WSRC-IM--90-83-7

DE92 019928

REVIEWS OF COMPUTING TECHNOLOGY: SECURING NETWORK APPLICATIONS: KERBEROS AND RSA (U)

by

S. M. Johnson

Westinghouse Savannah River Company
Savannah River Site
Aiken, South Carolina 29808

A technical report being deposited at OSTI

Reviews of Computing Technology:

Securing Network Applications: Kerberos and RSA (U)

by S. M. Johnson

WSRC-IM-90-83-7

June 1, 1992

Authorized Derivative Classifier

G. F. Hayhoe

G. F. Hayhoe

Westinghouse Savannah River Company
P. O. Box 616
Aiken, SC 29802

Table of Contents

1 Introduction

The phrase *computer security* invokes images of hackers, viruses, and other malicious attempts to access and manipulate data. However, the threat of improper use of data is not the only reason to incorporate security measures into the computing environment. Some of the technologies developed to control access to information resources are necessary to build a robust production environment for client-server computing.

Three computer security topics are especially important to networked environments: authentication, authorization, and confidentiality. Authentication is the process of determining the identity of a particular computer or human user called a principal. Authorization is the process of one principal granting some level of access to another. Confidentiality is the process of ensuring that a message passed between two principals cannot be read by a third party. In a client-server environment any conversation must begin by identification of the two principals (authentication). Once the identity of the principals has been established, the level of access to data or services each participant wishes to grant the other must be set (authorization). Authorization for files or system services is handled by the operating system, but authorization could be handled by an application for such things as database access. In addition to these measures, some steps may be taken to ensure that any messages passed between the participants cannot be read by anyone else (confidentiality).

This paper will focus on the first step in establishing network security, authentication, and describe the basic function of both RSA and *Kerberos* as used to provide authentication and confidential data transfer services. It will also discuss the Digital Signature Standard and the market acceptance of each. Proper identification of the principals involved in a network dialog is a necessary first step in providing network-wide security comparable to that of stand-alone systems.

In a distributed client-server computing environment, it is desirable to have a single username and password for each user. This username/password will define the user's access to all services on the network. Without a naming and authentication service,* users would have to supply the correct username and password for each application service they wished to access. This would be difficult to use, difficult to administer, and more vulnerable to security breakdowns.

Kerberos and RSA both form the basis for complete network application security systems. *Kerberos* is designed primarily to provide user authentication, although it does provide an easy mechanism for implementing confidential data exchange. RSA is an encryption technique most often used to provide confidential transfer of data but can be used as the basis for a user authentication system. Although neither *Kerberos* nor RSA directly provides an authorization service, they are both foundations upon which authorization services may be built. In either case, the most important aspect for a network administrator to consider is not the absolute security of RSA or *Kerberos* (they are both quite secure) but the administrative overhead of providing and distributing keys and names to the end users.

An authentication system using either *Kerberos* or RSA would likely appear transparent to the end user. The login process appears to be identical to a stand alone login. It is the work behind the scenes that is important to a distributed application environment. A distributed application environment would be too cumbersome to use if users were required to enter (and remember) a user name and password for every different system they accessed. It would also be terribly difficult to administer and secure.

For a diverse computing environment such as that at SRS, it is important to establish a standard method of defining and validating users to use network services. Software quality assurance requirements may dictate that only authorized users be able to access certain files during the development process or that only certain application versions be accessible. An authentication service may be critical to meeting these requirements.

In practice, the two most referenced systems for providing authentication, authorization, and confidentiality in a network environment are the *Kerberos* system and the RSA system. The *Kerberos* and RSA systems are often thought of as competing technologies and are often used to solve the same set of problems. However, *Kerberos* and RSA can be complementary technologies, and each has its strengths. Both systems can be considered de facto standards for user authentication and confidential exchange of data although there are "camps" of vendors who prefer either *Kerberos* or RSA.

* For the purposes of this paper, a *server* is a computer that hosts a network application called a *service*.

2 Encryption Systems

There are two basic strategies used to encrypt data for transfer bet veen two parties: secret key encryption systems and public key encryption systems. The *Kerberos* network authentication system from MIT is an example of a secret key system. The RSA encryption system from RSA-DSI, Inc. is an example of a public key system. This section describes secret and public key encryption and highlights their differences.

Secret Key Encryption

In a traditional secret key encryption system, an encryption procedure is used to transform a message from plain text into a *cipher text*. The encryption procedure is a mathematical function that uses a *key* which is shared between two parties wishing to communicate. A message encrypted using a given key may be decrypted only by using the same key. This approach is like placing the message in a mathematical strongbox with a resetable combination lock. The sender and receiver exchange the combination (key) and can then easily lock and unlock (encypher and decypher) messages. Secret key encryption systems are technically refered to as *symmetrical* encryption systems because the same key is used to encrypt and decrypt messages. They are called secret key systems because the security of the system is completely dependent on keeping the key secret.

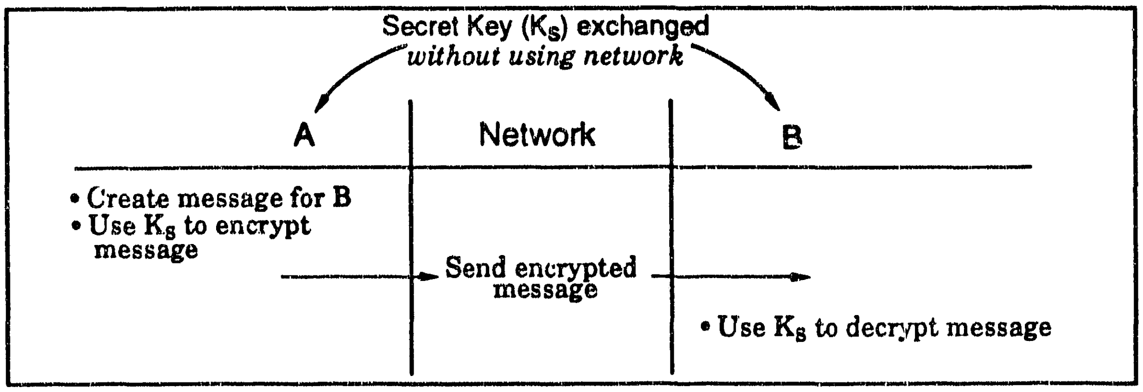

Secret Key (K_S) exchanged
without using network
A
Network
B
• Create message for B
• Use K_S to encrypt message
Send encrypted message
• Use K_S to decrypt message

Figure 1: Secret key encryption

Secret key encryption systems have the following properties:

- Decryption reverses the process of encryption, producing plain messages from encrypted messages.
- It is "impractical" to decrypt without the appropriate key. That is, we mean that it is very difficult mathematically or computationally to solve for the key, given an encrypted message.

For two parties, A and B, to exchange a private message over an open transmission medium using a secret key encryption strategy, they must first exchange a common key and an encryption procedure. The key must be exchanged via another, secure method before messages can be passed. Then A encrypts his message using the encryption procedure with the shared key and sends the message to B. Finally, B uses the shared key to decrypt the message.

The most used method for performing secret key encryptions is the Data Encryption Standard, DES. DES is defined in ANSI X3.92 and as Federal Information Processing Standard number 46 (FIPS 46). It is widely accepted to be quite secure and is used in a wide variety of applications. *Kerberos* uses DES for its encryption operations.

Public Key Encryption

Public key encryption systems are technically *asymmetrical* encryption systems because they use a matched pair of keys to encrypt and decrypt messages. In a public-key system, the encryption procedure uses a key. There is also a decryption procedure that operates on encrypted messages to produce the original messages. Instead of a single key that is used both to encrypt and decrypt a message, there is now a matched pair of keys, one of which is used to encrypt a message and the other to decrypt it. A public key system is like a mathematical strongbox with a new kind of combination lock that has two combinations, one for locking and one for unlocking the lock. Because the locking combination is public, anyone can lock up information, but only the intended recipient who knows the unlocking combination can unlock it.

Public key systems share the same two properties of secret key systems:

- Decryption reverses the process of encryption producing plain messages from encrypted messages. (In a public key system the decryption key is not the same as the encryption key.)
- It is impractical to decrypt without the proper key.

Public key systems also possess two additional properties:

- It is practical (with a computer) to generate mated key pairs.
- It is impractical to obtain the decryption key from the encryption key.

It is this last property that gives public key encryption its unique properties.

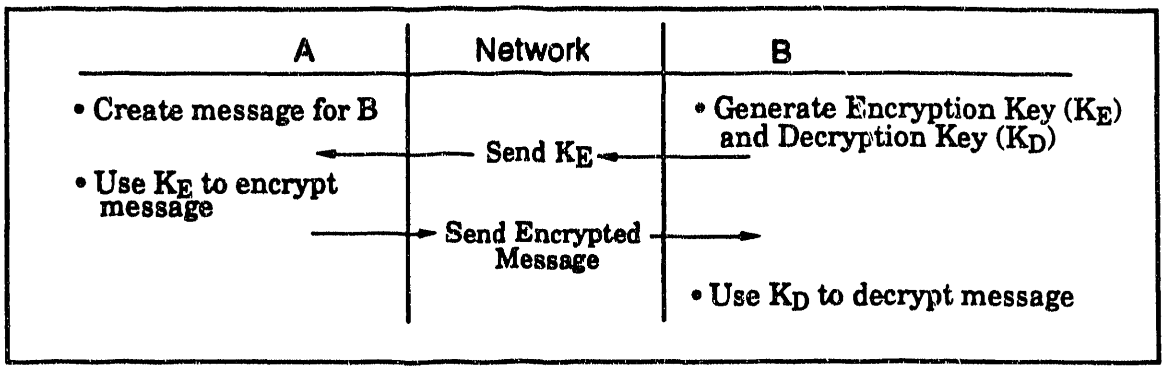

A
Network
B
• Create message for B
• Generate Encryption Key (K_E) and Decryption Key (K_D)
Send K_E
• Use K_E to encrypt message
Send Encrypted Message
• Use K_D to decrypt message

Figure 2: Public key encryption

Passing a message over an insecure medium from A to B using a public key system requires the following steps:

- B generates an encryption key and its decryption key mate. The decryption key. is held secret by B from all others, including A.
- B sends the encryption key to A. This transmission does not have to be secure since the encryption key does not need to be kept secret.
- A encrypts his message with the encryption key and sends the encrypted message to B.
- B decrypts the encrypted message using the decryption key, which never has to be transmitted across the network.

It is important to note that the security of this system does not depend on keeping the encryption key secret. Even if the encryption key is revealed, the security of the system is not endangered since decrypting with the encryption key will not work.

It is true that secret key systems have the advantage of requiring less computational and transmission overhead for two parties to conduct a private conversation than public key systems, but the chief difficulty in maintaining the security of secret key systems is the exchange of the secret keys themselves. The security of the system depends on the uncompromised exchange of keys.

Public key encryption systems, on the other hand, have somewhat more computational overhead and require more message exchanges than secret key systems, but maintaining security while exchanging keys is not a problem. The exchange or publication of the public keys is an administrative problem that must be solved in a practical system.

It cannot be flatly stated that either a secret key or public key strategy is *better*. Many factors influence the security of an encryption system. It is safe to say that for use in a network authentication system in an unclassified environment, both are capable of providing adequate protection.

3 Kerberos

Kerberos is a trusted third-party, secret key authentication service used to verify user's identities. *Kerberos* was designed to provide an authentication service, but applications can easily use it for confidential data exchange. *Kerberos* was developed at MIT as part of project Athena, a very large-scale research project pioneering the creation of a distributed, client-server system. *Kerberos* is named after the three-headed dog of Roman mythology who, as the servant of Pluto, guarded the gates to the underworld in exchange for immortality.

In the *Kerberos* system, each user and network server has an encryption key shared only with the *Kerberos* authentication server. The encryption itself is done using the Data Encryption Standard (DES) algorithm. (Export of the DES algorithm to foreign countries is prohibited.) Timestamps are used to detect and prevent replay of previous sessions. *Kerberos* offers network security comparable to the C2 level of operating system security as defined by the Department of Defense Trusted Computer System Evaluation Criteria known as the "Orange Book" (DoD 5200.28-STD).*

Project Athena implementors made the assumption that individual hosts were not under the administrative or physical control of the computing organization. A user could reboot his workstation or even reload system software from tape. It was not enough to guarantee physical control of a server because someone elsewhere on the network may be masquerading as that server. Therefore, a user must prove his/her identity to the server, and the server must also prove its identity.

*The "Orange Book" criteria are divided into four divisions: A, B, C, and D, with A representing the most comprehensive security. Division C is divided into two classes, C1 and C2, where C2 represents a higher class of security.

Computer systems containing C2 security features provide for discretionary (need-to-know) protection and for accountability of subjects and the actions they initiate. Discretionary access control allows users to specify and control sharing of data or information under their control. The access controls provide that system objects are protected from unauthorized access.

All classified multi-user systems (except process control) must conform to the C2 standard by July 15, 1992. C2 security is recommended for multi-user systems that process Unclassified Controlled Nuclear Information (UCNI), that have a heterogeneous population base, or that might be vulnerable to attack by unauthorized personnel.

Several design requirements were imposed on this identification mechanism:

- It must be secure enough that an attacker does not find the authentication mechanism to be the weak link. Someone with network access should not be able to watch the network and obtain enough information to impersonate another user.
- Many network services would depend on *Kerberos*, so it had to be reliable.
- It should be as transparent as possible. A user should not realize authentication is taking place.
- Finally, it should be scalable to support a large number of users and services, while accommodating hosts that do not support this authentication scheme.

Kerberos is called a trusted third-party system because the information supplied by the *Kerberos* server must be considered reliable for the system to work. The *Kerberos* server itself must be kept secure to maintain the trust of both clients and other application servers.

Kerberos operates as an independent server, maintains a database of its clients and their passwords, and generates session keys dynamically. These clients are both users and network services. The keys are large numbers shared between *Kerberos* and the client. Keys are delivered to the user by encrypting them using the user's password. *Kerberos* uses these keys to create messages which convince one client that another is who he or she claims to be. *Kerberos* does this by creating a temporary key, the session key, which is shared between a pair of communicating clients.

Using this mechanism three levels of security are possible. Individual applications determine the appropriate level. The first, and most common, merely establishes the identity of client and server (authentication). The second authenticates the sender/receiver of each message but does not care whether the contents are disclosed (a stricter form of authentication). The third authenticates and encrypts each message (confidentiality). *Kerberos* itself uses this method to send passwords over the network.

The original *Kerberos* system at MIT was implemented on UNIX systems. Most *Kerberos* implementations are still UNIX-based, using TCP/IP for the network transport. *Kerberos* is not constrained to UNIX, however. IBM bundles *Kerberos* with its TCP/IP package for OS/2 and has stated they will provide it for their other operating systems as well. DEC has announced they will provide *Kerberos* for VAX/VMS systems.

Kerberos Operation

Kerberos is an authentication service. An authentication service must assign names to those things that will be authenticated. In *Kerberos*, users and servers are the same as far as names are concerned. A *Kerberos* name consists of three parts, a primary name, an instance, and a realm, expressed as *name.instance@realm*. The instance or the realm may be implied and the name expressed in shortened form. Some examples of legal *Kerberos* names are listed below:

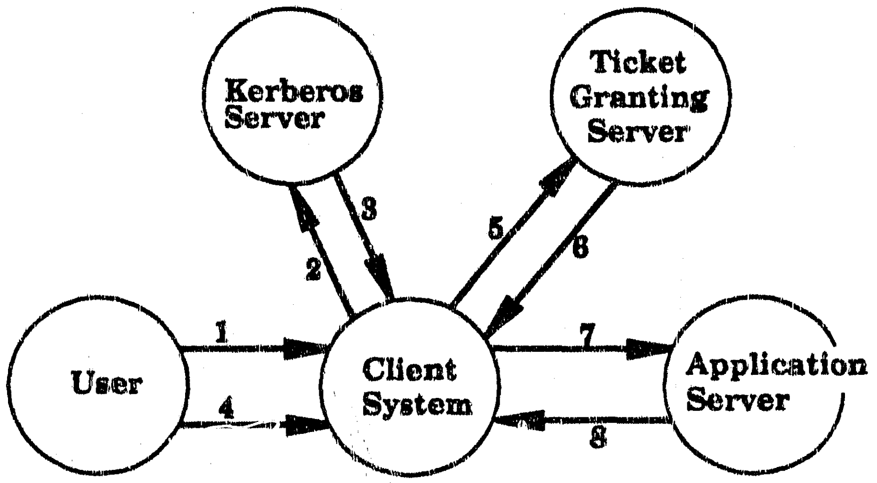
smj
ajj@SRS.GOV
telnet.slsrp1@SRS.GOV

The *primary name* is the name of the user or service. The *instance* is used to distinguish between variations on the primary name. A user may have several instances which have different special privileges, such as system administrator. For services, the instance usually refers to the name of the machine on which the server runs. The *realm* is used to distinguish between administrative domains. An entire company or university is an appropriate administrative domain.

Kerberos uses two types of credentials: *tickets* and *authenticators.* Both use secret key encryption but use different keys. Tickets are used to pass the identity of the user between the authentication server (referred to as *Kerberos*) and the application server. Information contained inside the ticket is used to ensure that the person using the ticket is the person to whom it was issued. The authenticator contains additional information which can be compared against the ticket to prove that the client is the same one who was issued the ticket.

A ticket is valid for a single server and a single client for a single session. It contains the name of the server, the name of the client, the network address of the client, a time stamp, a lifetime, and a random session key. All of this information is encrypted using the key of the server where it will be used. This ticket may be used over and over for access to the server until it expires. Because the ticket is encrypted in the key of the server, it is safe to allow the user to pass it to the server directly without having to worry about the user modifying the ticket.

An authenticator is used only once at the time a client wants to use a service. This is not a problem because the client application builds the authenticator itself. It contains the name of the client, the workstation's TCP/IP address (*Kerberos* is TCP/IP-based), and the current workstation time (workstation and server time only need to be syncronized within several minutes). The authenticator is encrypted in the session key which is part of the ticket.



Phase One: User obtains Credentials

1. User supplies username.
2. Client system sends a request for a Ticket Granting Server (TGS) ticket.
3. Kerberos returns a ticket for TGS encrypted in the key of the TGS and a session key for the TGS, all encrypted using the user's password.
4. User supplies password, and the client system uses password to decrypt ticket and session key.

Phase Two: User requests authentication

5. Client system sends a request for an application server ticket to TGS using the ticket obtained in step 3 plus an authenticator encrypted using the session key obtained in step 3.
6. TGS returns a ticket for server encrypted in the key of the server plus a session key for use with this server, all encrypted in the session key for the TGS.

Phase Three: User presents credentials

7. Client system send ticket obtained in step 6 plus an authenticator encrypted using the session key obtained in step 6.
8. Application server grants request.

Figure 3: A Kerberos session

Kerberos authentication is done in three phases. Figure 3 is a summary of the three phases. These three phases enable a user to establish his identity to an applications server. To do this, the user must give a server a ticket along with proof that the ticket was originally issued to him and not stolen. In the first phase, the user obtains credentials in requesting access to other services. The user then requests authentication to a specific service (phase 2). Finally, he presents his credentials to the server (phase 3).

It is important to remember that the *Kerberos* server is the clearing house for all authentication information. It contains the names of all network users and application servers and their encrypted passwords. The keys used between application servers and clients are not stored anywhere, but are generated on demand by the *Kerberos* server. The *Kerberos* server may be replicated so that it does not become a single point of failure for the entire network.

Except for the initial transfer of the secret key for the current session, which is encrypted in the users' password, no keys are ever transfered over the network. All data encryption uses the session key which is valid for the current session only. A workstation with no users attached contains no keys or passwords. Workstations retain a users' password in RAM only long enough to decrypt the session key. The password is then discarded.

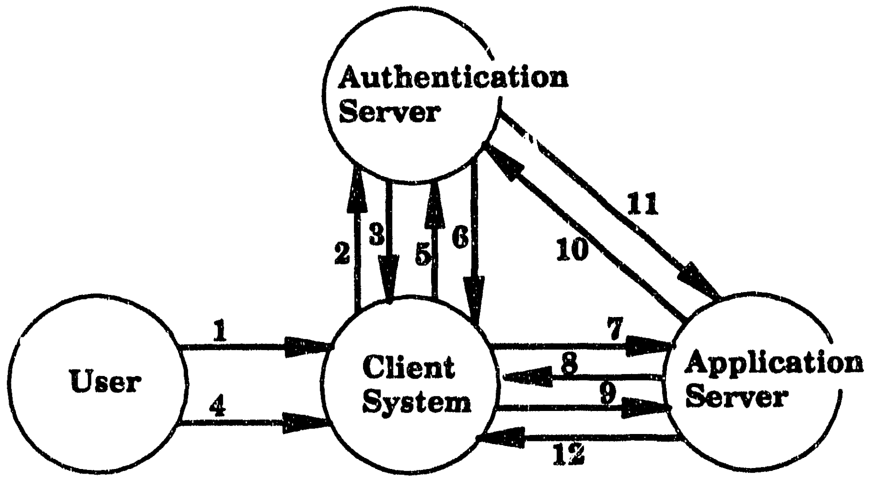
4 RSA

RSA is a public key encryption system developed by three scientists (R. L. Rivest, A. Shamir, and L. Adleman, hence RSA) at MIT in the 1970s. It has proven to be very secure despite years of attempts to defeat its encryption algorithm. While very secure, RSA is only one component of a successful system for providing authentication, authorization, and confidentiality. This section introduces the supporting infrastructure required to use RSA to provide security in the real world. A detailed description of the RSA algorithm is beyond the scope of this review.

The RSA encryption method is based on the fact that it is easy to generate two large prime numbers and multiply them, but it is much more difficult to factor the result. The product can be made public as part of the encyphering key without compromising the factors which constitute the decyphering key.

In order to provide for user authentication similar to *Kerberos,* an authentication service is required. This service would identify a user and distribute the matched encryption/decryption key pairs as indicated in Figure 4. Because an RSA key is hundreds of bits long and should be randomly generated, it is not appropriate to use as a user password. A typical RSA-based login dialog would require that the user's password be used as a key to a secret key encryption which would be used to deliver the private RSA key. Figure 4 illustrates the steps in such a dialog.

A login dialog such as the one illustrated appears identical to the end user to a normal login to a stand-alone system. Like *Kerberos,* this method has three phases; however, there are more steps in the RSA method, and the encrypting/decrypting requires more computer time than *Kerberos'* DES encryptions. In the first step, the user obtains the private key of a matched public/private key pair to be used for the current session. In the second phase, the user validates the key received by requesting authentication from the authentication server itself. Finally, the client computer authenticates itself to an application server.



Initialization
1. User supplies username.
2. Client system sends authentication request to authentication server.
3. Authentication server returns the private key for this session encrypted using the user's password.
4. User supplies password and client system decrypts the session key using the password.

Initial Authentication
5. Client system sends authentication request encrypted using the session key.
6. The authentication is confirmed.

Application Authentication
7. Client system sends a request for service to the application server.
8. Application server returns a random number.
9. Client server sends a message containing the random number encrypted in the private session key.
10. Application server asks for the client's public session key.
11. Authentication service returns the public session key.
12. Application server uses the public key to decrypt the request from step 9 and confirms request.

Figure 4: An RSA authentication service

5 Digital Signature Standard

There is another competing public key system, the Digital Signature Standard (DSS). The National Institute for Standards and Technology (NIST) has developed and circulated a draft Federal Information Processing Standard (FIPS) for DSS. The DSS is a public-key encryption system using an algorithm different from the RSA scheme. The DSS is designed to address only the security issue of confidentiality. The algorithm chosen by NIST was jointly developed by NIST and the National Security Agency (NSA). This algorithm has been the target of a great deal of criticism due to the belief of many security analysts in the industry that the chosen algorithm is much weaker than RSA s. The NSA has also been accused by many of the same experts as being unwilling to back an algorithm that they couldn't break. The standard also does not address software required to embed the DSS into a working system.

A signature applied to a paper document provides evidence that the document was prepared or approved. by the person signing the document. A digital signature as defined by NIST is a method for determining that a given document is authentic (*document* as used here is any piece of digital information). It should not be confused with a digitized copy of a written signature. A digital signature ensures that the document was created by the party indicated by the "signature" and is in its original form. In the case of a digital signature, this is accomplished by encrypting a piece of information which can validate the document. This information is the result of a function performed on the document itself. An example of such a function is a checksum. A checksum applied to a digital document is a small amount of data that *summarizes* the document. A signature would consist of the checksum encrypted in the private key of the person signing the document.

In practice, a signed document is delivered to a recipient who uses the public key of the sender to decrypt the signature. The appropriate function is applied to the body of the document and the result compared to the value in the signature. The recipient is thus assured that the document was "signed" by the indicated person (the recipient was able to decrypt the signature) and that the contents of the document are intact (the function produced the same result).

6 Summary

Kerberos and RSA are both recognized approaches to the issues of authentication, authorization, and confidentiality in data communications. Because both have significant numbers of commercial proponents who have committed to using their different approaches, the industry must develop some convergence technologies. In the meantime, it appears that large installations composed of significant numbers of *Kerberos* and RSA computers will have to implement a dual strategy for supporting network authentication.

The Open Software Foundation Distributed Computing Environment (OSF/DCE) uses a *Kerberos*-based authentication scheme. However, the OSF has stated that they will look at incorporating RSA-style encryption into their overall communications model for use in confidential messages such as digital signatures and private e-mail.

The OSF-aligned vendors of system software, including IBM, DEC, and HP, will likely use whatever security features OSF/DCE provides. This means that *Kerberos* will be widely implemented. SUN MicroSystems, although not aligned with OSF, is also offering *Kerberos* enhanced versions of some communications services such as the Network File System (NFS).

Two vendors, Microsoft and Novell, have both stated that they will not offer OSF/DCE, and Novell is building RSA-based authentication into their products. Given the size of the DOS market, it is inevitable that DCE will someday be provided for the PC by third parties. The situation for Novell servers is different. Currently, a security information gateway between the Novell and OSF worlds seems to be the most likely bridge for authentication services.

Apple Computer was committed to using RSA before the Apple-IBM alliance. Now it appears that Apple will have some degree of alignment with OSF, but this may only extend to A/UX systems. (A/UX is Apple's version of UNIX™.)

The most important consideration in choosing a system for network authentication is not the cryptographic strength of the system but the vendor support across the platforms in use at SRS. *Kerberos* and RSA authentication systems will have to coexist for the next several years because SRS strategic vendors fall into both camps. Vendors such as Novell and the OSF will probably provide tools for exchanging authentication information between *Kerberos* and RSA environments.

7 References

Adleman, Leonard M., and Rivest, Ronald L., "The Use of Public Key Cryptography in Communication System Design," *IEEE Communications*, November 1978, Vol. 16, No. 6.

Champine, George A., *MIT Project Athena: A Model for Distributed Campus Computing*, Digital Press, 1991.

Garfinkel, Simson, and Spafford, Gene, *Practical UNIX Security*, O'Reilly & Associates, Inc., 1991.

Gustavus J. Simmons, ed., *Contemporary Cryptology: The Science of Information Integrity*, IEEE Press, 1991.

Malamud, Carl, *Stacks: Interoperability in Today's Computer Networks*, Prentice Hall, 1992.

Steiner, J. G., Neuman, B. C., and Schiler, J. I., "Kerberos: An Authentication System for Open Network Systems," *USENIX Conference Proceedings*, Winter 1988.

END

DATE
FILMED
10/22/92