Cryptography in Public Internetworks with SunScreen

Introduction

C.S. & P., a Fortune 100 investment firm with corporate offices in San Francisco, is about to change its discount rate and implement a new policy on overseas investments. In order to communicate their new rate and policy information to their regional offices over a public internetwork, they need some way of:

- Ensuring that no one on the public network can read it,
- Proving that the information is really from the corporate office, and
- Proving that the information has not been modified along the way.

In other words, the corporate office needs a way of transmitting the information over a public network that ensures the corporate headquarters that only the regional offices can understand it (*privacy*) and ensures the regional offices that only the corporate headquarters could have signed it and sent it (*authentication*). These are the elements of secure communication.

The problem of secure communication is not new, but it presents new challenges as we move towards a paperless society. Messages on paper are shipped in sealed envelopes and bear written signatures. Digital messages are broadcast around the world by satellites and can be copied and altered by computers along the way. How do you protect a digital message so that only the intended recipients can understand it? How do you create an unforgable but easily verifiable digital signature? The solution to both of these problems lies in using cryptography as an integral part of data transmission in modern networks. Sun's Internet Commerce Group has developed SunScreen™, a product line of dedicated devices incorporating cryptography at the network layer to provide privacy and authentication over insecure public networks, such as the Internet.

This paper explores cryptography as it applies to secure data communications in business. The next section provides an overview of the current status of commercial cryptography and introduces the concepts of shared secrets, digital keys, and public key cryptosystems. The section following discusses authentication, digital certificates, and key management. After this we go further into key management by describing a simple key management system from Sun

Microsystems called SKIP that is used at the network (IP) layer. The final section returns to the example from the first paragraph, and shows how the transaction is executed using a SunScreen™ incorporating SKIP.

Current Status of Cryptography

Cryptography, as applied to data communications, is a matter of taking the original message, and producing an encoded version by using a special piece of information known to the sender and receiver. The original message is called the *plaintext*; the special information is called the *key*, and the resulting message is called the *ciphertext* (Figure 1). If the cryptosystem is well designed and only the sender and the receiver know the key, then they can be confident that no one else has seen the plaintext message.

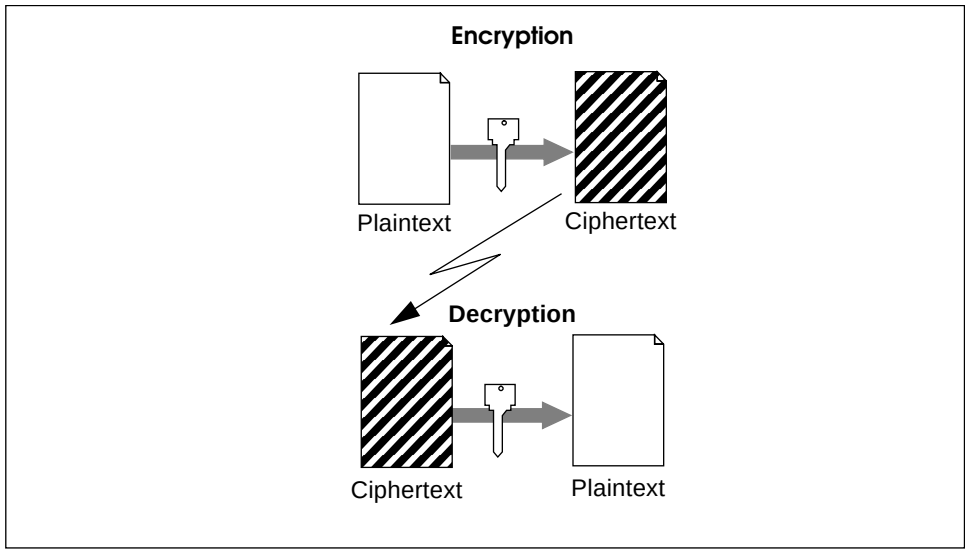

Figure 1 Encryption and Decryption

Modern cryptosystems, like the data on which they operate, are digital, they work by taking the digital representation of the plaintext, and manipulating it mathematically under the control of the digital key to produce the ciphertext.

What distinguishes the sender and the receiver from everyone else in the world is their knowledge of one or more cryptographic keys that enable them to read the encrypted messages. In conventional cryptographic systems, the sender and receiver know a single key in common and keep this key secret from everybody else. Such an arrangement is called a *shared key cryptosystem* (Figure 2). The best known of these current commercial use is the U.S. Data Encryption Standard(DES).

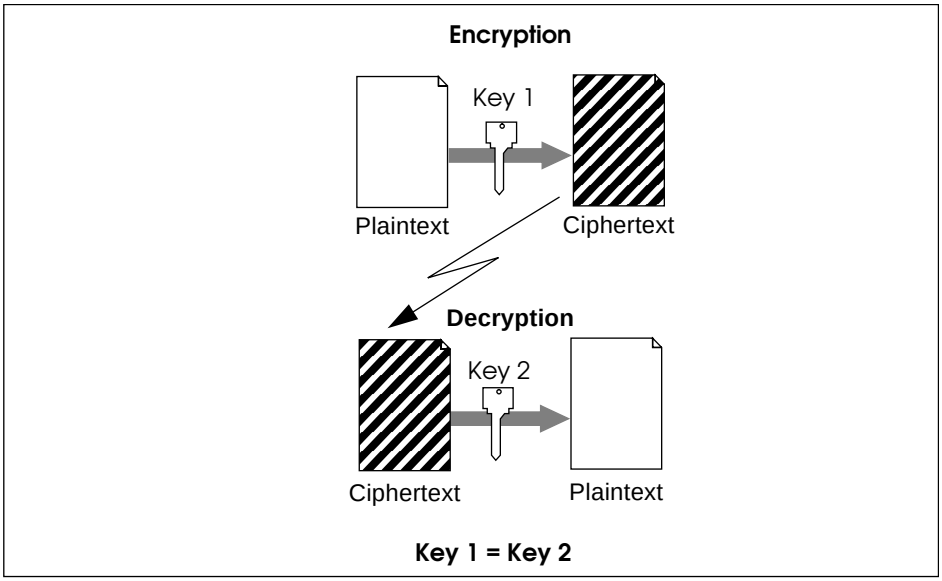

Figure 2 Shared Key Cryptography

Shared key cryptography presents the users with the substantial problem of managing the shared secret keys. They must either ship them in physical form or encrypt them in other keys that have been shipped in physical form. Physical shipment of keys is expensive and inflexible and represents a substantial limitation on the use of shared key cryptosystems.

A contemporary approach to this problem is called *public key cryptography* or sometimes *asymmetric cryptography*. In a public key cryptosystem a message can be encrypted in one key and decrypted in another. The keys are mathematically related in such a way that a knowledge of one key does not make it possible to figure out the other key. This permits one key, the public key, to be made widely known, while the corresponding private key is known only to a single user. The unique mathematical relationship between these two keys means that the private key can decrypt messages encoded with the public key and the public key can decrypt messages encoded with the private key.

Public key cryptography can be used for both privacy and authentication, as shown in Figure 3. The best-known example of a public key cryptosystem is RSA.

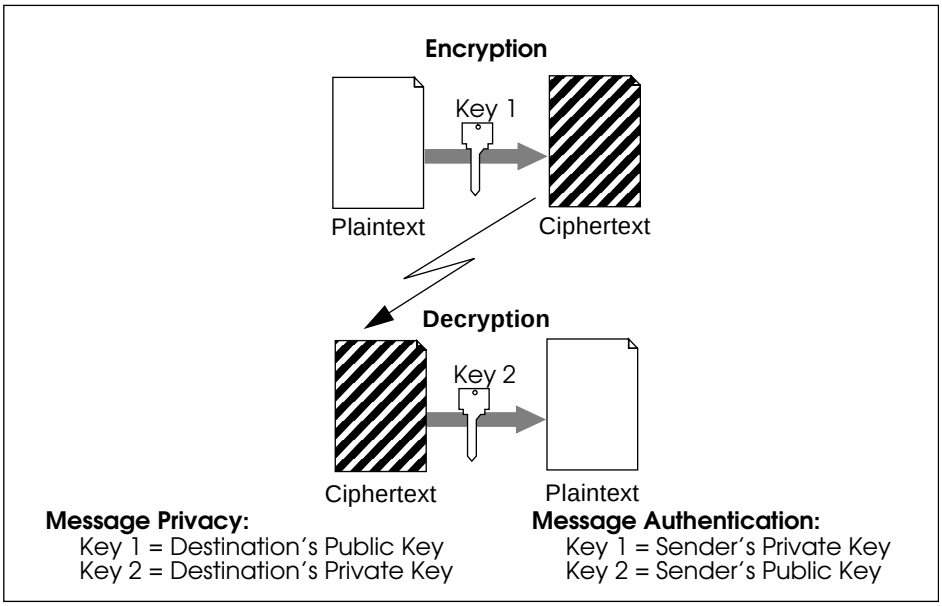

Figure 3 Public Key Cryptography

The rest of this section is devoted to describing shared key and public key systems in more detail, and the problems with some well-known existing systems.

Shared Key Cryptography

The best-known shared key cryptosystem is the Data Encryption Standard (DES), which was designed by IBM in response to a National Bureau of Standards (now the National Institute of Standards and Technology) request in the early 1970s. In its basic mode, it encrypts 64-bit blocks of plaintext, under 56 bits of key, using 16 iterations of an elaborate combination of table lookups and bit rearrangements.

DES was adopted as a standard by the US government in 1977, and has been re-certified every 5 years, most recently in 1993. It is widely felt, however, that advances in computing power together with the limitations of DES's 56 bit key (about a billion billion possibilities) are bringing DES to the end of its useful life. It seems unlikely to be re-certified in 1998. Fortunately, this fault can be remedied by encrypting each block three times using three different keys at a cost slightly less than a factor of three in speed. Although yet to be embraced by the Federal Government, this triple-DES system is fast emerging as a de-facto commercial standard.

For data encryption purposes, DES is usually used in a cipher block chaining mode (CBC) in which the ciphertext is tied together by XORing each 64-bit plaintext block with the previous ciphertext block before it is encrypted. This mode can as readily be used with triple-DES as with single.

The important advantage of shared key cryptosystems over any known public key system is that they can be made much faster. This is a major factor when encrypting large bodies of plaintext and mandates their use for this application despite the key distribution problems described above.

An example of a system that depends exclusively on shared-key cryptography is Kerberos, a network access control system developed at MIT. Kerberos relies on central storage of shared secret keys. This requires all network elements to trust a common set of servers and limits Kerberos application to networks in which such trust exists. It would not be suitable for a network connecting the Fortune 500, for example. The use of conventional keys also requires the participation of the trusted key server in each user login. If the server goes down for any length of time, network business comes to an almost compete halt.

Fortunately, a judicious combination of shared key cryptography and public key cryptography can achieve the performance of the former with the flexibility of the latter.

Public Key Cryptography

Public Key cryptography was invented in 1976 by Whitfield Diffie and Martin Hellman at Stanford University. As noted earlier, it bypasses the problem of arranging shared secret keys by using two mathematically complementary keys, one known to everyone, and one known only to its owner. The key that is known to everyone is called the *public key*, and the key that is known only to its owner of the key is called the *private key*. The two keys together are called a *key pair*. Public key cryptography solves the problem of securely distributing keys by removing the need to communicate secret keys. Only the public keys need be transmitted. These need not be kept secret, but it is very important to guarantee their authenticity and integrity.

If a message is encrypted with the public key, it must be decrypted with the private key. This is how privacy is achieved. If a message is encrypted with the private key, it must be decrypted with the public key. This is how authentication is achieved.

The most popular example of a public key system was developed at MIT in 1977 by Ron Rivest, Adi Shamir, and Len Adleman. It is known by their initials as RSA. Another system, called Diffie-Hellman after its inventors, produces shared secret keys directly from public `components' that are not in themselves keys. The two systems have complementary advantages and are often used in combination.

The construction of public key cryptosystems is mathematically more complex than the construction of conventional systems and has relied on the use of mathematical problems that have resisted solution over a long period of time.

Examples of such problems are factoring of large numbers into primes (used in RSA) and taking logarithms over finite fields (used in Diffie-Hellman). Both of these problems can be used to create one-way functions: functions that are much easier to compute in one direction than in the other.

The RSA system makes use of the fact that raising numbers to powers in modular arithmetic is easy. The enciphering equation in RSA is

$$c = m^e \ mod \ n.$$

(c, the ciphertext, is the remainder after m, the message, is raised to the power e, the enciphering exponent, and divided by n, the modulus.)

This can be done quickly by any computer that knows the message m, the modulus n, and the enciphering exponent e. Reversing this process, on the other hand, requires extracting p^{th} roots modulo n. This is extremely difficult for anyone who does not know the factors of n.

The RSA system lends itself very well to creating digital signatures and certain encryption applications. It is often desirable, however, to create a shared secret key for use in a shared-key cryptosystem directly. This uses another one-way function.

It is fairly easy to compute a y for the following equation, even if all the numbers are hundreds of digits long:

$$y = g^x \ mod \ p$$

(y is the remainder after g is raised to the power of x and then divided by p)

if you know g, x, and p. However, if you know y, g, and p, it is difficult to compute x in any reasonable amount of time, with numbers of comparable size. In this case when all the numbers are very large, and p is a very large prime number. In this case, the number y is call the public component and x is called the private component. The components are not keys as such, but we will show later how they can be used to create shared secret keys.

It is worth reiterating that the advantage of a public key system is that the secret components do not have to be shared in order to exchange information securely. The private portion is never given out to anyone, and it cannot feasibly be calculated from the public portion. There remains, however, the problem of making the public information available to the users who need it in such a way that they can be sure it is authentic.

Current Methods of Secure Transmission

Currently, the best way to transmit information securely in large scale networks with a diversity of subscribers is to use a combination of public key and shared key systems, that exploits the advantages of each method. An example follows. In this example, the message is being encrypted for privacy, so the destination's public key is being used to encrypt the DES key.

1. The first step is to encrypt the plaintext using a fast shared-key encryption mechanism such as DES, as shown in Figure 4.

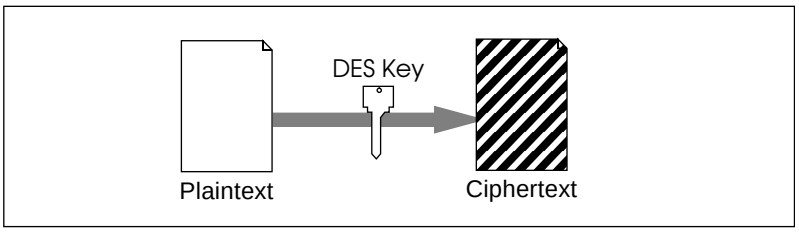

Figure 4 Encrypting Plaintext

2. The session key is then encrypted using the public part of the destination's key-pair, as shown in Figure 5.

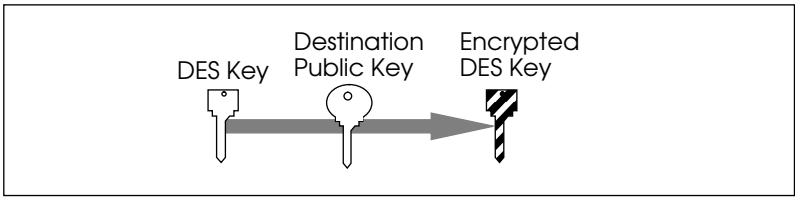

Figure 5 Encrypting the DES Key

3. Finally, the session-key encrypted plaintext and the public-key encrypted session-key are combined and sent to the destination, as shown in Figure 6.

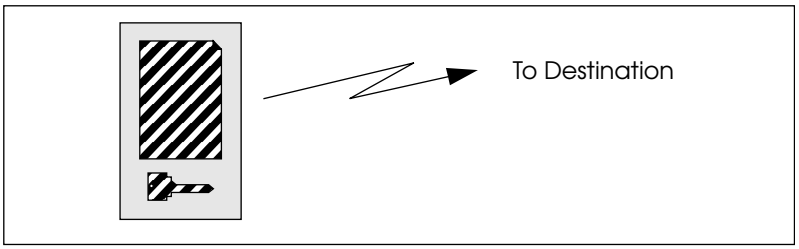

Figure 6 Sending the Secure Transmission

When the destination receives the message, it decrypt the session key using its own private key, then uses the session key to decrypt the remainder of the message.

This combination of shared and public key cryptographic methods avoids the problem of establishing a shared key outside the message by encrypting the relatively short shared key using the destination's public key. Also, since the body of the plaintext is encrypted using the faster shared key algorithm, the total time to create a ciphertext message is less than if the total plaintext was to be encoded using the public key.

Authentication

There are two distinct aspects to authentication: ensuring the integrity of the message and ensuring the identity of the sender.

A message is authenticated to show that it has not been tampered with while being transmitted (e.g., the contract has not been altered). The sender is authenticated to show that they are who they say they are (e.g., the contract was signed by the appropriate vice president).

Authenticating a Message

Conceptually, a message is signed by encrypting it with a private key and the signature is verified by decrypting it with the corresponding public key. There are, however, disadvantages to doing things exactly this way. Recall that public key systems are slow by comparison with their conventional counterparts. Encrypting a long message with a public key system for the purpose of signing it is no more attractive than encrypting a long message with a public key system for the purpose of keeping it secret.

The solution is to introduce another kind of conventional cryptographic mechanism called a *message digest* or *hash function*. A message digest algorithm produces a fixed-length digest from a message of arbitrary size, with the property that there is no known way of finding two messages with the same digest. This means that the digest, although typically much smaller than the message can be regarded as equivalent to the full messages for many purposes. The most common message digest algorithm is called MD5 and produces a digest 128 bits long.

Using message digests, signing a message goes as follows.

1. The user creates a message digest of the message (Figure 7).

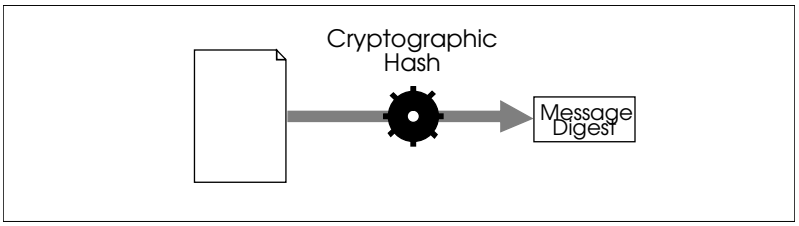

Figure 7 Creating a Message Digest

2. The message digest is then encrypted with the sender's private key (Figure 8).

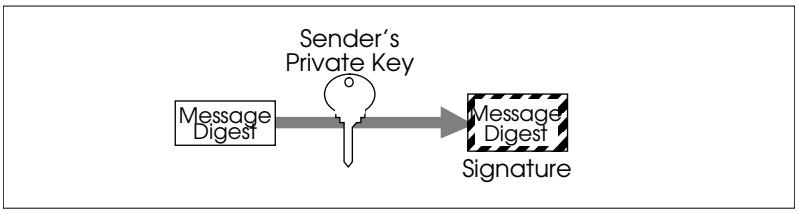

Figure 8 Encrypting the Message Digest

3. The original message and the encrypted message digest are sent to the destination (Figure 9).

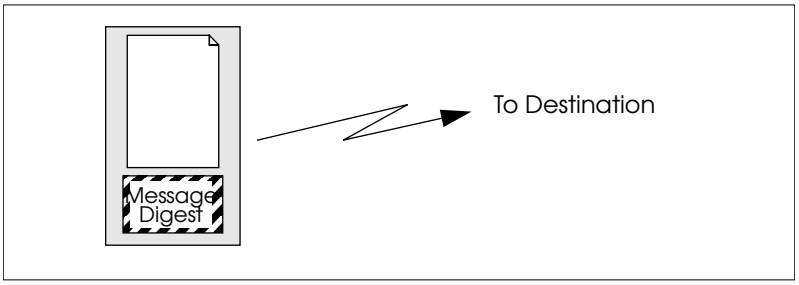

Figure 9 Final Configuration for Authenticated Message or Digital Signature

4. The destination receives the message and creates its own message digest of the message, using the same message digest function as the originator (top part of Figure 10).

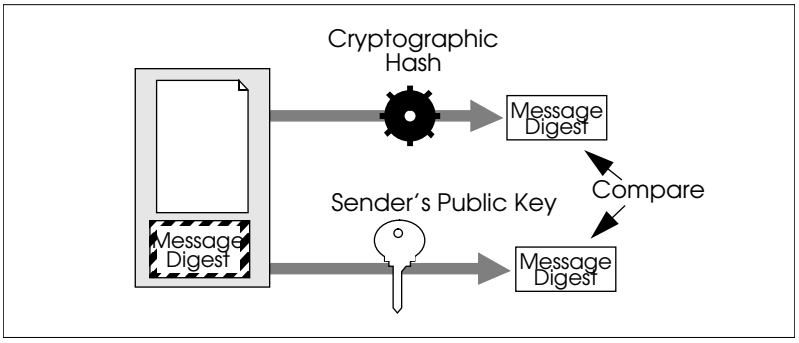

Figure 10 Authenticating a Message

5. The destination also decrypts the message digest it received (bottom part of Figure 10).

6. The destination now compares the message digest just created with the one that came with the message. If they match, the destination knows that the text of the message is the same as the user sent. If they do not match, the destination knows that the original message has been altered.

This process has another virtue, the one that gives it the name *digital signature*. Because only the user knows the private key, only the user could have created the encrypted message digest. Any destination that can obtain the public key can assure itself of the identity of the signer. This technique is used in the most popular programs for protecting electronic mail including PGP (Pretty Good Privacy) and PEM (Privacy Enhanced Mail).

User Authentication and CAs

But, how do we know that the key pair being used in the transactions in the previous section is actually the key pair for that user? We need a way of authenticating the relationship between the user and the public key.

We solve this problem by introducing a special sort of signed message called a *certificate* or *credentials*. A certificate contains information identifying the user: distinguished name, public key and expiration date, for example, digitally signed by a trusted network entity called a certifying authority (CA). It works as follows:

1. The user generates a key pair, and presents the public part of the key pair, along with other identifying information, to the CA.
2. The CA, once satisfied of identity of the user (person, institution, or host), takes the user's public key and creates a message digest of it. (Figure 11)

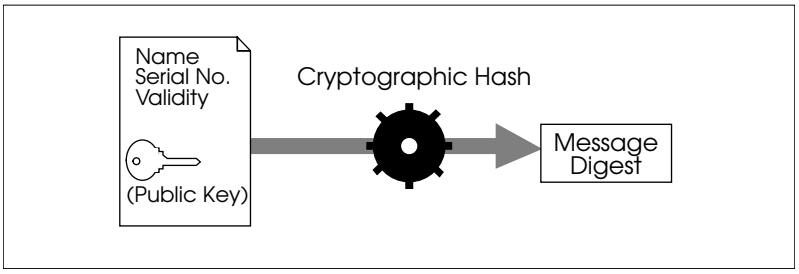

Figure 11 Creating a Message Digest of a Public Key

3. The message digest is then encrypted with the CA's private key, creating the CAs signature of the user's public key. (Figure 12)

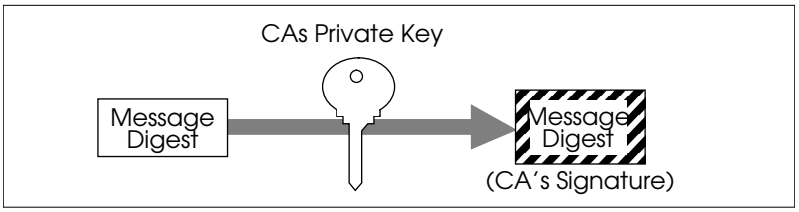

Figure 12 Creating the CAs Signature

4. The combination of the user's public key and the CA's signature authenticating the user's public key creates the certificate (Figure 13).

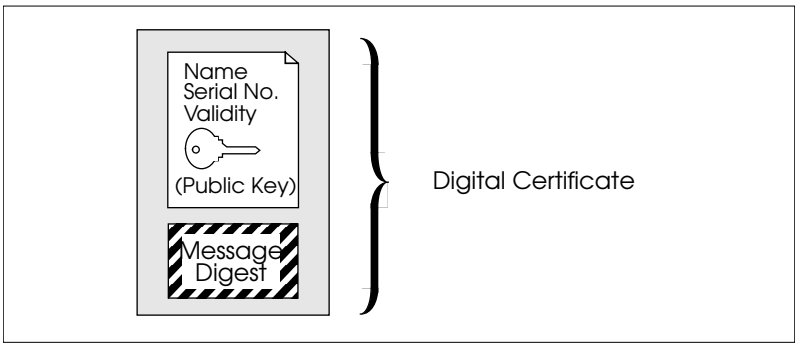

Figure 13 Digital Certificate

The CAs public key must be known to every user of the network. This permits anyone wishing to authenticate a certificate to follow the same procedure for authenticating a message described above and shown in Figure 10. The CA's public key is available in a certificate so that it, too, can be verified.

CA Structure

In the section above, we mentioned a Certifying Authority, or CA, which issues and manages the certificates that officially associate a public key to a user. Certificates are only good for a certain length of time, and the CA keeps a list of valid certificates, and their expiration dates. On occasion, a certificate may have to be revoked early, and the CA keeps a list of the revoked certificates as well as the valid ones. The CA makes its lists of valid and revoked or expired certificates available to anyone who asks for them.

There are two common approaches to certification: a hierarchical system of CAs and "web of trust." In a certification hierarchy, the top, or *root* CA authenticates the CAs below it. The second-level CAs would then authenticate the users and CAs below them, etc. In a *web of trust*, a user's public key can be presented in the form of a certificate signed by any user known to the person receiving it. One user, in seeking another's public key, can obtain it from a number of different sources and verify that they all agree.

The major commercial application for CAs is to authenticate businesses and the employees of those business. In the scenario at the beginning of this paper, a certificate would be issued to the employee of C. S. & P. by the C. S. & P. certifying authority. This certificate attests that the employee represented by the certificate is an employee of C. S. & P., and has a particular public key. The message may also contain the certificate for the C. S. & P. certifying authority, issued by a root CA. Figure 14 shows a a hierarchical CA structure.

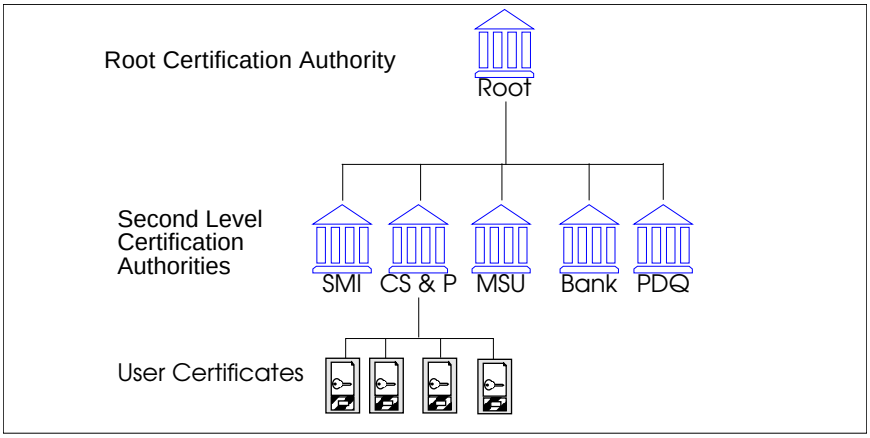

Figure 14 Certification Hierarchy

Simple Key-Management for IP (SKIP)

Although most of the privacy and authentication schemes in use today are designed to be used with session oriented protocols, the common protocols at the network layer are sessionless datagram protocols. IP and IPng (IP next generation) are two examples. Session oriented key management schemes can be adapted for use at the network layer, but it is better to employ a privacy and authentication scheme adapted to the needs sessionless datagram protocols. The Simple Key-Management for Internet Protocols (SKIP) scheme is one such scheme.

SKIP is application-independent, and can be used with many applications, such as FTP, Mosaic, and Telnet. SKIP was developed by Ashar Aziz at Sun Microsystems, Inc., and it has been proposed to the Internet Engineering Task Force (IETF) as a standard. It uses the principles of Diffie-Hellman Key Exchange to generate unique keys that can only be used by the source and destination nodes.

The remainder of this section describes how SKIP works and how it can be used as an integral part of a network access control system in both point-to-point and multicast situations.

How SKIP Works

SKIP uses the knowledge of its own private component and the destination's public component to calculate a unique key that can only be used between them. Each side's public component can be defined as $g^x \ mod \ p$, where x is the private component. In this system, g and p are fixed values known to both parties. The first node is called Node I. Node I has a public component K_i and a private component i. The second node is called Node J. Node J has a public component K_j and a private component j. Every node's public component is distributed in the form of a certificate. They are connected by an insecure network, as shown in Figure 15.

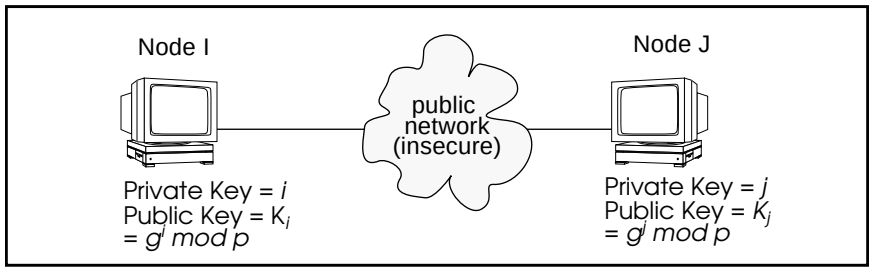

Figure 15 SKIP Over an Insecure Network

Because Node I knows its own private component and Node J's public component, it can use the two components to compute a unique key that only the two of them could know. It does this by raising Node J's public component to the power of its own private component:

$(K_j)^i$ or $(g^j \ mod \ p)^i$ which is equivalent to $g^{ji} \ mod \ p$.

Node J can do the same thing using its private component and Node I's public component.

$(K_i)^j$ or $(g^i \ mod \ p)^j$ which is equivalent to $g^{ij} \ mod \ p$.

Since g^{ij} is the same as g^{ji} we now have a unique key, K_{ij}, that can be calculated by both nodes independently, and only by these nodes. This is called the *pair-wise* key.

The pair-wise key is used to encrypt the packet key, which was used to encrypt the data or message. The packet key can be changed as often as desired or necessary, since keys may be vulnerable to cryptanalysis (determination of the key by opponents) if the intruder has a large enough sample of encrypted traffic. Each time the packet key is changed, the new key is encrypted in the pair-wise key. Any intruder trying to determine the pair-wise key in use between these nodes would have to solve the discrete log problem $y = g^x \ mod \ p$ for x, where y equals either K_j or K_i in order to retrieve the packet keys.

SKIP as Part of an Access Control System in a Point to Point Application

In order to minimize the impact of requiring key-management facilities on every node, packets can be encrypted using SKIP at an access control device (ACD), such as *SunScreen SPF-100*. *SunScreen™ SPF-100*, developed by Sun Microsystems' Internet Commerce Group, is part of a product line of dedicated devices incorporating SKIP at the network layer, enabling privacy and authentication over public internetworks using sessionless datagrams. Refer to the *Introduction to Network Security and SunScreen* technical white paper by the Internet Commerce Group for more details about the *SunScreen™* product line.

In the scenario shown in Figure 16, each IP packet destined for a node protected by a *SunScreen SPF-100* is encrypted as described in the previous section. The packet then gets a new header showing the remote *SunScreen* as the destination, and the encrypting *SunScreen* as the source. By doing this, the true network addresses of the source and destination nodes are hidden, thereby masking the true topology of the networks. When the destination *SunScreen* receives the encrypted packet, it decrypts the packet, recovers the final destination address, and sends the packet on in clear form.

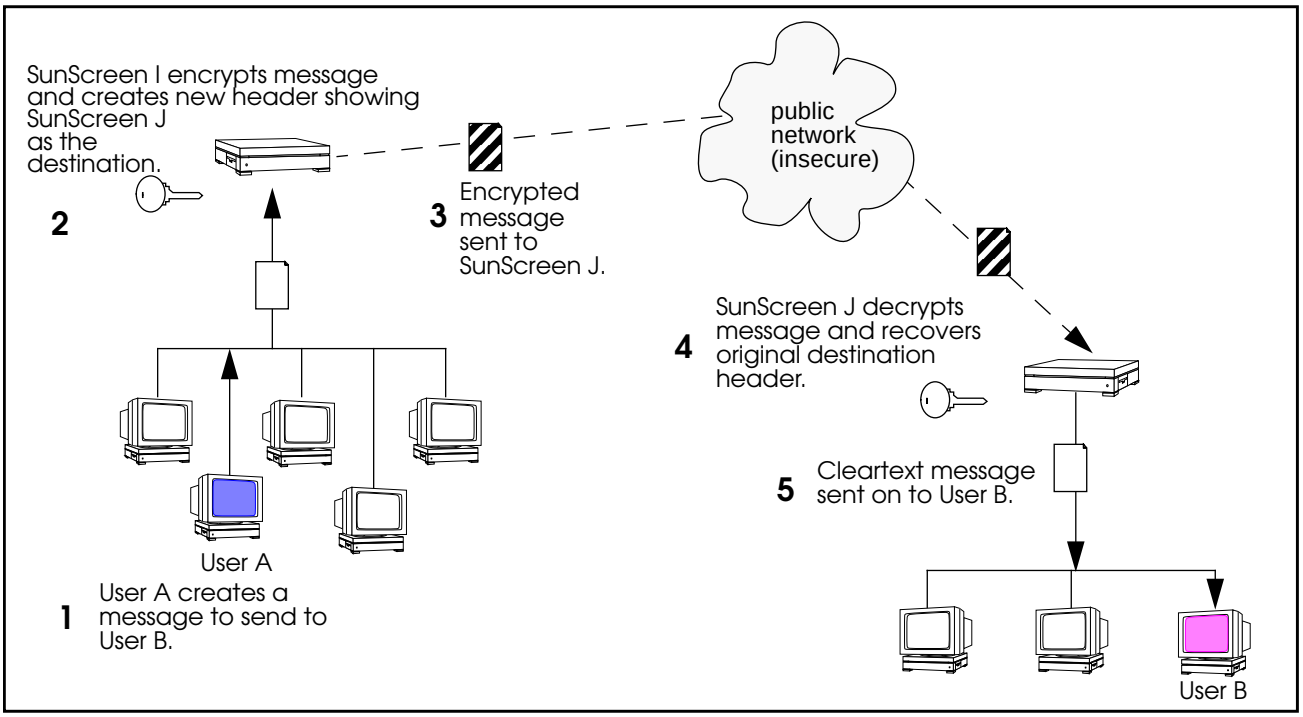

Figure 16 SKIP and SunScreen

Since only the *SunScreen*s need to participate in the key management protocol, there are a relatively small number of certificates that need to be managed.

SKIP and Host/Client

SKIP works equally well in a host/client scenario. When a user workstation (client) requests a service from a file server (host), it goes directly to the host with the request, using the same packet-encryption that is used in the SunScreen to SunScreen scenario. If the client workstation does not know the public key of the server, it requests it from the directory service. The workstation then uses the server's public key to create a pair-wise key, which it then uses to encrypt the packet key. Optionally, the client and server may obtain each other's certificates bilaterally, using a certificate exchange protocol.

The user workstation then sends its message directly to the file server. Upon receipt, the file server calculates the pair-wise key (if it is not already cached), and uses it to decrypt the packet key, which is then used to decrypt the packets. Figure 17 shows an overview of SKIP being used for client/server authentication and encryption. The directory service is updated periodically with new keys and revoked key lists.

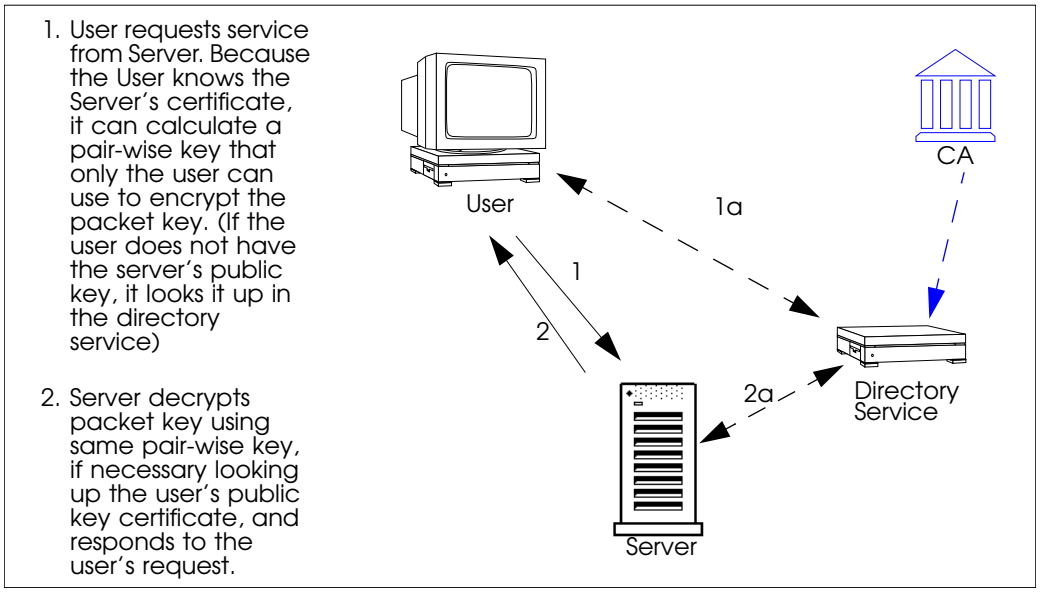

Figure 17 SKIP Client/Server Overview

SKIP and Multicast

In the example in the section describing how SKIP works, only two nodes are talking to each other. But what if one node needs to transmit to many nodes (multicast)? Must the message be encrypted with the pair-wise key for each node? No, because SKIP allows for multicast groups, so long as there is a key-management awareness in the establishment and joining of multicast groups.

First, the notion of a group owner is needed. The group owner knows the list of addresses that are potential members of the group. The group owner also has generated a symmetric shared key *G* that is known to all members of the

group. The group key *G* is distributed securely to nodes wishing to transmit to the group address M using the pair-wise protocol described in the previous section.

Refer to Figure 18. Nodes wishing to receive encrypted datagrams sent to multicast address M need to acquire the secret key *G*. This is done by each node that wants to receive encrypted datagrams sending a request to join message to the group owner. If the requesting node's IP address is part of the group's receive membership, the group key *G* is sent to the node in a secure pair-wise datagram as described in the previous section.

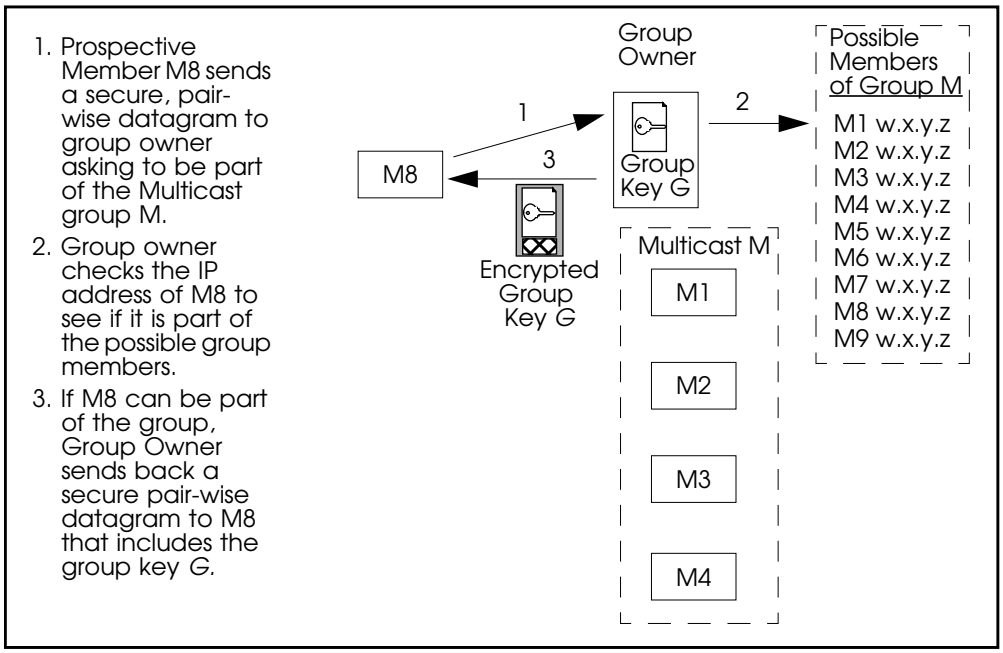

Figure 18 Joining a Multicast Group

In order for a member of the group to send a secure transmission to the multicast address M, it would need to generate a random packet key *k* and use it to encrypt the message. The packet key *k* is then encrypted with the group key *G* and sent to the multicast address M.

The advantage to this scheme is that every member of the group can change the packet key *k* as often as it needs to, without involving key-setup communications overhead involving every member of the group.

Summary

Returning to the problem posed in the opening paragraphs of this paper, we can now show that this problem is easily solved using the cryptographic algorithms which are a part of a *SunScreen*™ SPF-100 incorporating SKIP.

In the example, C.S. & P. would have a *SunScreen™* at each of their regional offices, and at the corporate office. The corporate office would send its information in the form of authenticated, encrypted network packets to each of the regional offices. Since the packets are encrypted, they cannot be read by a competitor, even over an insecure network. Because the regional offices can authenticate the packets, they know the packets came from the corporate headquarters, and have not been altered in transit. The encryption and authentication were handled at the network layer, so it was not necessary to establish a session between the regional offices and the corporate office in order to send and receive packets. Because both privacy and authentication were incorporated in a manner transparent to the application, no special actions needed to be taken in order to send or receive packets. Figure 19 shows this configuration. The person at the Atlanta office was on the road when the information was sent, but was able to receive the information on a laptop and read it using an end-node version of SKIP.

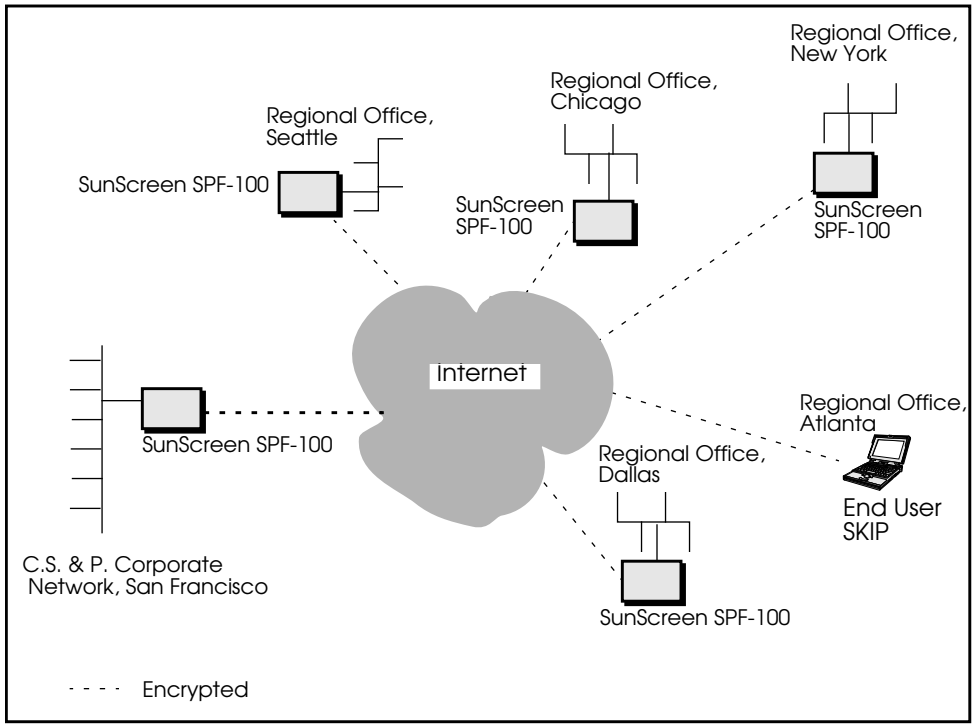

Figure 19 C. S. & P. Conducting Business over the Internet with SunScreen SPF-100.

References

Aziz, Ashar, *Simple Key Management for Internet Protocols, Working Draft.* 1994, Internet Commerce Group, Sun Microsystems Laboratories, a Sun Microsystems, Inc. Business.

Cheswick, William R., and Bellovin, Steven M., *Firewalls and Internet Security: Repelling the Wily Hacker*, 1994, Addison-Wesley Publishing Company.

Comer, Douglas E., *Internetworking with TCP/IP*, Volume I; Second Edition, 1991, Prentice Hall, Inc.

Diffie, Whitfield, and Hellman, Martin, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, IT-22:644--654, 1977

Fahn, Paul, *Answers to FREQUENTLY ASKED QUESTIONS About Today's Cryptography*, 1993, RSA Laboratories, a division of RSA Data Security, Inc.

Introduction to Network Security and SunScreen, 1994, Internet Commerce Group, Sun Microsystems Laboratories, a Sun Microsystems, Inc. Business.

Schiller, Jeffrey I., "Secure Distributed Computing", *Scientific American*, Volume 271, Number 5, November 1994.

Schneier, Bruce, *Applied Cryptography, Protocols, Algorithms, and Source Code in C*, 1994, John Wiley and Sons, Inc.

Stallings, William, *Network and Internetwork Security, Principles and Practice,* 1995, Prentice Hall, Inc.