Methodology for Network Security Design

Donald Graft
Mohnish Pabrai
Uday Pabrai

DATA SECURITY ISSUES ARE BECOMING increasingly important as civilization moves toward a global information age. The migration away from paperwork-oriented ways of doing things requires the development of digital equivalents for traditional processes such as sealing envelopes, signing letters, and acknowledging receipt of items. The development of systems with such capabilities is one of the most complex and challenging tasks facing today's engineers. At the same time, the rewards to be reaped from breaking such systems acts as an attractive lure for modern criminals. One study estimates that the average traditional bank robber nets $20,000 with a 90% chance of prosecution; the average electronic funds transfer nets $500,000 with a 15% chance of prosecution [1].

An important subproblem to that of providing security in general is that of providing secure communications between centers of activity, i.e., network security. This is distinguished from the subproblem of providing security within a center of activity (e.g., a computer). This article addresses the development of a design methodology for network security based on the International Standards Organization (ISO) 7498 Open Systems Interconnection (OSI) Reference Model [2] and 7498-2 Security Architecture [3].

It should be pointed out, lest one get the impression that all the obstacles are purely technical, that legal and practical problems also stand in the way of a transition to a digital society. For example, consider a real-world attorney who acts as a "go-between" to shield a client's identity. She could be replaced with a digital entity, but that entity would not enjoy the legal privileges of the attorney-client relationship.

The Need for a Network Security Design Methodology

If network security systems are designed using *ad hoc* and unpredictable methods, their integrity will be in doubt and the transition to the information age jeopardized. Therefore, a reliable and coherent design methodology for network security is badly needed. The problem has received little attention. This can perhaps be explained by the relative immaturity of the underlying technology. Ward and Mellor observe that many engineering disciplines evolve through predictable phases [4]. In the first phase, technologies for solving a problem begin to emerge. Engineering is dominated by attempts to fit the problems to the few available solutions. In the second phase, powerful alternative technologies become available and less force-fitting of problems to solutions is required. In the third and final stage, the discipline matures and becomes fully problem-centered, with a focus on characteristics such as cost and flexibility rather than the solubility of problems.

It is our opinion that the discipline of network security is in the latter half of phase two. The transition to the third phase must be accompanied by a mature methodology that insists on a problem-centered approach. Current software engineering practices provide a useful analogy. The almost universal acceptance of a formal requirements analysis phase is an embodiment of the problem-centered approach. Software has benefited by gains in quality, development time, and maintainability. There is no reason to believe that such gains could not be achieved in the design of network security.

We have been able to find only one paper addressing, in a significant way, the issue of network security methodology [5]. These authors mention but do not develop a treatment of design, instead concentrating on the surrounding issues: definition of protected resources, statement of security policy, threat analyses, assessment and review of the operational system, and certification.

Objectives and Approach

Our objective in this article is to investigate the feasibility of defining a methodology for the design of network security. Although clearly the problem-centered approach can be achieved by defining separate requirements and implementation phases, it is not so clear that a step-by-step "cookbook" approach is feasible. For example, it may be that selection of underlying security mechanisms and design of protocols using these mechanisms are so intertwined that they cannot be treated separately. Nevertheless, we attempt to do so. We hope to expose such problems by attempting to define a methodology.

The approach taken is simple: define a methodology and attempt to apply it to a relatively simple application. By doing so, we can see where theoretical analysis as well as quantitative decision-making enters into the design.

Of course, network security design is only a part of the overall process for specification and design of any networked system. We only consider network security in this article, but a real-world treatment would need to be integrated into the overall methodology for a networked system.

0163-6804/90/0011-0052 $01.00 © 1990 IEEE

Specification Phase

Determine the System Requirements
—State the Intended Application
—Define the Security Perimeters
—Define the Required Security Services
—Define the Required Security Management Features

Identify Constraints on the Design
—Review Applicable Standards
—Determine Network Type and Topology
—Consider Organizational Factors

Design Phase

Define the Security Architecture
Locate the Required Functionality within the Architecture
Define the Service Primitives
Select Underlying Service Mechanisms
Define Service Protocols

Implementation Phase

Develop Required Hardware and Software
Testing and Verification
Performance Analysis
Accreditation and Certification
(Possible Iteration with Design Phase)

Fig. 1. Methodology outline.

A Methodology for Network Security Design

Figure 1 presents an outline of the methodology we have proposed. The following sections develop the ideas in detail.

Specification Phase

The idea of formalizing the distinction between the essence of a system (what it must do) and the implementation of the system (how it does what it must do) derives from work on software development methodologies [6]. The application of this idea to network security design ensures that a problem-centered approach is taken and that the problem is fully understood before any implementation thinking occurs.

We have found it useful to divide consideration of system specification into two components: statement of requirements and identification of constraints. Requirements are factors determined by the problem itself. Constraints are factors that derive more from the environment of the problem than from the problem itself. For example, given that the problem is to prevent disclosure of transmitted data, a requirement would be to transmit the data in unreadable form; a constraint might be that it should cost no more than one dollar per message to add secrecy.

Determine Requirements

In the requirements phase of network design, we state the problem that we are trying to solve. "Infection" by implementation thinking should be avoided at this stage.

It is important to realize when specifying requirements that "there is no free lunch." Consider the work required by a sender to transmit his data securely versus the work required of an attacker to successfully read the data. In an "insecure" system, the attacker does not do much work. In a "somewhat secure" system, the sender increases his work and the attacker's work increases proportionally. It is somewhat secure because the attacker may not have the means to perform the work, or the value of the data to him may not justify the attack. In "more secure" systems, the attacker's work increases faster than the sender's work. Taken to an extreme, an "ideally secure" system would require little sender work but very high attacker work. A variant of an insecure system is one in which the sender's work has been increased but the attacker's has not proportionally increased. The point we are making is that to obtain high attacker work values, one cannot escape additional sender work. One of the major decisions in specifying a system is to decide how much security can be afforded. This is a quantitative decision based on, among other factors, costs of performing work and monetary values of data as a function of time.

- *State the Intended Application*—This step consists simply of stating the intended application. The information should orient the designers to the problem to be solved without duplicating the more detailed information provided in the following steps.
- *Security Perimeters*—An important starting point in specifying system requirements is identifying the domain of applicability of the security services. By analogy to physical security perimeters, Branstad has developed the notion of a logical security perimeter [7]. A logical perimeter is drawn around areas in which "trust" is required, i.e., areas in which security services are not provided and protection is achieved through trusted personnel or systems. The portions of the network outside these perimeters define the domain of applicability of the security services. Branstad observes that many networks have a perimeter around the network as a whole, i.e., no security services are provided.

Care must be taken to depict the security perimeters with an appropriate level of resolution. If the resolution is too fine, then implementation thinking begins to creep in. For example, Figure 2 shows two possible depictions of security perimeters. In Figure 2a, the perimeter is shown transecting OSI layer 4. This implies two implementation decisions: an OSI security architecture is being used, and the services are located at the transport layer. Figure 2b depicts the perimeters without making implementation decisions. Of course, one can always refine the specification of perimeters during the design stage to depict implementation decisions.

- *Security Services*—In this step, a detailed statement of the required security services is made. The information should be framed in terms of application requirements and should be devoid of any consideration of specific security mechanisms or protocols. The reader is referred to [8] [9] for descriptions of security services and to [3] [7] [10] [11] for discussion of security services in the context of the OSI Reference Model. The left column of Table I summarizes possible security services.

The reader may wonder where protection against traffic analysis falls within the classification of Table I. We derive such protection from a combination of protection against message content, message length, and message time secrecy. For example, the time at which a message is sent can be concealed by means of appropriate traffic padding. More important, however, than providing a "correct" classification for all possible services is providing a classification that is appropriate for the problem to be solved. The list of services or its organization need not be rigid.

Attack recovery is presented in [3] as an optional feature attached to data integrity services. We take a broader view by treating it as a separate service category. It is conceivable that services other than data integrity could use recovery services. For example, attacks on access control mechanisms may invoke recovery services. The approach we have adopted for specifying security services is to associate with each service a key letter and commentary field. The key letter is chosen from the set: M = Mandatory, O = Optional, NS = Not Supported, and C = Configurable. M-category services must be provided. O-category services may be provided but are not mandatory. NS-category services are those that must not be provided. For example, in a treaty-verification application, authentication must be provided but data secrecy must not be provided [12]. C-category services are those that are configurable by the network administrator (usually when the security package is in-

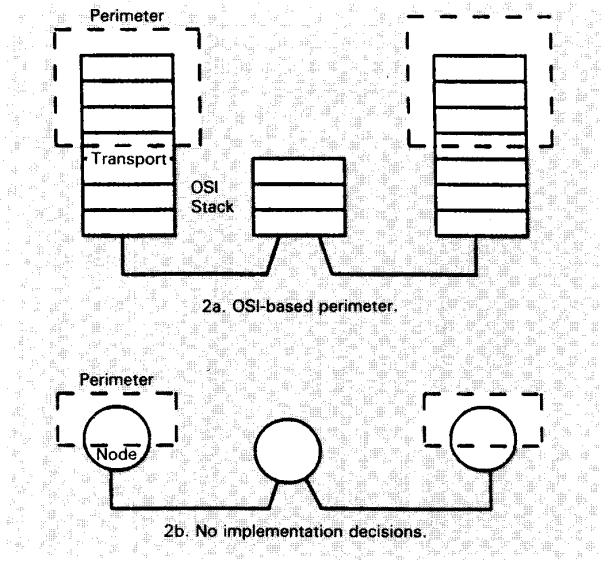

Fig. 2. Security perimeter resolution.

stalled). The comment field provides any additional qualifications necessary to fully specify the required service.

- *Security Management*—This step consists of a statement of the requirements related to the management of security. It should include consideration of areas such as whether services are negotiable both locally and on an end-to-end basis, what service combinations are allowed, event reporting and logging, configuration, and whether a central network management function is permissible. Traditionally, consideration of security management has included key distribution methods. We see this as too implementation-specific to be included at this stage. Useful discussion of security management can be found in [3] [13].

Identify Constraints

Constraints are factors that limit the designers' options but are not mandated by the problem to be solved. We divide constraints into three categories: applicable standards, network type and topology, and organizational.

- *Applicable Standards*—This material should specify the standards that must be adhered to together with any allowed deviations from those standards. A proliferation of standards is occurring in the field of network security, as is evidenced by the following list of some organizations creating standards: the International Consultative Committee for Telephone and Telegraph (CCITT), ISO, American National Standards Institute (ANSI), the National Standards Association (NSA), the National Bureau of Standards (NBS), the National Council of Schoolhouse Construction (NCSC), the Defense Advanced Research Projects Agency (DARPA), the Department of Defense, and the Department of Commerce. It must be accepted that adherence to standards can force the use of specific security mechanisms. For example, the Data Encryption Standard (DES) requires use of specific 56-bit private-key encryption method.
- *Network Type and Topology*—Specific network types and topologies can limit implementation choices. For example, authentication at connection setup time is not possible in a connectionless network.
- *Organization*—Organizational constraints are those imposed by the specifying organization. Most commonly encountered are budgetary constraints. Intended service start dates may also limit implementation options.

Design Phase

The specification phase serves as a statement of the problem to be solved and the constraints limiting the designers' implementation options. In the design phase, a solution is developed that satisfies the specifications.

Definition of the Security Architecture

At this stage, the overall security architecture is defined. Many implementations are based on the OSI Reference Model, but that is not the only option. For example, the National Computer Security Center (NCSC) has developed an adjunct to its Trusted Network Security Evaluation Criteria (TNSEC) that specifies an architecture for trusted networks [8] [14]. It is also possible to adopt a proprietary architecture. For details on the OSI Reference Model and its extension to network security, refer to [2] [3]. We concentrate here on the OSI approach because it has the potential to result in solutions appropriate for international communications (unlike such programs as NSA's COMSEC program).

Placement of Functionality Within Security Architecture

During this stage, the security functionality is placed within the chosen security architecture. We will concentrate on the OSI model for illustrative purposes. Placement of functionality within the seven defined layers of the OSI model remains both highly controversial and very interesting. The issues have been well described in [3] [7] [9] [11] [15] [16]. The issues involved in placement are both technical and practical. Examples of technical issues are: link-layer functionality cannot work with transparent intermediate nodes; application layer functionality cannot hide protocol headers; and application layer functionality can reduce the effectiveness of lower-layer services (e.g., data compression at the presentation layer). Examples of practical issues are: the amount of trusted functionality should be minimized; services should not be duplicated in different layers; and added functionality should not duplicate existing OSI functionality.

One technical issue that is very important is that placement of functionality for a given service cannot be done without considering other OSI functions that must coexist within the application. For example, if encryption is to be used together with data compression at the presentation level, it should be placed lower than compression within the architecture for two reasons: encryption placed above compression can reduce the effectiveness of the compression; and encryption placed below compression can be more effective due to the initial "scrambling" by the compression service.

Definition of Service Primitives

This stage defines the service primitives required to implement the specified services. The primitives determine the interface presented to the applications and the parameters that must be passed between architectural layers. Refer to [7] for a set of service primitives based on transport-layer placement of functionality.

Selection of Underlying Service Mechanisms

The previous stages have defined the locations and interfaces for the required functionality. At this stage, underlying mechanisms are selected to implement the services. We make an important distinction between selection of underlying mechanisms and protocols. A mechanism is a basic technology or algorithm (such as DES encryption or timestamping). A protocol is an end-to-end operation that uses one or more mechanisms to implement a service. The mechanisms are selected based on the required services, constraints, and performance factors. Descriptions of available service mechanisms can be found in [3] [7] [18]. Care must be taken to ensure that the selected mechanisms are technically appropriate for the application. For example, the low entropy of encrypted digitally en-

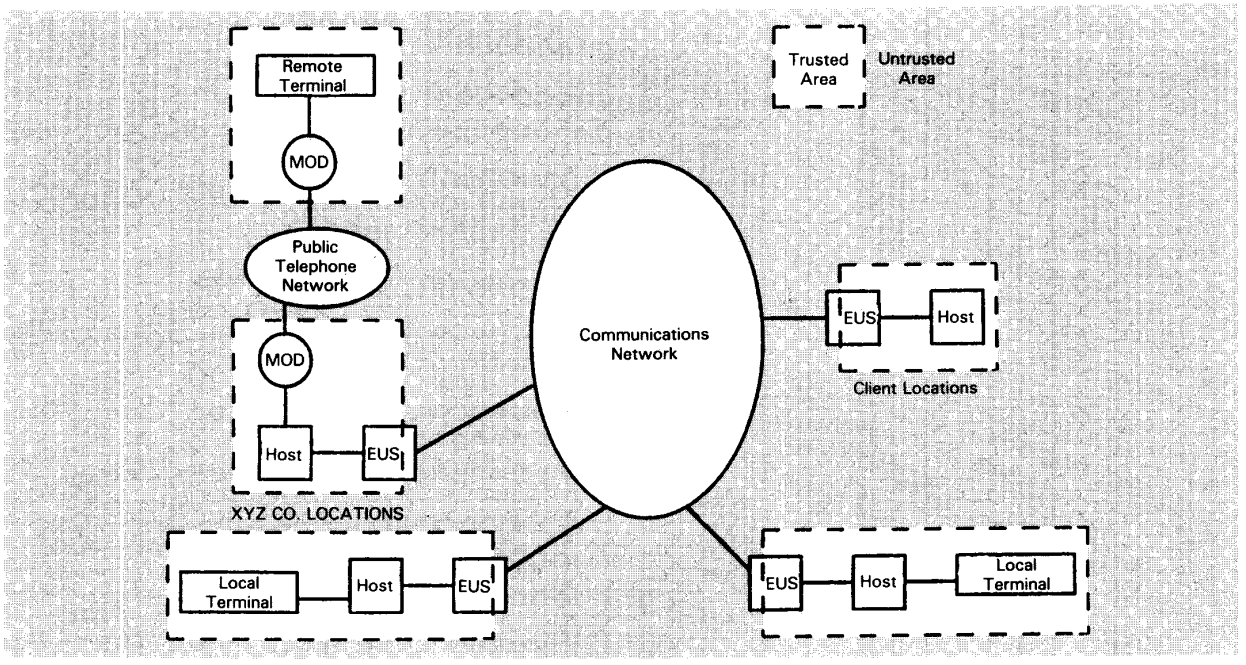

Fig. 3. XYZ Corporation security perimeters.

coded speech makes attack feasible. Simmons and Holdridge were able to produce recognizable "plainspeech" from an RSA-encrypted data stream [19].

Design of Service Protocols

At this stage, the service protocols that tie service mechanisms together to provide the required services are designed. As with service mechanisms, protocols are selected based on the required services, constraints, and performance factors. Great care must be taken to ensure that the protocol does not undermine the security of the underlying mechanisms. For example, in a very interesting paper on protocol failures, Moore observes that the theoretically unbreakable Vernam cipher (one-time pad) can be combined with Shamir's three-pass protocol to produce what appears to be an unbreakable scheme requiring no key distribution [20]! Unfortunately, if the cryptanalyst obtains ciphertext from all three passes, the plaintext is easily derivable. (Interestingly, Moore points out that there exists an encryption mechanism that is secure when combined with Shamir's protocol. This means that data secrecy can be achieved without key distribution. The downside, of course, is that the data must be encrypted, decrypted, and transmitted three times, resulting in more overhead than that imposed by reasonable key distribution protocols.)

The problem of proving correctness for security protocols is an important and very active research area. While a comprehensive theory that might guide a methodology is not yet available, there is reason to hope that such results may be obtained in the future. For general discussions of security service protocols, refer to [9] [20–22].

Implementation Phase

The implementation phase translates the design into reality. We concentrate in this article on the specification and design phases and deal only briefly with the implementation phase. This phase consists of allocating the design to hardware and software, developing the required hardware and software, testing and verifying the implementation, gathering performance data, and obtaining required accreditation or certification. The latter two activities are discussed in [5].

Application of the Methodology—An Example

In this section, we apply the proposed methodology to an invented example. The goal is not to provide a solution to a real problem or to rigorously assess alternative designs, but rather to assess the proposed methodology. Therefore, some of our specifications may seem overly simplified or inappropriate for a real-world application. Also, for the sake of brevity, descriptions are kept brief and would undoubtedly be more detailed in a real-world application. Due to space and time limitations, we do not address the implementation phase.

Specification Phase

We now provide specifications for a security application for an imaginary company, the XYZ Corporation.

System Requirements

- *Intended Application*—XYZ Corporation is a major consultant in the software development field. They provide development services for a number of clients. The clients' businesses are highly sensitive, and disclosure of design and other data would be very damaging. Nevertheless, the frequency of required contacts between XYZ Corporation and the clients necessitates communication via data networks rather than by courier. The data transmitted consists of development contracts, design specifications, completed designs, reviews, and billing. Both XYZ Corporation and the clients require acknowledgment of delivery. XYZ Corporation employees often work from remote terminals that access the hosts via modems and the public telephone network.
- *Security Perimeters*—Figure 3 shows the security perimeters for the XYZ Corporation application.
- *Security Services*—Table I defines the required security services for the XYZ Corporation application.

One point that arises from considering the perimeters and security services is that different parts of the network may have very different security needs. For example, it may be that only secrecy and appropriate access controls are needed for the re-

Table I. XYZ Corporation Security Services

Service	Category	Comments
Ensuring Confidentiality		All confidentiality services to be applied to both remote terminal and interlocation traffic.
Message Content Secrecy:		
User Data	M	
Selected Fields	NS	
Protocol Headers	O	
Message Length Secrecy	O	
Message Time Secrecy	O	
Ensuring Data Integrity		The remaining services to be applied to interlocation traffic only.
Message Insertion	M	"Message" can be interpreted as either a single PDU or a sequence of PDUs.
Message Replay	M	
Message Deletion	M	
Message Resequencing	M	
Message Alteration	M	
Message Delay	O	Indistinguishable from network delay when small.
Detection of Service Denial:		
Permanent	M	Defined as delay exceeding one minute for a given PDU.
Temporary	O	
Ensuring Authenticity		
Identity	M	Client and XYZ Corporation identities.
Location	O	
Time	M	
Ensuring Nonrepudiation		
Proof of Transmission	O	
Proof of Reception	O	
Access Control		
Network Resources	M	
End System Resources	M	
Multilevel Control	NS	
Attack Recovery	O	

M: Mandatory; O: Optional; NS: Not Supported

mote terminal connections, while all the specified services are needed for the communications network connections. Therefore, appropriate specifications should be developed for the differing parts of the network. We have dealt with this by means of comments in Table I, but a more rigorous approach would provide separate tables.

- *Security Management*—Security services shall not be negotiable either locally or end-to-end. A central management node may be employed if necessary. If so, it must be located at an XYZ Corporation location and accessible through the communications network. A trusted third party (arbitrator) is not available. A log must be maintained at each XYZ Corporation host that records attacks on system security.

Design Constraints

- *Standards*—The design should be consistent with the ISO OSI/RM Parts 1 and 2 [2] [3]. Any deviations from these standards must be justified and reviewed with XYZ Corporation.
- *Network Type and Topology*—The communications network linking locations is a connectionless packet-switched network of arbitrary connectivity. The network linking the remote terminals is the public telephone network.
- *Organizational*—There are no significant organizational constraints.

Design Phase

We now apply the methodology to the design phase. We emphasize that the intent is not to develop a rigorously complete design that would serve as input to the implementation phase, but rather to show how the steps are applied and how quantitative data guides the design decisions.

Definition of the Security Architecture

The specifications mandate a design consistent with [2] and [3]. These architectures are well-described in the ISO references and in [23].

Placement of Functionality within the Security Architecture

We have chosen to place the functionality for all of the XYZ Corporation security services in the transport layer. Our selection of the transport layer is based on the following considerations: transport is the first layer with end-to-end significance—by placing the functionality as low as possible, one conceals the most data; we rejected services at layers 1 to 3 because that would require trusted intermediate nodes; and transport seems to be the most flexible placement when other OSI functions (such as data compression) may exist.

The decision to place all services in the transport layer is consistent with [2] and [3] with on exception: Nonrepudiation is given as an application layer service in those references. We will see that the form of nonrepudiation being provided for XYZ Corporation is a weak form that can be supported at the transport layer.

Definition of Service Primitives

Due to the relative simplicity of XYZ Corporation's security services and the requirement that services be nonnegotiable either locally or end-to-end, the only primitives required are those appropriate for normal connectionless service, i.e., data request and data indication. The transport layer transparently manages the security services in an unconditional manner.

Selection of Underlying Service Mechanisms

We now consider the underlying mechanisms needed to implement the specified services. First, we look at message content secrecy. Encryption is the only available basic mechanism, given that we must send data over physically unprotected channels. In choosing an encryption method, we are faced with a work tradeoff analogous to that described earlier.

Being responsible designers, we find Vernam encipherment to be the most attractive (little sender work and very high attacker work). Unfortunately, the Vernam method requires a secret and authentic key distribution channel and keys as long as the plaintext. If such a channel was available, we could just send our original plaintext on it! Vernam encipherment is clearly inappropriate for this application.

The next most attractive method is RSA encipherment. An advantage of RSA is that the key distribution channel need not be secret but it must be authentic. A serious disadvantage of RSA is its slowness. The best Very Large Scale Integration (VLSI) implementations can support a data rate of only 1–5 kb/s, much too slow for practical applications.

After RSA, we come to the DES method. It has a good $f()/f'()$ ratio and runs about 1,000 times as fast as RSA. Unfortunately, like the Vernam cipher, it requires a secret key distribution channel. At least the key size is small relative to the size of the plaintext. We would like to use DES if the problem of the secret key channel could be overcome. Fortunately, that is relatively easily achieved by bootstrapping DES from one of the

public-key methods. For example, RSA could be used to distribute keys for a DES encryption. DARPA has recently approved an RSA/DES hybrid for electronic mail systems. The requirements included a strong form of authentication, for which RSA was ideally suited through its digital signature mode. We shall use the RSA/DES hybrid for XYZ Corporation.

Consider now the requirements for data integrity. These can be relatively easily achieved by means of a sequence number and data checksum, both done prior to the DES encryption. A timeout mechanism can detect permanent denial of service.

Having dealt with the easier services, we now face providing authentication services. We immediately run into a problem of the definition of a service and its "strength." Consider the following argument. If Alice is sending Bob DES-encrypted text that he can decipher, the data must be authentically coming from Alice because she is the only other person that knows the key. True, Bob can be sure that the data is authentically Alice's, but an outsider could not be sure because Bob could have forged message. Thus, this is a weak form of authentication. The concept of authentication strength is discussed in [12], where it is observed that this weak form is often sufficient because third-party proof is not required. Also, a claim of forgery would mean that one or the other party is not playing fair, and the other side would know it. The offended party could just "take its bat and ball and go home."

For XYZ Corporation, this weak form of authentication is sufficient. The main goal is for each company to be sure that the data it receives is authentically ascribable to the other company. This will be based on the sharing of a common secret DES key.

A detailed discussion of access controls would be extensive and is beyond the scope of this article.

Design of Service Protocols

We now consider the design of service protocols for the XYZ Corporation application. The starting point for adding a new customer is the exchange of public keys for the RSA encipherment. This will be done redundantly via such channels as mail. the public phone network, facsimile, and possibly couriers. The public keys need only be changed infrequently. An attacker would have to compromise all the redundant channels.

A more ambitious solution would be to implement a protocol such that all key distribution is performed completely within the communications network, with no reliance on outside channels. The design of such a protocol is a complicated problem and beyond the scope of this article. A step in this direction might be to implement a trusted key distribution center. The reader is referred to [24] for a typical example of a key-distribution protocol.

Having now obtained keys for RSA, the general idea for a message exchange is to first send an RSA-encrypted DES key to be used for encryption of the actual plaintext. A major issue is whether this should be done on a per-Protocol-Data-Unit (PDU) basis, on a per-time-period basis, or on a per-session basis. The per-PDU scheme can be quickly rejected as it leads to poor performance. The DES key is 8 bytes long. A typical packet-switched frame averages around 80 bytes (with lots of very little frames). Therefore, 10% of the total transmission would have to be RSA-encrypted (this would need to be a double encryption to obtain secrecy and authentication). The RSA component would therefore impose an intolerable bottleneck (recall that RSA encryption is 1,000 times slower than DES encryption). Also, there may be practical problems with passing different parts of a PDU to different encryption hardware.

Therefore, it seems appropriate to use a per-time-period or per-session approach. We have chosen to implement a per-time-period approach. The idea is that, periodically, software in the transport layer sends an RSA-encrypted key to be used for DES encryption/decryption until the next key is sent. Using this protocol, we meet XYZ Corporation's service requirements in an efficient way while retaining the advantage of not requiring a secret key distribution channel.

Conclusion

We have examined a possible methodology for network security design and attempted to apply it to a simple application. We found that several pitfalls await the requirements specifier. One problem is that defining and classifying security services is not as straightforward as one would like. Different parts of the network, for example, may have differing needs. We have found that it is not always easy to separate security mechanisms from security protocols, and certainly both need to be considered in proofs of correctness.

A more fundamental criticism of the methodology is its rigid sequencing of specification followed by design followed by implementation. Sometimes, subparts of the overall problem are found to be so large that all the steps of the method must be reapplied to that subpart. For example, providing a more desirable solution to the problem of managing public keys within the XYZ Corporation may require application of the complete methodology, beginning again at the specification stage. It may be that the methodology is insufficiently adaptable to rethinking or changes occurring during the design process.

Another criticism that might be leveled against the methodology is that it ignores the newer developments in the computing world, i.e., object-oriented programming and client-server computing. Some would argue that these developments make a "bottom-up" approach to methodology more appropriate [25] [26]. Others argue that a hybrid approach is desirable [27] [28].

Notwithstanding these criticisms, we feel that a methodology for network security design is still badly needed. We believe that methodologies for software development can be used as a foundation and have demonstrated this using the DeMarco method. Emerging methodologies may be found to be more appropriate. Nevertheless, we have shown that the idea is feasible. In the process, we have exposed some issues that must be addressed by any methodology for network security design.

Glossary

Public-Key Cryptosystem: The concept of the public-key cryptosystem was introduced by Diffie and Hellman in 1976. The basic idea is that each user A has a public-key E_A, which is registered in a public directory, and a private key D_A, which is known only to the user. E_A is used for enciphering and D_A for deciphering. Data is encrypted using the public key, but can only be decrypted by the secret private key, D_A.

RSA Encryption Algorithm: RSA is named after its developers, Ronald Rivest, Adi Shamir, and Leonard Adleman. In this public-key cryptographic system, a central key-generation authority generates two good primes, p and q, then calculates the modulus $M = p.q$ and generates encryption/decryption pairs (e_i, d_i). Each subscriber in the system would be issued a secret key d_i, along with public information that consists of the common modulus M and the complete list of public keys (e_i). Anyone possessing this public information can send a message to the nth subscriber by using the RSA encryption algorithm with the public key e_n. This protocol maintains secrecy of the message without requiring secrecy of keys.

DES Encryption Algorithm: The DES is the first and, to the present date, only publicly available cryptographic algorithm that has been endorsed by the U.S. government. Plaintext is encrypted in blocks of 64 bits, yielding 64 bits of ciphertext.

The algorithm, which is parametrized by a 56-bit key, has 19 distinct stages. The algorithm was designed to allow encryption to be done with the same key as decryption.

Vernam Cipher: Let $M = m_1m_2...$denote a plaintext bit stream and $K = k_1k_2...$a key bit stream, the Vernam cipher generates a ciphertext bit stream $C = (m_i + k_i) \bmod 2$, $i = 1,2...$

Acknowledgments

The authors wish to acknowledge the assistance of William Lidinsky and Douglas H. Smith for fruitful discussions and for calling our attention to several important references.

References

[1] A. C. Capel, C. Laferriere, and K. C. Toth, "Protecting the Security of X.25 Communications," *Data Commun.*, pp. 123–139, Nov. 1988.
[2] ISO, "Information Processing Systems—OSI Reference Model," ISO Pub. No. 7498, Oct. 1984.
[3] ISO, "Information Processing Systems—OSI Reference Model—Part 2: Security Architecture," Pub. No. 7498, part 2, 1989.
[4] P. T. Ward and S. J. Mellor, *Structured Development for Real-Time Systems*, New York, NY: Yourdon Press, 1985.
[5] L. G. Pierson and E. L. Witzke, "A Security Methodology for Computer Networks," *AT&T Tech. J.*, pp. 28–36, May/June 1988.
[6] T. DeMarco, *Structured Analysis and System Specification*, New York, NY: Yourdon Press, 1978.
[7] D. K. Branstad, "Considerations for Security in the OSI Architecture," *IEEE Network Mag.*, pp. 34–39, Apr. 1987.
[8] M. D. Abrams and A. B. Jeng, "Network Security: Protocol Reference Model and the Trusted Computer System Evaluation Criteria," *IEEE Network Mag.*, pp. 24–33, Apr. 1987.
[9] V. L. Voydock and S. T. Kent, "Security Mechanisms in High-Level Network Protocols," *Comp. Surveys*, pp. 135–171, June 1983.
[10] L. K. Barker and L. D. Nelson, "Security Standards—Government and Commercial," *AT&T Tech. J.*, pp. 9–18, May/June 1988.
[11] M. Harrop, "Security in Open Systems," *Networks for the 1990s*, R. Reardon, ed., New York, NY: John Wiley and Sons, 1988.
[12] G. J. Simmons, "How to Insure that Data Acquired to Verify Treaty Compliance are Trustworthy," *Proc. of the IEEE*, pp. 621–627, May 1988.
[13] D. Denning, "Protecting Public Keys and Signature Keys," *IEEE Comp.*, pp. 27–35, Feb. 1983.
[14] NCSC, "Trusted Network Interpretation," NCSC Pub. No. NCSC-T6-005, July 1987.
[15] B. C. Karp, L. K. Barker, and L. D. Nelson, "The Secure Data Network System," *AT&T Tech. J.*, pp. 19–27, May/June 1988.
[16] J. J. Tardo, "Standardizing Cryptographic Services at OSI Higher Layers," *IEEE Commun. Mag.*, pp. 25–27, July 1985.
[17] E. F. Brickell and A. M. Odlyzko, "Cryptanalysis: A Survey of Recent Results," *Proc. of the IEEE*, pp. 578–593, May 1988.
[18] W. Diffie, "The First Ten Years of Public-Key Cryptography," *Proc. of the IEEE*, pp. 560–577, May 1988.
[19] G. J. Simmons and D. B. Holdridge, "Forward Search as a Cryptanalytic Tool Against a Public-Key Privacy Channel," *Proc. of the Symp. on Security and Privacy*, pp. 117–128, 1982.
[20] J. H. Moore, "Protocol Failures in Cryptosystems," *Proc. of the IEEE*, pp. 594–602, May 1988.
[21] R. De Milo and M. Merritt, "Protocols for Data Security," *IEEE Comp.*, pp. 39–51, Feb. 1983.
[22] R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Commun. of the ACM*, pp. 993–999, Dec. 1978.
[23] J. Henshall and S. Shaw, *OSI Explained—End-to-End Computer Communication Standards*, Chichester, U.K.: Ellis Horwood Limited, 1988.
[24] W. Lu and M. K. Sundareshan, "Secure Communication in Internet Environments," *IEEE Trans. on Commun.*, pp. 1,014–1,023, Oct. 1989.
[25] S. C. Bailin, "An Objected-Oriented Requirements Specification Method," *Commun. of the ACM*, pp. 608–623, May 1989.
[26] B. D. Kurtz, D. Ho, and T. Wall, "An Objected-Oriented Methodology for Systems Analysis and Specification," *Hewlett-Packard J.*, pp. 86–90, Apr. 1989.
[27] P. T. Ward, "How to Integrate Object Orientation with Structured Analysis and Design," *IEEE Software*, pp. 74–82, Mar. 1989.
[28] K. Shumate, "Layered Virtual Machine/Object-Oriented Design," *Proc. of the Fifth Washington ADA Symp.*, June 1988.