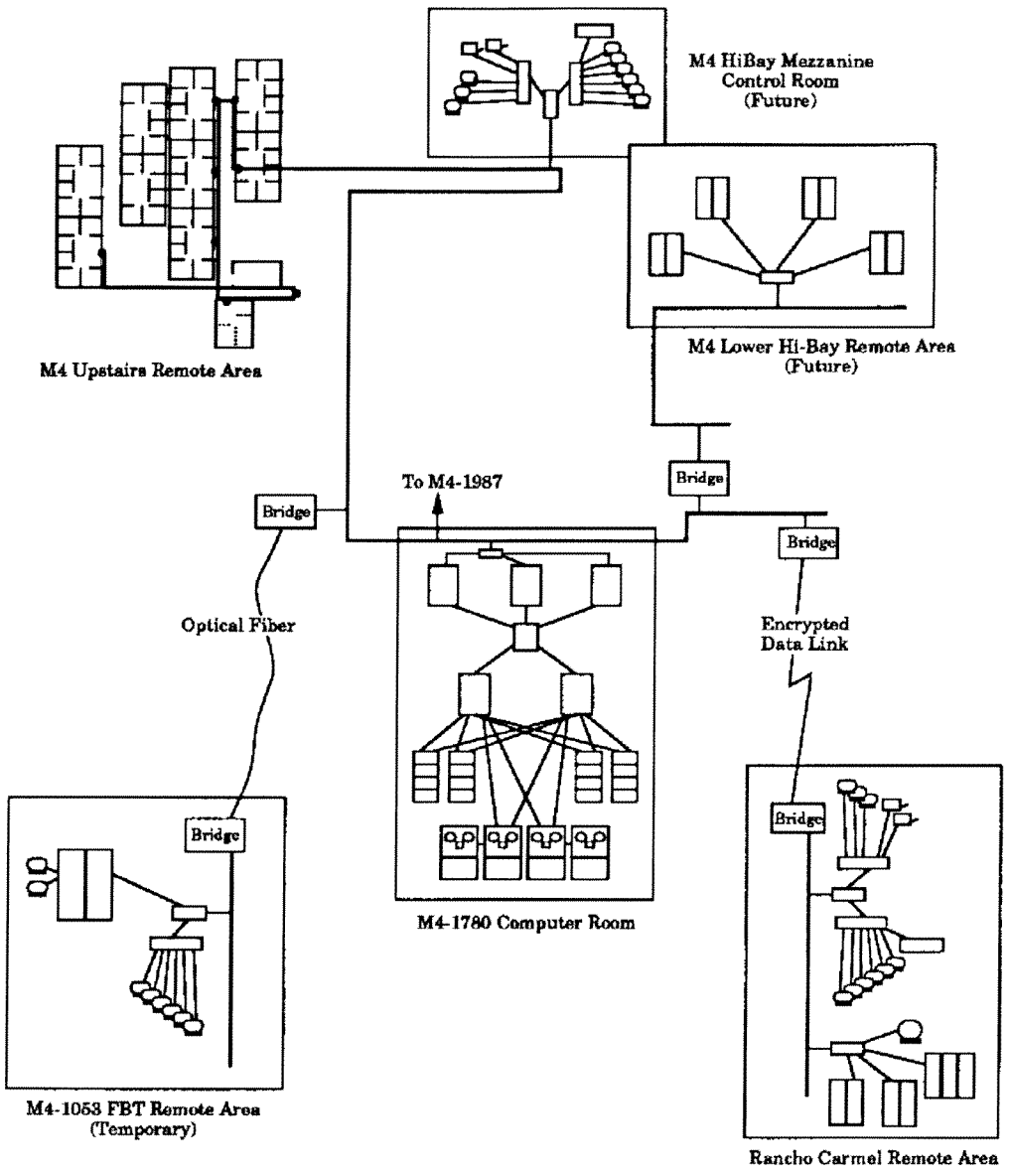


Figure 2

1.2 Future Systems

Recently, efforts have been directed toward increasing worker access to information by improving system compatibility and implementing protocol-level coherence. System requirements and a preliminary plan to implement such a system are described in [15], which attempts to transparently link the organizations that exchange data with spacecraft integration and test operations. This approach can be generalized to include all aspects of

a spacecraft program, including ground systems projects, from proposal phases through system operations and support. Bringing all the isolated automation efforts together into a unified system holds the promise of greater productivity, but it also raises some serious security questions.

As the scale of a system increases, the difficulties of implementing effective protection increase as do the unhappy consequences of security compromise. The secrecy requirements of classified projects are governed by customer regulations [3], which are very effectively implemented at the present time. Ironically, one of the primary means of achieving adequate protection on classified programs is by limiting the scale of these systems: distributed interconnection to systems not within the same project compartment is not allowed [4]. A description of how to allow limited connections to open system services without compromising secrecy is presented later in this paper. Constructing large distributed systems for the unclassified programs also presents some interesting security challenges, particularly as scale increases to incorporate all disciplines into a unified system.

2 Security Threats

The security threats to unclassified programs can be considered to be a subset of the threats to the classified programs. That is, unclassified programs have a similar need for protection as do the classified programs, except that the secrecy requirements of the classified programs are absolute in that they are not subject to negotiation. While some sources in the literature define specific classes of security threats in great detail [9][14], for the purposes of this paper, I broadly lump security threats into two categories: espionage and sabotage. This separation of sensitivity (protection of classified or proprietary information) from criticality (integrity of operations and protection against denial of services) is consistent with approach taken in reference [6].

The privacy facet of security is also important. Espionage threatens privacy as well as secrecy, but my emphasis here will be aimed at protecting the privacy of individual users from the system itself.

2.1 Espionage

Competition among spacecraft vendors is intense. Access to information regarding the proposal activities of competitors can give a company an unfair advantage in winning

contracts. Imagine, for instance, some shadowy third party building a collection of all the system drawings, specifications, and plans for a proposed new generation of unclassified satellites, which he then offers to share with some other company that is bidding on the same satellite project. While there are ethical prohibitions against using improperly obtained information, basing a security plan on the assumed good will of potential adversaries is not a prudent approach. There are criminal sanctions against some forms of wire tapping and industrial espionage, and certainly recovery of damages in a civil court is possible after the fact. However, the potential involvement of third parties can help those who would benefit from unscrupulous access to disassociate themselves from culpability, and may increase their temptation to profit from such dishonest practices.

2.2 Intruder Programs

While the Space Park portion of TIENET is physically secure from outside attack, it is possible for an outside agent to pay an otherwise honest employee to load certain programs on his personal computer at work. The agent would not have to tell the employee what the program is or what it does. He could simply (find someone who might be about to be lain off and) say "I'll give you $5000 to put this floppy disk in your machine and run the installation program." The intruder program suite (IPS) could set itself up to run in the background every time the machine was turned on after that. The IPS could do a variety of things such as listen to the LAN and collect passwords, browse a particular CAD drawing server and download all recently changed drawings, relay downloaded files to an outside site via modem, and give its owners periodic status reports and data dumps via the Internet. This scenario illustrates the need for stronger security measures in the future.

2.3 Sabotage

To suspect a legitimate business enterprise of desiring to sabotage a competitor's operations might border on the paranoid, but sabotage is certainly a possibility for a terrorist group or for a lone sociopathic cracker. Besides physical sabotage, attacking the Space Park systems could follow the pattern outlined above for the espionage case. Because the financial incentives may be less for sabotage, and the protection approaches are similar to those for protection against espionage, security provisions that are effective against espionage can be considered to be similarly effective against terrorist, cracker, or disgruntled employee non-physical sabotage.

2.4 Privacy

Apart from the sensitivity and criticality issues as related to corporate data that were discussed above is the threat to the individual user's privacy. This threat comes primarily from the authorized administrators, who are generally trustworthy, but often have no need to know private information. For example, a department manager may be reluctant to keep his subordinate's performance appraisals on a file server if he believes that an administrator has access to them.

The potential problem is that if users have no sense of privacy for their data, they will find ways to circumvent the system. This is analogous to the trend that developed in the early 1980s of managers bringing in personal computers to keep their databases and spreadsheets. The motivation for the PC trend was efficiency (in the face of non responsiveness of computing centers), however, rather than privacy. It is conceivable that future managers might string their own network cabling and set up their own isolated LANs[1] to bypass what could come to be viewed as an unacceptable invasion of privacy.

3 Proposed Baseline System Security Model

The attack by the IPS described in the threat scenario above succeeds because the TIENET servers rely on simple password authentication. Once the user logs onto the Banyan Vines[2] system, all the electronic data management (EDM) services are available to his machine. This log-on sequence can be remembered by the IPS and replayed. Even if the human user changes and all the passwords are changed, the IPS can listen and replay. Eradicating an IPS can be very difficult if it uses replication to enhance its own self-availability.

---

[1]Setting up an isolated LAN to provide something (privacy) that is not available through the system is analogous to purchasing standalone PCs to provide something (computing service) which is not available normally.

[2]Noted for its internetworking capability, Banyan Vines is the Space Park network operating system on Unix hosts.

3.1 Authentication and Encryption

Because the PCs are subject to modification by their users (i.e., installation of an IPS), they can be said to be inherently untrusted. However, an IPS can be foiled if all critical network services were to require encrypted authentication and authorization as described in the referenced papers on the Kerberos authentication service [13] and authorization by restricted proxies [11]. In view of the untrusted nature of the user PC, the authentication services [5, 10] must be augmented by "smartcard" [8] user log-in that hides the initial password from the PC.[3]

3.2 Use of Smartcards

This arrangement is secure (from outsiders) because the smartcard is always in the authorized user's possession (and the encryption is done by the smartcard), and secure from insider-assisted attack because an IPS resident on the PC would have no means to engage the services of universally suspicious network resource providers. When the user quits the session and takes his smartcard with him, the IPS remains helpless. When a different user inserts his smartcard and logs on, the IPS still sees nothing but incomprehensible (to the IPS) encrypted messages.

3.3 Smartcard Table of Authorized Programs

To allow legitimate programs to access services, the smartcard must contain a list of programs authorized for network access that includes their sizes in bytes and checksums.[4] In doing this, the smartcard is appropriating one of the conventional protection roles of the operating system kernel. The essential feature of this scheme is that the authentication server will give tickets only to the smartcard on the user's behalf, not to the kernel or the client program. The smartcard then passes tickets or restricted proxies to requesting client

---

[3]The message the smart card passes through the PC to the authentication server is encrypted with a private key. The smartcard has a small keypad for the user to enter his personal identification number. The PC must have a socket to accept the smartcard.

[4]This proposed model could work just as well (protecting corporate data) if each user's table of authorized programs were kept by his local authentication server. However, keeping the tables encrypted on the smartcards offers the user protection against administrative tampering with his local privacy encryption server. See section 3.4 Privacy.

programs as warranted. Symmetric key encryption protects all ticket passing. A sampling of smartcard protocols is given in Appendix A.

Even if an insider collaborates with an attacker by handing over his smartcard to him for reverse engineering and tampering, compromise could not occur because the attacker could not change the smartcard's table of authorizations because the table would itself be encrypted with a private key that the smartcard doesn't know. In the initial log-in protocol, the smartcard sends the encrypted table of authorized PC programs to the authentication server for decryption. The authentication server then encrypts the table in a session key and returns the table to the smartcard for decryption[5] and storage in the smartcard's volatile memory.[6] Only then would the smartcard be ready to enable a PC program to request services on the user's behalf.

3.4 Performance

While this proposed model using authentication servers and smartcards may seem cumbersome in that it will increase network traffic, add encryption-decryption overhead to many service requests, and require all sensitive corporate network service access to take

---

[5]Several manufacturers produce chip-sets for smartcard production to the international standard ISO 7816. While current non-volatile memory capacity is adequate (NEC cards have up to 2M EPROM), the processors tend to be rather low powered (Motorola 6805 8-bit processor, for instance). Future smartcard processing power will have to improve to support these encryption processes. There is a relatively new standard (PCMCIA [1]) for a credit card sized device with a 68 pin powered interface. While current PCMCIA devices are geared toward replacing floppy disks with solid state memory, this standard interface is an alternative to ISO 7816, and might be easily adapted to house a powerful CPU and micro-keyboard. PCMCIA interfaces are becoming available in portable computers, so it is possible that this interface may become standard equipment with desktop systems in the near future. Already there are credit card sized fax modems and ethernet LAN adapters for portable computers in addition to the memory devices.

[6]The smartcard would need no battery. The authentication server's public key, the smartcard's private shared keys, and encrypted program authorization table would be written into EPROM in the smartcard. Inserting the smartcard into the host PC to initiate a session would allow the smartcard to tap DC power for volatile DRAM storage and for other active component operation. Removing the smartcard from the host PC would cause any decrypted data stored in the smartcard to be destroyed.

place via smartcard-capable PC hardware, it provides air-tight security in a world in which attacker incentives will most likely increase. As computer power and network bandwidth continue to increase, the percentage of resources absorbed by these security safeguards will likely decrease.

It should be noted that the authentication and encryption protocols apply only for access to sensitive corporate network services. Ordinary access to Internet services would not be affected.

3.5 Privacy

On the surface, the requirement for smartcard access to the network would seem to preclude at least one aspect of privacy. After all, a user could not just bring in his own software[7] and run it. He would first have to go to an authorized administrator, and have his programs added to the authorization table in his smartcard.[8] However, this security model can actually guarantee the user absolute privacy for his personal files on both the network servers and his local hard disk.

3.5.1 File Privacy

Each user can be provided with a private encryption service program[9] to be run on his local machine to encrypt any specified file before writing to a file server or local disk. This program will be authorized in a direct exchange with the smartcard to encrypt and decrypt files (see Appendix A), but with a smartcard keyboard password chosen by the user. This way, in bypassing the authentication server, not even a system administrator's program

---

[7]This applies only to programs requiring remote services. Since it is generally desirable for programs executing remote procedure calls (RPC) to bind to their servers as late as possible, a user could still compile his programs locally with no problem. He would still need the help of a system administrator to get them added to his smartcard authorization table.

[8]This also gives the administrator a chance to screen programs for known viruses and Trojan Horses. The integrity of the smartcards is critical, and modifying their encrypted program authorization tables requires cooperation with the trusted authentication server. Thus, only an authorized system administrator from within a physically secure facility will be able to modify smartcards.

[9]Extremely suspicious users could even provide their own encryption programs.

listening to the network or hiding in a server could read the user's private file(s), nor could an evil administrator use a clone of the user's smartcard to peruse the data because he wouldn't know the user's private smartcard keyboard password.

Additional protection from an evil administrator can be implemented at the human policy level. It is common practice in classified environments to have "no-lone-rules" in especially critical areas. Mischief from a lone untrustworthy insider can be significantly hampered by implementing this rule for such operations as updating users' smartcard authorization tables. The authentication server assisting such procedures could enforce such a policy by requiring two administrators to validate such operations.

3.4.2 Email Privacy

While it would certainly be possible to build a secure email system using this security model and the smartcard protocols, protection for email would extend only to those messages being sent to recipients participating in the corporate security system. A more global email solution is needed. Fortunately, Privacy Enhanced Mail (PEM) [6], which will work with existing email systems, is being developed to provide confidentiality, authentication, and message integrity as a standard service on the Internet. With PEM, each participant is assigned a public key, with private name and key binding taking place within a hierarchical service structure. Control and administration of this system will be provided by the Internet Society, a neutral non-profit organization.

3.5 Scale

Increasing the scale of distributed systems will be important in any scheme to improve worker productivity by enabling information access. Just as the vulnerability of a distributed system increases as its scale increases, the problems and benefits of heterogeneity increase. Problems increase because heterogeneity brings incompatibility; benefits increase because heterogeneity brings more varieties of capability and service to distributed systems. Thus, coherence at the protocol level is an important technique for dealing with heterogeneity in all functions, including security. The proposed security model implements coherence at the protocol level (see the Appendix A for some sample smartcard protocols).

Some approaches to security presume a certain degree of homogeneity of hardware and software of all participating nodes. For example, Secure SunOS from Sun Microsystems, Inc. is TCSEC[10] B1 security level compliant, but at the cost of requiring Sun workstation nodes running Secure SunOS [1]. This example illustrates the value of the protocol abstraction level[11] implementation in large scale systems.

Because this proposed security model is based on a Kerberos-style authentication service [12], it will scale to an infinite number of nodes over a global geographic area, and across multiple administrative realms. For instance, suppose a user is authorized for corporate access compartments X, Y, and Z. Now suppose this person is sent on a business trip to some far away exotic place such as Cleveland. He merely remembers to take his smartcard with him, and when he gets to the Cleveland corporate offices, he inserts his smartcard in the nearest visitor's courtesy node, and logs in. The authentication service performs a cross-realm authentication from the local authentication server through various realms, back to the his home realm. A list of the transited realms is included in the credentials that are ultimately presented to the Cleveland authentication server. This would all be transparent to the traveler, and he can begin work with X, Y, and Z data right away.[12]

If a traveler should lose his smartcard, he can have a temporary one issued to him by the local site administrator after sufficient identification (by picture badge, fingerprints, etc.) has been provided.

---

[10]DoD Trusted Computer Security Evaluation Criteria (see reference [3]).

[11]The protocol abstraction layer is not really a level in a hierarchy, but a position in a cycle, as described in Appendix B. This cyclic interpretation resolves the human/hardware contradiction in the use of the term "physical."

[12]This transparency of user location is not confined to authentication and authorization services. For instance, the protocol for cross-realm authentication could include a trigger to initiate a set of "travel status" services which could do things like forward his custom user-centered naming environment, email, and calendar.

4 Extension of the Model to Classified Programs

As they currently exist, the compartmentalized networks of the classified world have no need for the authentication server and smartcard security model described above because the classified program security model is based on comprehensive physical security (armor, electromagnetic shielding, and armed security officers) and trust of users based on background investigations. Additionally, there are severe criminal sanctions associated with deliberate national security compromise.

In the future however, as transparent distributed system access becomes the norm for unclassified spacecraft programs, and the significant efficiency gains achievable with large open systems become manifest, the absolute barriers of the classified compartments will come to be seen as increasingly hobbling to the productivity of the workers. This change in view will prompt managers to look for technological solutions that will enable reasonable and secure changes to the traditional classified security models.

For example, an engineer on a classified project is trusted to use an ordinary telephone without forgetting himself and revealing classified information. Program personnel are briefed on the telephone protocols, and each knows which words can be used and which cannot. The difficulty with extending this telephone model to email is due to the power of the computer. It is too easy for the user to goof and send a classified document to somebody outside by mistake! Instead of a word or phrase being spoken carelessly, the email lapse results in a significant breach of security. Therefore, outside connections to networks have never been allowed in classified areas.[13]

This traditional approach results in a significant isolation of information workers. For example, most drawings on classified projects are of unclassified components. These unclassified drawings are only available outside the classified project area for downloading from the electronic data management (EDM) system. Among the other resources that are not available to the classified facilities are process specifications and company policy manuals. This security-imposed isolation also prevents the people from participating in company email and calendar systems. This situation is exacerbated by the fact that most

---

[13]The encrypted data link shown in figure 2 is not an outside connection: the remote site is within the same security compartment.

aerospace companies use a matrix organization. Functional management takes care of personnel issues and long term human resource allocation while project management oversees the day-to-day work assignments. Hence, workers on classified programs are often physically separated from their functional managers and support services.

4.1 Service Request Messages

There is no security reason to prohibit information from outside a classified facility to enter. The trouble is, if a service request cannot get out, no server will send anything inside. I propose that a classified project could allow connection to the outside internet if a gateway guardian process is used to prevent release of classified information. The guardian would allow messages to come into the facility without restriction. It would intercept and examine all outgoing messages. It would allow simple service requests to approved servers to pass to the outside world. Figure 3 shows a typical classified facility connected to the outside world through a gateway guardian.

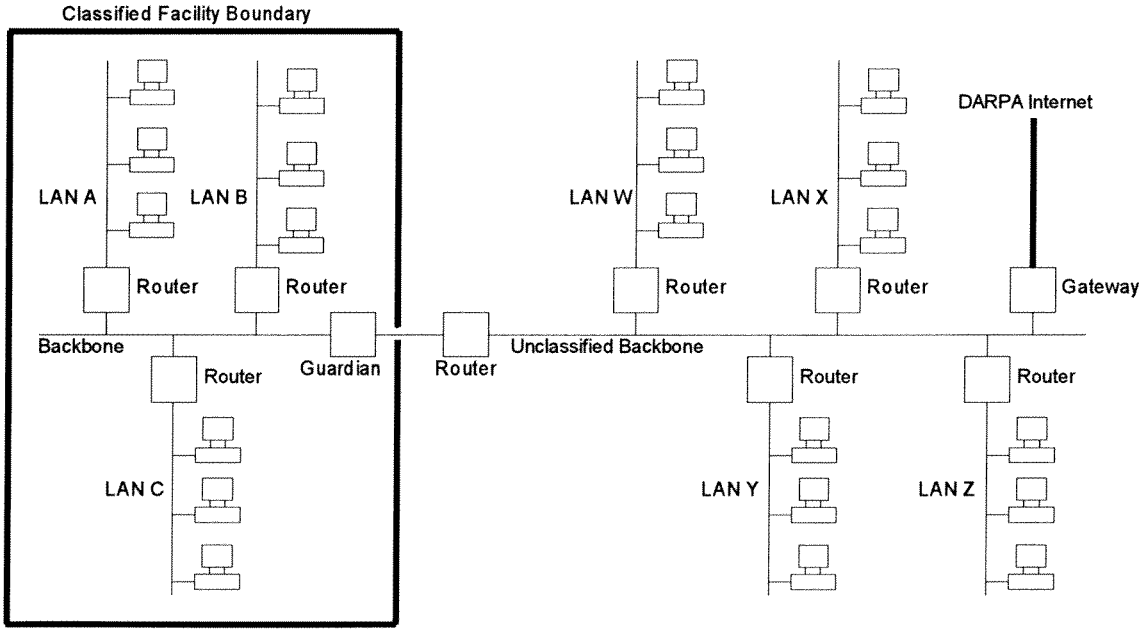


Figure 3

4.2 Email Messages

The email problem can be handled by using the same philosophy behind allowing worker access to telephones: the guardian could allow email messages under 500 bytes in length, but would trap any message longer. This will prevent anyone from accidentally attaching a

classified document or picture to the email message. As an additional safeguard, the guardian could screen all outgoing messages for forbidden words and phrases.

In the compartmentalized worlds, it is forbidden to associate some classified projects with each other. Hence the gateway guardian will also have to screen email distribution lists to ensure that recipients in disjoint compartments do not appear together on any one list.

4.3 Exception Handling

If an exception to the allowed messages should occur, the guardian will stop the message, archive it with log data for later inspection by security personnel, and inform the sender about the exception and the reason, so he will not repeat the message. Programming and maintaining these restrictions imparts cost to the system, but it will be more than outweighed by the advantages obtained. Under this gateway guardian model the workers on classified projects can enjoy the same internetwork services the open areas utilize without compromising program secrecy. Of course, if the unclassified community adopts the encryption protocols proposed in the baseline model described above (smartcard authentication and encryption of critical messages), the classified world will also have to adopt that model in order to access protected services.

5 Conclusion

I have described a method for obtaining protection and privacy in an open system that is based on the Kerberos model and extended with an original approach to smartcard technology for blunting intruder program attack and securing the blessings of privacy for the users. Adoption of this model by spacecraft contractors can both safeguard proprietary information and enhance the productivity of the now-isolated classified programs.

The irony of one traditional approach to security, the deliberate limiting of scale by imposing isolation, was mentioned. It will be an additional irony if the unclassified world should adopt these proposals while the classified world does not: we will then have an impenetrable open system together with a set of physically secure but incommunicado and internally vulnerable systems.

6 Acknowledgments

I wish to thank Larry Lynn, Taylor Nielsen, and Hank Surface who commented on an earlier draft of this paper.

Appendix A: Smartcard Protocols

For simplicity, the authentication semantics outlined here do not utilize redundancy. For increased availability, additional private keys shared with other servers might also be used.

Initial Log-in:

1. The user enters his administrator-assigned password with the smartcard keyboard.
2. The smartcard uses the authentication server's (AS's) public key to encrypt the password send it along with the smartcard's unique ID (through the PC) over the network to the AS.
3. The AS decrypts the password, and upon verification, sends to the smartcard a shared private key, encrypted in the smartcard's public key.
4. The smartcard decrypts the message from the AS, and sends the smartcard's encrypted table of programs authorized to request network services to the AS.
5. The AS decrypts the smartcard's table of programs authorized to request network services, and encrypts it in the shared private key it sent in the previous message.
6. The smartcard decrypts the received table of programs authorized to request network services and stores it in its volatile RAM.
7. User session control is returned.

Network Service Request:

1. At bind-time, a client process informs the smartcard of its intent to bind with a list of services.
2. The smartcard checks either[14] (1) the program executable file size and checksum or (2) a unique ID offered by the program against its table of authorizations, and if approved, gives the process the appropriate proxies encrypted in private keys shared with the appropriate servers.

---

[14]The exact implementation of this depends on a lot of factors. Having an authentication server participate at compile time in branding a program with a 64 bit unique ID may be the most effective and easiest to implement in the long run.

Local Encryption for Privacy:

Batch Mode:[15]

1. The suspicious user starts the local encryption service program (LESP) which then obtains the smartcard's public key.

2. The LESP identifies itself to the smartcard by showing it some shared secret encrypted in the smartcard's public key, and gives the smartcard its public key.

3. The smartcard prompts the suspicious user to enter a new password, and has him verify it as free of typographical error.

4. The smartcard generates a new key based on the password and encrypts it in the LESP's public key, and passes it to the LESP.

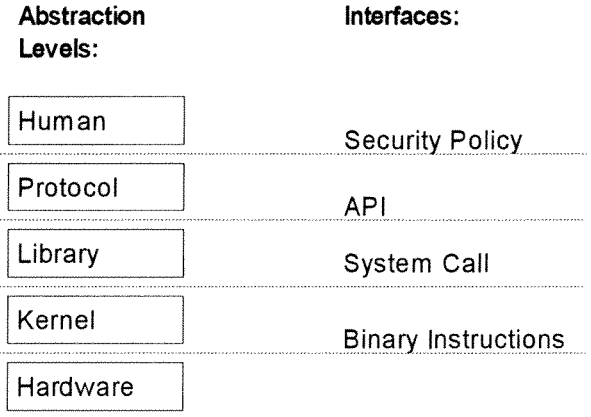
5. The LESP encrypts the batch of sensitive files, and stores them either locally or on the network file server, as directed.

---

[15]More transparent for the user would be to have applications present options to the user for encryption. The applications could then deal with the LESP directly, which would bother the user only for his password when needed.

Appendix B: The Abstraction Cycle

The communication protocol level of abstraction in distributed systems is important in implementing scalable secure systems because it is at the protocol level that interoperability can be obtained in spite of incompatibilities at lower levels. The lowest level of abstraction is usually defined at the hardware-physical level. Higher levels of abstraction hide detail and complexity, allowing the lower level differences in systems to be smoothed over. Homogeneous hardware systems are said to have binary compatibility. For network media, this level corresponds to network adapter electronics: frequency, bandwidth, modulation baud rate, etc. The figure below shows a simplified schematic of the layers and their interfaces:

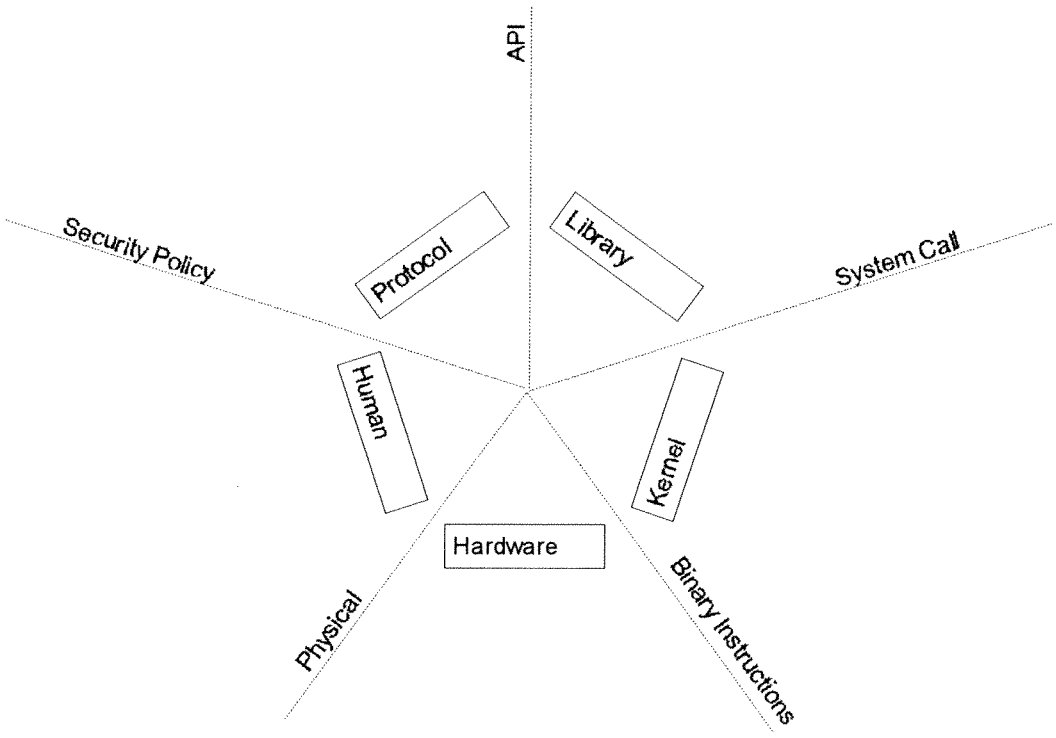


The next higher level of abstraction is the operating system kernel. Different hardware nodes running the same operating system are said to have system call compatibility. The network example of this level is the set of communication adapter modes, e.g., Ethernet's promiscuous mode reads all network packets whether they are addressed to the node or not. Layered above the kernel is often a library of user program calls, which presents an application programming interface (API) to user level programs. Systems supporting this type of interoperability are said to have API compatibility. In the network, this level corresponds to message packet conventions, i.e., packet size, address format, etc.

Communication (local and remote) protocols and user level programs are built upon the API level. Examples of communication protocols are RPC, TCP/IP, and UDP. Above the protocol level is the human level where policy originates. The security policies determine the software design as well as requirements for physical security. It is the role of physical security to ask such questions as "how long would it take a strong man with sharp tools to cut into this steel conduit, install a network monitoring transmitter, and get out?" The

human being also interfaces directly with the hardware, as when he enters key sequences on the keyboard. So the human interfaces enter into the abstraction at both a very high level (policy) and at the lowest level (raw bit sequences are sent from the keyboard with every keystroke).

The linear hierarchy illustrated above cannot accommodate this dual nature of the interfaces of human beings with secure distributed systems. However, bending the schematic around into a cycle, as shown below, seems to fit the abstraction elements together rather nicely.



As this figure shows, the human and hardware abstraction elements are not at opposite ends of a hierarchy, but are in proximity in a cycle.

References

1. Addison, Katherine, et al., "Computer Security at Sun Microsystems, Inc.," 10th National Computer Security Conference Proceedings, 1987.

2. Alford, Roger C., "The PCMCIA Redefines Portability," BYTE, vol. 17 number 14, page 237-244, December 1992.

3. DoD, "DoD Trusted Computer System Evaluation Criteria," DoD 5200.28 STD, December, 1985.

4. DoD, "Guidance for Applying the DoD Trusted Computer System Evaluation Criteria in Specific Environments," CSC-STD-004-85, June, 1985.

5. Gasser, Morrie, et al., "The Digital Distributed System Security Architecture," Proceedings of the 1989 National Computer Security Conference.

6. Nicastro, Linda, "Privacy Enhanced Mail," "UNIX Networking," a supplement to "Open Systems Today," page 62, 1992.

7. Johnson, Howard L. and Layne, J. Daniel, "A Mission-Critical Approach to Network Security", 10th National Computer Security Conference Proceedings, 1987.

8. Martin, S. Louis, "Smart Card Development Expands as Standard Nears Final Approval," "Computer Design," vol. 27, page 30, September, 1988.

9. Mullender, Sape, "Distributed Systems," page 118, ACM Press, 1989.

10. Neuman, B. Clifford, "Protection and Security Issues for Future Systems." Workshop on Operating Systems for the 90s and Beyond, Dagstuhl Castle, Germany, July, 1991.

11. Neuman, B. Clifford, "Proxy-Based Authorization and Accounting for Distributed Systems," Technical Report 91-02-01, CSE, U. of Washington, March, 1991.

12. Neuman, B. Clifford, "Scale in Distributed Systems," Readings in Distributed Computing Systems, IEEE Computer Society Press, 1992.

13. Steiner, J. G., et. al, "Kerberos: An Authentication Service for Open Network Systems," Proceedings of the Winter 1988 USENIX Conference, p. 191-210, February, 1988.

14. Voydock et al., "Security Mechanisms in High-Level Network Protocols," ACM Computing Surveys, vol.15, Number 2, p. 135-171, June, 1983.

15. Wagner, Richard J., "A Strategy for the Total Automation of Spacecraft Integration Information," TRW internal document, May, 1992.

Statistics

Word count:	4,645
Vocabulary:	1350 words
Words/sentence:	26.2
Paragraphs:	113
Gunning Fog Index:	17.5
Flesch Grade Level:	17.0