87

An Introduction to Cryptography

87.1 The Basics 1122
What Is Cryptography? • Related Terms and Definitions • A Brief History • The Alphabet-Soup Players: Alice, Bob, Eve, and Mike • Ties to Confidentiality, Integrity, and Authentication • Section Summary

87.2 Symmetric Cryptographic Algorithms 1125
Stream Ciphers • Block Ciphers • DES • AES (Rijndael) • Weaknesses and Attacks • Section Summary

87.3 Asymmetric (Public Key) Cryptography 1129
Background • RSA • Elliptic Curve Cryptosystems (ECC) • Attacks • Real-World Applications • SSL and TLS

87.4 Choosing an Algorithm 1132
Export and International Use Issues • Section Summary

87.5 Key Management and Exchange 1133
Generation • Exchange • Installation and Storage • Change Control • Destruction • Examples and Implementations • Section Summary

87.6 Hash Functions 1135
Message Digests • MD2, MD4, and MD5 • SHA • Applications of Message Digests • Digital Signatures • Digital Signature Algorithm (DSA) • Message Authentication Codes (MACs) • Section Summary

87.7 Other Cryptographic Notes 1137
Steganography • Digital Notary Public • Backdoors and Digital Snake Oil

References 1138

Javek Ikbal

This chapter presents some basic ideas behind cryptography. This is intended for an audience evaluators, recommenders, and end users of cryptographic algorithms and products rather than implementers. Hence, the mathematical background will be kept to a minimum. Only widely adopted algorithms are described with some mathematical detail. We also present promising technologies and algorithms that information security practitioners might encounter and may have to choose or discard.

AU7495—Chapter87—25/1/2007—21:01—PARTHIBAN—14747—XML MODEL CRC12a – pp. 1121–1139.

87.1 The Basics

87.1.1 What Is Cryptography?

Cryptography is the art and science of securing messages so unintended audiences cannot read, understand, or alter that message.

87.1.2 Related Terms and Definitions

A message in its original form is called the plaintext or cleartext. The process of securing that message by hiding its contents is encryption or enciphering. An encrypted message is called ciphertext, and the process of turning the ciphertext back to cleartext is called decryption or deciphering. Cryptography is often shortened to crypto.

Practitioners of cryptography are known as cryptographers. The art and science of breaking encryptions is known as cryptanalysis, which is practiced by cryptanalysts. Cryptography and cryptanalysis are covered in the theoretical and applied branch of mathematics known as cryptology, and practiced by cryptologists.

A cipher or cryptographic algorithm is the mathematical function or formula used to convert cleartext to ciphertext and back. Typically, a pair of algorithms is used to encrypt and decrypt.

An algorithm that depends on keeping the algorithm secret to keep the ciphertext safe is known as a restricted algorithm. Security practitioners should be aware that restricted algorithms are inadequate in the current world. Unfortunately, restricted algorithms are quite popular in some settings. Exhibit 87.1 shows the schematic flow of restricted algorithms. This can be mathematically expressed as $E(M) = C$ and $D(C) = M$, where M is the cleartext message, E is the encryption function, C is the ciphertext, and D is the decryption function.

A major problem with restricted algorithms is that a changing group cannot use it; every time someone leaves, the algorithm has to change. Because of the need to keep it a secret, each group has to build its own algorithms and software to use it.

These shortcomings are overcome by using a variable known as the key or cryptovariable. The range of possible values for the key is called the keyspace. With each group using its own key, a common and well-known algorithm may be shared by any number of groups.

The mathematical representation now becomes: $E_k(M) = C$ and $D_k(C) = M$, where the subscript k refers to the encryption and decryption key. Some algorithms will utilize different keys for encryption and decryption. Exhibit 87.2 illustrates that the key is an input to the algorithm.

Note that the security of all such algorithms depends on the key and not the algorithm itself. We submit to the information security practitioner that any algorithm that has not been publicly discussed, analyzed, and withstood attacks (i.e., zero restriction) should be presumed insecure and rejected.

EXHIBIT 87.1 Encryption and decryption with restricted algorithms.

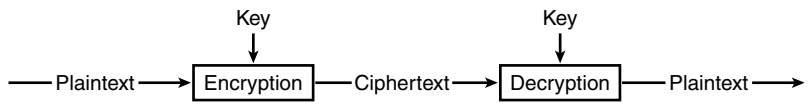

EXHIBIT 87.2 Encryption and decryption with keys.

87.1.3 A Brief History

Secret writing probably came right after writing was invented. The earliest known instance of cryptography occurred in ancient Egypt 4000 years ago, with the use of hieroglyphics. These were purposefully cryptic; hiding the text was probably not the main purpose—it was intended to impress. In ancient India, government spies communicated using secret codes. Greek literature has examples of cryptography going back to the time of Homer. Julius Caesar used a system of cryptography that shifted each letter three places further through the alphabet (e.g., A shifts to D, Z shifts to C, etc.). Regardless of the amount of shift, all such monoalphabetic substitution ciphers (MSCs) are also known as Caesar ciphers. While extremely easy to decipher if you know how, a Caesar cipher called ROT-13 (N = A, etc.) is still in use today as a trivial method of encryption. Why ROT-13 and not any other ROT-N? By shifting down the middle of the English alphabet, ROT-13 is self-reversing—the same code can be used to encrypt and decrypt. How this works is left as an exercise for the reader. Exhibit 87.3 shows the alphabet and corresponding Caesar cipher and ROT-13.

During the seventh century A.D., the first treatise on cryptanalysis appeared. The technique involves counting the frequency of each ciphertext letter. We know that the letter E occurs the most in English. So if we are trying to decrypt a document written in English where the letter H occurs the most, we can assume that H stands for E. Provided we have a large enough sample of the ciphertext for the frequency count to be statistically significant, this technique is powerful enough to cryptanalyze any MSC and is still in use.

Leon Battista Alberti invented a mechanical device during the 15th century that could perform a polyalphabetic substitution cipher (PSC). A PSC can be considered an improvement of the Caesar cipher because each letter is shifted by a different amount according to a predetermined rule.

The device consisted of two concentric copper disks with the alphabet around the edges. To start enciphering, a letter on the inner disk is lined up with any letter on the outer disk, which is written as the first character of the ciphertext. After a certain number of letters, the disks are rotated and the encryption continues. Because the cipher is changed often, frequency analysis becomes less effective.

The concept of rotating disks and changing ciphers within a message was a major milestone in cryptography.

The public interest in cryptography dramatically increased with the invention of the telegraph. People wanted the speed and convenience of the telegraph without disclosing the message to the operator, and cryptography provided the answer.

After World War I, U.S. military organizations poured resources into cryptography. Because of the classified nature of this research, there were no general publications that covered cryptography until the late 1960s; and the public interest went down again.

During this time, computers were also gaining ground in nongovernment areas, especially the financial sector; and the need for a nonmilitary crypto-system was becoming apparent. The organization currently known as the National Institute of Standards and Technology (NIST), then called the National Bureau of Standards (NBS), requested proposals for a standard cryptographic algorithm. IBM responded with Lucifer, a system developed by Horst Feistel and colleagues. After adopting two modifications from the National Security Agency (NSA), this was adopted as the federal Data Encryption Standard (DES) in 1976.[1] NSA's changes caused major controversy, specifically because it suggested DES use 56-bit keys instead of 112-bit keys as originally submitted by IBM.

During the 1970s and 1980s, the NSA also attempted to regulate cryptographic publications but was unsuccessful. However, general interest in cryptography increased as a result. Academic and business

English Alphabet	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Caesar Cipher (3)	D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
ROT-13	N O P Q R S T U V W X Y Z A B C D E F G H I J K L M

EXHIBIT 87.3 Caesar cipher (Shift-3) and ROT-13.

interest in cryptography was high, and extensive research led to significant new algorithms and techniques.

Advances in computing power have made 56-bit keys breakable. In 1998, a custom-built machine from the Electronic Frontier Foundation costing $210,000 cracked DES in four and a half days.[2] In January 1999, a distributed network of 100,000 machines cracked DES in 22 hours and 15 minutes.

As a direct result of these DES cracking examples, NIST issued a Request for Proposals to replace DES with a new standard called the Advanced Encryption Standard (AES).[3] On November 26, 2001, NIST selected Rijndael as the AES.

87.1.4 The Alphabet-Soup Players: Alice, Bob, Eve, and Mike

In our discussions of cryptographic protocols, we will use an alphabet soup of names that are participating in (or are trying to break into) a secure message exchange:

- *Alice*, first participant
- *Bob*, second participant
- *Eve*, eavesdropper
- *Mike*, masquerader

87.1.5 Ties to Confidentiality, Integrity, and Authentication

Cryptography is not limited to confidentiality only—it can perform other useful functions.

- *Authentication*. If Alice is buying something from Bob's online store, Bob has to assure Alice that it is indeed Bob's Web site and not Mike's, the masquerader pretending to be Bob. Thus, Alice should be able to authenticate Bob's Web site, or know that a message originated from Bob.
- *Integrity*. If Bob is sending Alice, the personnel manager, a message informing her of a $5000 severance pay for Mike, Mike should not be able to intercept the message in transit and change the amount to $50,000. Cryptography enables the receiver to verify that a message has not been modified in transit.
- *Non-repudiation*. Alice places an order to sell some stocks at $10 per share. Her stockbroker, Bob, executes the order, but then the stock goes up to $18. Now Alice claims she never placed that order. Cryptography (through digital signatures) will enable Bob to prove that Alice did send that message.

87.1.6 Section Summary

- Any message or data in its original form is called plaintext or cleartext.
- The process of hiding or securing the plaintext is called encryption (verb: to encrypt or to encipher).
- When encryption is applied on plaintext, the result is called ciphertext.
- Retrieving the plaintext from the ciphertext is called decryption (verb: to decrypt or to decipher).
- The art and science of encryption and decryption is called cryptography, and its practitioners are cryptographers.
- The art and science of breaking encryption is called cryptanalysis, and its practitioners are cryptanalysts.
- The process and rules (mathematical or otherwise) to encrypt and decrypt are called ciphers or cryptographic algorithms.
- The history of cryptography is over 4000 years old.

- Frequency analysis is an important technique in cryptanalysis.
- Secret cryptographic algorithms should not be trusted by an information security professional.
- Only publicly available and discussed algorithms that have withstood analysis and attacks may be used in a business setting.
- Bottom line: do not use a cryptographic algorithm developed in-house (unless you have internationally renowned experts in that field).

87.2 Symmetric Cryptographic Algorithms

Algorithms or ciphers that use the same key to encrypt and decrypt are called symmetric cryptographic algorithms. There are two basic types: stream and block.

87.2.1 Stream Ciphers

This type of cipher takes messages in a stream and operates on individual data elements (characters, bits, or bytes).

Typically, a random-number generator is used to produce a sequence of characters called a key stream. The key stream is then combined with the plaintext via exclusive-OR (XOR) to produce the ciphertext. Exhibit 87.4 illustrates this operation of encrypting the letter Z, the ASCII value of which is represented in binary as 01011010. Note that in an XOR operation involving binary digits, only XORing 0 and 1 yields 1; all other XORs result in 0. Exhibit 87.4 shows how a stream cipher operates.

Before describing the actual workings of a stream cipher, we will examine how shift registers work because they have been the mainstay of electronic cryptography for a long time.

A linear feedback shift register (LFSR) is very simple in principle. For readers not versed in electronics, we present a layman's representation. Imagine a tube that can hold four bits with a window at the right end. Because the tube holds four bits, we will call it a four-bit shift register. We shift all bits in the tube and, as a result, the bit showing through the window changes. Here, shifting involves pushing from the left so the right-most bit falls off; and to keep the number of bits in the tube constant, we place the output of some addition operation as the new left-most bit. In the following example, we will continue with our four-bit LFSR, and the new left-most bit will be the result of adding bits three and four (the feedback) and keeping the right-most bit (note that in binary mathematics, $1+1=10$, with 0 being the right-most bit, and $1+0=1$). For every shift that occurs, we look through the window and note the right-most bit. As a result, we will see the sequence shown in Exhibit 87.5.

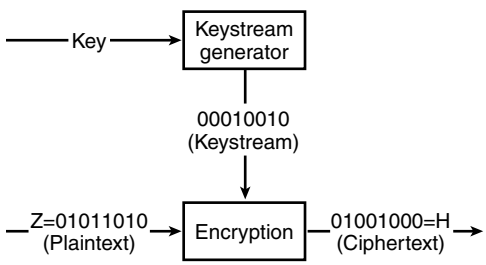

EXHIBIT 87.4 Stream cipher operation.

1111.-> 0111 -> 0011 -> 0001 -> 1000 -> 0100 -> 0010 -> 1001 -> 1100 -> 0110 -> 1011 -> 0101 -> 1010 -> 1101 -> 1101 -> 1110 -> 1111
Keystream: 111100010011010 (Right-most bit through the window before repetition).

EXHIBIT 87.5 4-bit LFSR output.

Note that after $2^{(N=4)}-1=15$ iterations, we will get a repetition. This is the maximum number of unique sequences (also called period) when dealing with a four-bit LFSR (because we have to exclude 0000, which will always produce a sequence of 0000s). Choosing a different feedback function may have reduced the period, and the longest unique sequence is called the maximal length. The maximal length is important because repeating key streams mean the same plaintext will produce the same ciphertext, and this will be vulnerable to frequency analysis and other attacks.

To construct a simple stream cipher, take an LFSR (or take many different sizes and different feedback functions). To encrypt each bit of the plaintext, take a bit from the plaintext, XOR it with a bit from the key stream to generate the ciphertext (refer to Exhibit 87.4), and so on.

Of course, other stream ciphers are more complex and involve multiple LFSRs and other techniques.[4] We will discuss RC4 as an example of a stream cipher. First, we will define the term S-box.

An S-box is also known as a substitution box or table and, as the name implies, it is a table or system that provides a substitution scheme. Shift registers are S-boxes; they provide a substitution mechanism.

RC4 uses an output feedback mechanism combined with 256 S-boxes (numbered S0…S255) and two counters, i and j.

A random byte K is generated through the following steps:

$i = (i+1) \bmod 256$
$j = (j + S_i) \bmod 256$
swap (S_i, S_j)
$t = (S_i + S_j) \bmod 256$
$K = S_t$

Now, K XOR Plaintext = Ciphertext, and K XOR Ciphertext = Plaintext

87.2.2 Block Ciphers

A block cipher requires the accumulation of some amount of data or multiple data elements before ciphering can begin. Encryption and decryption happen on chunks of data, unlike stream ciphers, which operate on each character or bit independently.

87.2.3 DES

The Data Encryption Standard (DES) is over 25 years old; because of its widespread implementation and use, it will probably coexist with the new Advanced Encryption Standard (AES) for a few years.

Despite initial concern about NSA's role in crafting the standard, DES generated huge interest in cryptography; vendors and users alike were eager to adopt the first government-approved encryption standard that was released for public use.

The DES calls for reevaluations of DES every five years. Starting in 1987, the NSA warned that it would not recertify DES because it was likely that it soon would be broken; they proposed secret algorithms available on tamper-proof chips only. Users of DES, including major financial institutions, protested; DES got a new lease on life until 1992. Because no new standards became available in 1992, it lived on to 1998 and then until the end of 2001, when AES became the standard.

DES is a symmetric block cipher that operates in blocks of 64 bits of data at a time, with 64-bit plaintext resulting in 64-bit ciphertext. If the data is not a multiple of 64 bits, then it is padded at the end. The effective key-length is 56 bits with 8 bits of parity. All security rests with the key.

A simple description of DES is as follows:[1]

Take the 64-bit block of message (M).
Rearrange the bits of M (initial permutation, IP).
Break IP down the middle into two 32-bit blocks (L & R).
Shift the key bits, and take a 48-bit portion from the key.
Save the value of R into R_{old}.

Expand R via a permutation to 48 bits.
XOR R with the 48-bit key and transform via eight S-boxes into a new 32-bit chunk.
Now, R takes on the value of the new R XOR-ed with L.
And L takes on the value of R_{old}.
Repeat this process 15 more times (total 16 rounds).
Join L and R.
Reverse the permutation IP (final permutation, FP).

There are some implementations without IP and FP; because they do not match the published standard, they should not be called DES or DES-compliant, although they offer the same degree of security.

Certain DES keys are considered weak, semiweak, or possibly weak: a key is considered weak if it consists of all 1s or all 0s, or if half the keys are 1s and the other half are 0s.[5]

Conspiracy theories involving NSA backdoors and EFFs DES-cracking machine notwithstanding, DES lives on in its original form or a multiple-iteration form popularly known as Triple-DES.

Triple-DES is DES done thrice, typically with two 56-bit keys. In the most popular form, the first key is used to DES-encrypt the message. The second key is used to DES-decrypt the encrypted message. Because this is not the right key, the attempted decryption only scrambles the data even more. The resultant ciphertext is then encrypted again with the first key to yield the final ciphertext. This three-step procedure is called Triple-DES. Sometimes, three keys are used.

Because this follows an Encryption > Decryption > Encryption scheme, it is often known as DES-EDE.

ANSI standard X9.52 describes Triple-DES encryption with keys k_1, k_2, k_3 as:

$$C = E_{k3}(D_{k2}(E_{k1}(M)))$$

where E_k and D_k denote DES encryption and DES decryption, respectively, with the key k. Another variant is DES-EEE, which consists of three consecutive encryptions. There are three keying options defined in ANSI X9.52 for DES-EDE:

The three keys k_1, k_2, and k_3 are different (three keys).
k_1 and k_2 are different, but $k_1 = k_3$ (two keys).
$k_1 = k_2 = k_3$ (one key).
The third option makes Triple-DES backward-compatible with DES and offers no additional security.

87.2.4 AES (Rijndael)

In 1997, NIST issued a Request for Proposals to select a symmetric-key encryption algorithm to be used to protect sensitive (unclassified) federal information. This was to become the Advanced Encryption Standard (AES), the DES replacement. In 1998, NIST announced the acceptance of 15 candidate algorithms and requested the assistance of the cryptographic research community in analyzing the candidates. This analysis included an initial examination of the security and efficiency characteristics for each algorithm.

NIST reviewed the results of this preliminary research and selected MARS, RC6™, Rijndael, Serpent, and Twofish as finalists. After additional review, in October 2000, NIST proposed Rijndael as AES. For research results and rationale for selection, see Reference 5.

Before discussing AES, we will quote the most important answer from the Rijndael FAQ:

> If you're Dutch, Flemish, Indonesian, Surinamer or South African, it's pronounced like you think it should be. Otherwise, you could pronounce it like reign dahl, rain doll, or rhine dahl. We're not picky. As long as you make it sound different from region deal.[6]

Rijndael is a block cipher that can process blocks of 128-, 192-, and 256-bit length using keys 128-, 192-, and 256-bits long. All nine combinations of block and key lengths are possible.[7] The AES standard specifies only 128-bit data blocks and 128-, 192-, and 256-bit key lengths. Our discussions will be confined to AES and not the full scope of Rijndael. Based on the key length, AES may be referred to as AES-128, AES-192, or AES-256. We will present a simple description of Rijndael. For a mathematical treatment, see References 8 and 9.

Rijndael involves an initial XOR of the state and a round key, nine rounds of transformations (or rounds), and a round performed at the end with one step omitted. The input to each round is called the state. Each round consists of four transformations: SubBytes, ShiftRow, MixColumn (omitted from the tenth round), and AddRoundKey.

In the SubBytes transformation, each of the state bytes is independently transformed using a nonlinear S-box.

In the ShiftRow transformation, the state is processed by cyclically shifting the last three rows of the state by different offsets.

In the MixColumn transformation, data from all of the columns of the state are mixed (independently of one another) to produce new columns.

In the AddRoundKey step in the cipher and inverse cipher transformations, a round key is added to the state using an XOR operation. The length of a round key equals the size of the state.

87.2.5 Weaknesses and Attacks

A well-known and frequently used encryption is the stream cipher available with PKZIP. Unfortunately, there is also a well-known attack involving known plaintext against this—if you know part of the plaintext, it is possible to decipher the file.[10] For any serious work, information security professionals should not use PKZIP's encryption.

In 1975, it was theorized that a customized DES cracker would cost $20 million. In 1998, EFF built one for $220,000.[2] With the advances in computing power, the time and money required to crack DES has significantly gone down even more. Although it is still being used, if possible, use AES or Triple-DES.

87.2.6 Section Summary

- Symmetric cryptographic algorithms or ciphers are those that use the same key to encrypt and decrypt.
- Stream ciphers operate one bit at a time.
- Stream ciphers use a key stream generator to continuously produce a key stream that is used to encrypt the message.
- A repeating key stream weakens the encryption and makes it vulnerable to cryptanalysis.
- Shift registers are often used in stream ciphers.
- Block ciphers operate on a block of data at a time.
- DES is the most popular block cipher.
- DES keys are sometimes referred to as 64-bit, but the effective length is 56 bits with 8 parity bits; hence, the actual key length is 56 bits.
- There are known weak DES keys; ensure that those are not used.
- DES itself has been broken and it should be assumed that it is not secure against attack.
- Make plans to migrate away from DES; use Triple-DES or Rijndael instead of DES, if possible.
- Do not use the encryption offered by PKZIP for nontrivial work.

87.3 Asymmetric (Public Key) Cryptography

Asymmetric is the term applied in a cryptographic system where one key is used to encrypt and another is used to decrypt.

87.3.1 Background

This concept was invented in 1976 by Whitfield Diffie and Martin Hellman[11] and independently by Ralph Merkle. The basic theory is quite simple: is there a pair of keys so that if one is used to encrypt, the other can be used to decrypt—and given one key, finding the other would be extremely hard?

Luckily for us, the answer is yes, and this is the basis of asymmetric (often called public key) cryptography.

There are many algorithms available, but most of them are either insecure or produce ciphertext that is larger than the plaintext. Of the algorithms that are both secure and efficient, only three can be used for both encryption and digital signatures.[4] Unfortunately, these algorithms are often slower by a factor of 1000 compared to symmetric key encryption.

As a result, hybrid cryptographic systems are popular: Suppose Alice and Bob want to exchange a large message. Alice generates a random session key, encrypts it using asymmetric encryption, and sends it over to Bob, who has the other half of the asymmetric key to decode the session key. Because the session key is small, the overhead to asymmetrically encipher/decipher it is not too large. Now Alice encrypts the message with the session key and sends it over to Bob. Bob already has the session key and deciphers the message with it. As the large message is enciphered/deciphered using much faster symmetric encryption, the performance is acceptable.

87.3.2 RSA

We will present a discussion of the most popular of the asymmetric algorithms—RSA, named after its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman. Readers are directed to Reference[12] for an extensive treatment. RSA's patent expired in September 2000; and RSA has put the algorithm in the public domain, enabling anyone to implement it at zero cost.

First, a mathematics refresher:

- If an integer P cannot be divided (without remainders) by any number other than itself and 1, then P is called a prime number. Other prime numbers are 2, 3, 5, and 7.
- Two integers are relatively prime if there is no integer greater than one that divides them both (their greatest common divisor is 1). For example, 15 and 16 are relatively prime, but 12 and 14 are not.
- The mod is defined as the remainder. For example, 5 mod 3 = 2 means divide 5 by 3 and the result is the remainder, 2.

Note that RSA depends on the difficulty of factoring large prime numbers. If there is a sudden leap in computer technology or mathematics that changes that, security of such encryption schemes will be broken. Quantum and DNA computing are two fields to watch in this arena.

Here is a step-by-step description of RSA:

1. Find P and Q, two large (e.g., 1024-bit or larger) prime numbers. For our example, we will use P = 11 and Q = 19, which are adequate for this example (and more manageable).
2. Calculate the product PQ, and also the product $(P-1)(Q-1)$. So PQ = 209, and $(P-1)(Q-1) = 180$.
3. Choose an odd integer E such that E is less than PQ, and such that E and $(P-1)(Q-1)$ are relatively prime. We will pick E = 7.
4. Find the integer D so that $(DE-1)$ is evenly divisible by $(P-1)(Q-1)$. D is called the multiplicative inverse of E. This is easy to do: let us assume that the result of evenly dividing

(DE−1) by (P−1)(Q−1) is X, where X is also an integer. So we have X=(DE−1)/(P−1)(Q−1); and solving for D, we get D=(X(P−1)(Q−1)+1)/E. Start with X=1 and keep increasing its value until D is an integer. For our example, D works out to be 103.

5. The public key is (E and PQ), the private key is D. Destroy P and Q (note that given P and Q, it would be easy to work out E and D; but given only PQ and E, it would be hard to determine D). Give out your public key (E, PQ) and keep D secure and private.
6. To encrypt a message M, we raise M to the Eth power, divide it by PQ, and the remainder (the mod) is the ciphertext. Note that M must be less than PQ. A mathematical representation will be ciphertext = ME mod PQ. So if we are encrypting 13 (M=13), our ciphertext = 13^7 mod 209 = 29.
7. To decrypt, we take the ciphertext, raise it to the Dth power, and take the mod with PQ. So plaintext = 29^{103} mod 209 = 13.

Compared to DES, RSA is about 100 times slower in software and 1000 times slower in hardware. Because AES is even faster than DES in software, the performance gap will widen in software-only applications.

87.3.3 Elliptic Curve Cryptosystems (ECC)

As we saw, solving RSA depends on a hard math problem: factoring very large numbers. There is another hard math problem: reversing exponentiation (logarithms). For example, it is possible to easily raise 7 to the 4th power and get 2401; but given only 2401, reversing the process and obtaining 7^4 is more difficult (at least as hard as performing large factorizations).

The difficulty in performing discrete logarithms over elliptic curves (not to be confused with an ellipse) is even greater;[13] and for the same key size, it presents a more difficult challenge than RSA (or presents the same difficulty/security with a smaller key size). There is an implementation of ECC that uses the factorization problem, but it offers no practical advantage over RSA.

An elliptic curve has an interesting property: it is possible to define a point on the curve as the sum of two other points on the curve. Following is a high-level discussion of ECC. For details, see Reference 13.

Example: Alice and Bob agree on a nonsecret elliptic curve and a nonsecret fixed curve point F. Alice picks a secret random integer A_k as her secret key and publishes the point $A_P = A_k{}^*F$ as her public key. Bob picks a secret random integer B_k as his secret key and publishes the point $B_P = B_k{}^*F$ as his public key. If Alice wants to send a message to Bob, she can compute $A_k{}^*B_P$ and use the result as the secret key for a symmetric block cipher like AES. To decrypt, Bob can compute the same key by finding $B_k{}^*A_P$, because $B_k{}^*A_P = B_k{}^*(A_k{}^*F) = A_k{}^*(B_k{}^*F) = A_k{}^*Bp$.

ECC has not been subject to the extensive analysis that RSA has and is comparatively new.

87.3.4 Attacks

It is possible to attack RSA by factoring large numbers, or guessing all possible values of (P−1)(Q−1) or D. These are computationally infeasible, and users should not worry about them. But there are chosen ciphertext attacks against RSA that involve duping a person to sign a message (provided by the attacker). This can be prevented by signing a hash of the message, or by making minor cosmetic changes to the document by signing it. For a description of attacks against RSA, see Reference[14]. Hash functions are described later in this chapter.

87.3.5 Real-World Applications

Cryptography is often a business enabler. Financial institutions encrypt the connection between the user's browser and Web pages that show confidential information such as account balances. Online merchants similarly encrypt the link so customer credit card data cannot be sniffed in transit. Some even use this as a selling point: "Our Web site is protected with the highest encryption available." What they are really saying is that this Web site uses 128-bit Secure Sockets Layer (SSL).

As an aside, there are no known instances of theft of credit card data in transit; but many high-profile stories of customer information theft, including theft of credit card information, are available. The theft was possible because enough safeguards were not in place, and the data was usable because it was in cleartext, that is, not encrypted. Data worth protecting should be protected in all stages, not just in transit.

87.3.6 SSL and TLS

Normal Web traffic is cleartext—your ISP can intercept it easily. SSL provides encryption between the browser and a Web server to provide security and identification. SSL was invented by Netscape[15] and submitted to the Internet Engineering Task Force (IETF). In 1996, IETF began with SSL v3.0 and, in 1999, published TLS v1.0 as a proposed standard.[16] TLS is a term not commonly used, but we will use TLS and SSL interchangeably.

Suppose Alice, running a popular browser, wants to buy a book from Bob's online book store at bobsbooks.com, and is worried about entering her credit card information online. (For the record, SSL/TLS can encrypt connections between any two network applications and not Web browsers and servers only.) Bob is aware of this reluctance and wants to allay Alice's fears—he wants to encrypt the connection between Alice's browser and bobsbooks.com. The first thing he has to do is install a digital certificate on his Web server.

A certificate contains information about the owner of the certificate: e-mail address, owner's name, certificate usage, duration of validity, and resource location or distinguished name (DN), which includes the common name (CN, Web site address or e-mail address, depending on the usage), and the certificate ID of the person who certifies (signs) this information. It also contains the public key, and finally a hash to ensure that the certificate has not been tampered with.

Anyone can create a digital certificate with freely available software, but just like a person cannot issue his own passport and expect it to be accepted at a border, browsers will not recognize self-issued certificates. Digital certificate vendors have spent millions to preinstall their certificates into browsers, so Bob has to buy a certificate from a well-known certificate vendor, also known as root certificate authority (CA). There are certificates available with 40- and 128-bit encryptions. Because it usually costs the same amount, Bob should buy a 128-bit certificate and install it on his Web server. As of this writing, there are only two vendors with wide acceptance of certificates: VeriSign and Thawte. Interestingly, VeriSign owns Thawte, but Thawte certificate prices are significantly lower.

So now Alice comes back to the site and is directed toward a URL that begins with https instead of http. That is the browser telling the server that an SSL session should be initiated. In this negotiation phase, the browser also tells the server what encryption schemes it can support. The server will pick the strongest of the supported ciphers and reply back with its own public key and certificate information. The browser will check if it has been issued by a root CA. If not, it will display a warning to Alice and ask if she still wants to proceed. If the server name does not match the name contained in the certificate, it will also issue a warning.

If the certificate is legitimate, the browser will:

- Generate a random symmetric encryption key
- Encrypt this symmetric key with the server's public key
- Encrypt the URL it wants with the symmetric key
- Send the encrypted key and encrypted URL to the server

The server will:

- Decrypt the symmetric key with its private key
- Decrypt the URL with the symmetric key
- Process the URL

- Encrypt the reply with the symmetric key
- Send the encrypted reply back to the browser

In this case, although encryption is two-way, authentication is one-way only: the server's identity is proven to the client but not vice versa. Mutual authentication is also possible and performed in some cases. In a high-security scenario, a bank could issue certificates to individuals, and no browser would be allowed to connect without those individual certificates identifying the users to the bank's server.

What happens when a browser capable of only 40-bit encryption (older U.S. laws prohibited export of 128-bit browsers) hits a site capable of 128 bits? Typically, the site will step down to 40-bit encryption. But CAs also sell super or step-up certificates that, when encountered with a 40-bit browser, will temporarily enable 128-bit encryption in those browsers. Step-up certificates cost more than regular certificates.

Note that the root certificates embedded in browsers sometimes expire; the last big one was VeriSign's in 1999. At that time, primarily financial institutions urged their users to upgrade their browsers. Finally, there is another protocol called Secure HTTP that provides similar functionality but is very rarely used.

87.4 Choosing an Algorithm

What encryption algorithm, with what key size, would an information security professional choose? The correct answer is: it depends; what is being encrypted, who do we need to protect against, and for how long?

If it is stock market data, any encryption scheme that will hold up for 20 minutes is enough; in 20 minutes, the same information will be on a number of free quote services. Your password to the *New York Times* Web site? Assuming you do not use the same password for your e-mail account, SSL is overkill for that server. Credit card transactions, bank accounts, and medical records need the highest possible encryption, both in transit and in storage.

87.4.1 Export and International Use Issues

Until recently, exporting 128-bit Web browsers from the United States was a crime, according to U.S. law. Exporting software or hardware capable of strong encryption is still a crime. Some countries have outlawed the use of encryption, and some other countries require a key escrow if you want to use encryption. Some countries have outlawed use of all but certain approved secret encryption algorithms. We strongly recommend that information security professionals become familiar with the cryptography laws of the land, especially if working in an international setting.[17]

87.4.2 Section Summary

- In asymmetric cryptography, one key is used to encrypt and another is used to decrypt.
- Asymmetric cryptography is often also known as public key cryptography.
- Asymmetric cryptography is up to 1000 times slower than symmetric cryptography.
- RSA is the most popular and well-understood asymmetric cryptographic algorithm.
- RSA's security depends on the difficulty of factoring very large (>1024-bit) numbers.
- Elliptic curve cryptography depends on the difficulty of finding discrete logarithms over elliptic curves.
- Smaller elliptic curve keys offer similar security as comparatively larger RSA keys.
- It is possible to attack RSA through chosen plaintext attacks.
- SSL is commonly used to encrypt information between a browser and a Web server.

- Choosing a cipher and key length depends on what needs to be encrypted, for how long, and against whom.
- There are significant legal implications of using encryption in a multinational setting.

87.5 Key Management and Exchange

In symmetric encryption, what happens when one person who knows the keys goes to another company (or to a competitor)? Even with public key algorithms, keeping the private key secret is paramount: without it, all is lost. For attackers, the reverse is true; it is often easier to attack the key storage instead of trying to crack the algorithm. A person who knows the keys can be bribed or kidnapped and tortured to give up the keys, at which time the encryption becomes worthless. Key management describes the problems and solutions to securely generating, exchanging, installing and storing, verifying, and destroying keys.

87.5.1 Generation

Encryption software typically generates its own keys (it is possible to generate keys in one program and use them in another); but because of the implementation, this can introduce weaknesses. For example, DES software that picks a known weak or semiweak key will create a major security issue. It is important to use the largest possible keyspace: a 56-bit DES key can be picked from the 256 ASCII character set, the first 128 of ASCII, or the 26 letters of the alphabet. Guessing the 56-bit DES key (an exhaustive search) involves trying out all 56-bit combinations from the keyspace. Common sense tells us that the exhaustive search of 256 bytes will take much longer than that for 26 bytes. With a large keyspace, the keys must be random enough so as to be not guessable.

87.5.2 Exchange

Alice and Bob are sitting on two separate islands. Alice has a bottle of fine wine, a lock, its key, and an empty chest. Bob has another lock and its key. An islander is willing to transfer items between the islands but will keep anything that he thinks is not secured, so you cannot send a key, an unlocked lock, or a bottle of wine on its own.

How does Alice send the wine to Bob? See the answer at the end of this section.

This is actually a key exchange problem in disguise: how does Alice get a key to Bob without its being compromised by the messenger? For asymmetric encryption, it is easy—the public key can be given out to the whole world. For symmetric encryption, a public key algorithm (like SSL) can be used; or the key may be broken up and each part sent over different channels and combined at the destination.

Answer to our key/wine exchange problem: Alice puts the bottle into the chest and locks it with her lock, keeps her key, and sends the chest to the other island. Bob locks the chest with his lock, and sends it back to Alice. Alice takes her lock off the chest and sends it back to Bob. Bob unlocks the chest with his key and enjoys the wine.

87.5.3 Installation and Storage

How a key is installed and stored is important. If the application does no initial validation before installing a key, an attacker might be able to insert a bad key into the application. After the key is installed, can it be retrieved without any access control? If so, anyone with access to the computer would be able to steal that key.

87.5.4 Change Control

How often a key is changed determines its efficiency. If a key is used for a long time, an attacker might have sufficient samples of ciphertext to be able to cryptanalyze the information. At the same time, each change brings up the exchange problem.

87.5.5 Destruction

A key no longer in use has to be disposed of securely and permanently. In the wrong hands, recorded ciphertext may be decrypted and give an enemy insights into current ciphertext.

87.5.6 Examples and Implementations

87.5.6.1 PKI

A public key infrastructure (PKI) is the set of systems and software required to use, manage, and control public key cryptography. It has three primary purposes: publish public keys, certify that a public key is tied to an individual or entity, and provide verification as to the continued validity of a public key. As discussed before, a digital certificate is a public key with identifying information for its owner. The certificate authority (CA) "signs" the certificate and verifies that the information provided is correct. Now all entities that trust the CA can trust that the identity provided by a certificate is correct. The CA can revoke the certificate and put it in the certificate revocation list (CRL), at which time it will not be trusted anymore. An extensive set of PKI standards and documentation is available.[18] Large companies run their own CA for intranet/extranet use. In Canada and Hong Kong, large public CAs are operational. But despite the promises of the "year of the PKI," market acceptance and implementation of PKIs are still in the future.

87.5.6.2 Kerberos

From the comp.protocol.kerberos FAQ:

> Kerberos; also spelled Cerberus. *n*. The watchdog of Hades, whose duty it was to guard the entrance—against whom or what does not clearly appear; it is known to have had three heads.
>
> **—Ambrose Bierce**
> *The Enlarged Devil's Dictionary*

Kerberos was developed at MIT in the 1980s and publicly released in 1989. The primary purposes were to prevent cleartext passwords from traversing the network and to ease the log-in process to multiple machines.[19] The current version is 5—there are known security issues with version 4. The three heads of Kerberos comprise the key distribution center (KDC), the client, and the server that the client wants to access. Kerberos 5 is built into Windows 2000 and later, and will probably result in wider adoption of Kerberos (notwithstanding some compatibility issues of the Microsoft implementation of the protocol[20]).

The KDC runs two services: authentication service (AS) and ticket granting service (TGS). A typical Kerberos session (shown in Exhibit 87.6) proceeds as follows when Alice wants to log on to her e-mail and retrieve it.

1. She will request a ticket granting ticket (TGT) from the KDC, where she already has an account. The KDC has a hash of her password, and she will not have to provide it. (The KDC must be extremely secure to protect all these passwords.)
2. The TGS on the KDC will send Alice a TGT encrypted with her password hash. Without knowing the password, she cannot decrypt the TGT.
3. Alice decrypts the TGT; then, using the TGT, she sends another request to the KDC for a service ticket to access her e-mail server. The service ticket will not be issued without the TGT and will only work for the e-mail server.
4. The KDC grants Alice the service ticket.
5. Alice can access the e-mail server.

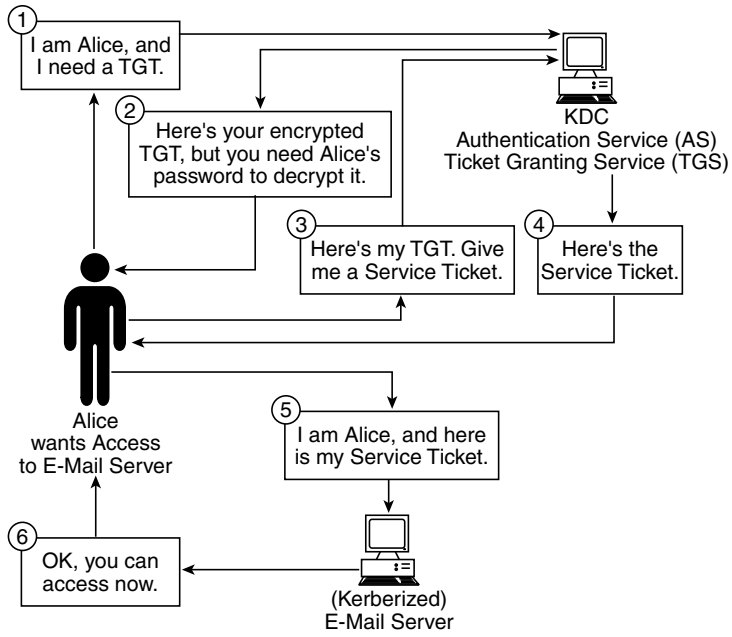

EXHIBIT 87.6 Kerberos in operation.

Note that both the TGT and the ST have expiration times (default is ten hours); so even if one or both tickets are captured, the exposure is only until the ticket expiration time. All computer system clocks participating in a Kerberos system must be within five minutes of each other and all services that grant access. Finally, the e-mail server must be kerberized (support Kerberos).

87.5.7 Section Summary

- Key management (generating/exchanging/storing/installing/destroying keys) can compromise security.
- Public key cryptography is often the best solution to key distribution issues.
- A public key infrastructure (PKI) is a system that can manage public keys.
- A certificate authority (CA) is a PKI that can validate public keys.
- Digital certificates are essentially public keys that also include key owner information. The key and information are verified by a CA.
- If an entity trusts a CA, it can also trust digital certificates that the CA signs (authenticates).
- Kerberos is a protocol for eliminating cleartext passwords across networks.
- A ticket granting ticket (TGT) is issued to the user, who will use that to request a service ticket. All tickets expire after a certain time.
- Under Kerberos, tickets are encrypted and cleartext passwords never cross the network.

87.6 Hash Functions

A hash function is defined as a process that can take an arbitrary-length message and return a fixed-length value from that message. For practical use, we require further qualities:

- Given a message, it should be easy to find the hash.
- Given the hash, it should be hard to find the message.
- Given the message, it should be hard to find another (specific or random) message that produces the same hash.

87.6.1 Message Digests

A message digest is the product of a one-way hash function applied on a message: it is a fingerprint or a unique summary that can uniquely identify the message.

87.6.2 MD2, MD4, and MD5

Ron Rivest (the R in RSA) designed all of these. All three produce 128-bit hashes. MD4 has been successfully attacked. MD5 has been found weak in certain cases; it is possible to find another random message that will produce the same hash. MD2 is slower, although no known weaknesses exist.

87.6.3 SHA

The secure hash algorithm (SHA) was designed by NIST and NSA, and is used in the digital signature standard, officially known as the Secure Hash Standard (SHS) and is available as FIPS-180-1.[21]

The current SHA produces a 160-bit hash and is also known as SHA-1. There are additional standards undergoing public comments and reviews that will offer 256-, 384-, and 512-bit hashes. The draft standard is available.[16] The proposed standards will offer security matching the level of AES. The draft is available as FIPS-180-2.[22]

87.6.4 Applications of Message Digests

Message digests are useful and should be used to provide message integrity. Suppose Alice wants to pay $2000 to Eve, a contract network administrator. She types an e-mail to Bob, her accountant, to that effect. Before sending the message, Alice computes the message digest (SHA-1 or MD5) of the message and then sends the message followed by the message digest. Eve intercepts the e-mail and changes $2000 to $20,000; but when Bob computes the message digest of the e-mail, it does not match the one from Alice, and he knows that the e-mail has been tampered with.

But how do we ensure that the e-mail to Bob indeed came from Alice, when faking an e-mail source address is notoriously easy? This is where digital signatures come in.

87.6.5 Digital Signatures

Digital signatures were designed to provide the same features of a conventional ("wet") signature. The signature must be non-repudiatable, and it must be nontransferable (cannot be lifted and reused on another document). It must also be irrevocably tied back to the person who owns it.

It is possible to use symmetric encryption to digitally sign documents using an intermediary who shares keys with both parties, but both parties do not have a common key. This is cumbersome and not practical.

Using public key cryptography solves this problem neatly. Alice will encrypt a document with her private key, and Bob will decrypt it with Alice's public key. Because it could have been encrypted with only Alice's private key, Bob can be sure it came from Alice. But there are two issues to watch out for: (1) the rest of the world may also have Alice's public key, so there will be no privacy in the message; and (2) Bob will need a trusted third party (a certificate authority) to vouch for Alice's public key.

In practice, signing a long document may be computationally costly. Typically, first a one-way hash of the document is generated, the hash is signed, and then both the signed hash and the original document

are sent. The recipient also creates a hash and compares the decrypted signed hash to the generated one. If both match, then the signature is valid.

87.6.6 Digital Signature Algorithm (DSA)

NIST proposed DSA in 1991 to be used in the Digital Signature Standard and the standard issued in May 1994. In January 2000, it announced the latest version as FIPS PUB 186-2.[23] As the name implies, this is purely a signature standard and cannot be used for encryption or key distribution.

The operation is pretty simple. Alice creates a message digest using SHA-1, uses her private key to sign it, and sends the message and the digest to Bob. Bob also uses SHA-1 to generate the message digest from the message and uses Alice's public key on the received message digest to decrypt it. Then the two message digests are compared. If they match, the signature is valid.

Finally, digital signatures should not be confused with the horribly weakened "electronic signature" law passed in the United States, where a touch-tone phone press could be considered an electronic signature and enjoy legal standing equivalent to an ink signature.

87.6.7 Message Authentication Codes (MACs)

MACs are one-way hash functions that include the key. People with the identical key will be able to verify the hash. MACs provide authentication of files between users and may also provide file integrity to a single user to ensure files have not been altered in a Web site defacement. On a Web server, the MAC of all files could be computed and stored in a table. With only a one-way hash, new values could have been inserted in the table and the user will not notice. But in a MAC, because the attacker will not know the key, the table values will not match; and an automated process could alert the owner (or automatically replace files from backup).

A one-way hash function can be turned into a MAC by encrypting the hash using a symmetric algorithm and keeping the key secret. A MAC can be turned into a one-way hash function by disclosing the key.

87.6.8 Section Summary

- Hash functions can create a fixed-length digest of arbitrary-length messages.
- One-way hashes are useful: given a hash, finding the message should be very hard.
- Two messages should not generate the same hash.
- MD2, MD4, and MD5 all produce 128-bit hashes.
- SHA-1 produces a 160-bit hash.
- Encrypting a message digest with a private key produces a digital signature.
- Message authentication codes are one-way hashes with the key included.

87.7 Other Cryptographic Notes

87.7.1 Steganography

Steganography is a Greek word that means sheltered writing. This is a method that attempts to hide the existence of a message or communication. In February 2001, *USA Today* and various other news organizations reported that terrorists are using steganography to hide their communication in images on the Internet.[24] A University of Michigan study[25] examined this by analyzing two million images downloaded from the Internet and failed to find a single instance.

In its basic form, steganography is simple. For example, every third letter of a memo could hide a message. And it has the added advantage over encryption that it does not arouse suspicion: often, the

presence of encryption could set off an investigation; but a message hidden in plain sight would be ignored.

The medium that hides the message is called the cover medium, and it must have parts that can be altered or used without damaging or noticeably changing the cover media. In case of digital cover media, these alterable parts are called redundant bits. These redundant bits or a subset can be replaced with the message we want to hide.

Interestingly, steganography in digital media is very similar to digital watermarking, where a song or an image can be uniquely identified to prevent theft or unauthorized use.

87.7.2 Digital Notary Public

Digital notary service is a logical extension of digital signatures. Without this service, Alice could send a digitally signed offer to Bob to buy a property; but after property values drop the next day, she could claim she lost her private key and call the message a forgery. Digital notaries could be trusted third parties that will also time-stamp Alice's signature and give Bob legal recourse if Alice tries to back out of the deal. There are commercial providers of this type of service.

With time-sensitive offers, this becomes even more important. Time forgery is a difficult if not impossible task with paper documents, and it is easy for an expert to detect. With electronic documents, time forgeries are easy and detection is almost impossible (a system administrator can change the time stamp of an e-mail on the server). One do-it-yourself time-stamping method suggests publishing the one-way hash of the message in a newspaper (as a commercial notice or advertisement). From then on, the date of the message will be time-stamped and available for everyone to verify.

87.7.3 Backdoors and Digital Snake Oil

We will reiterate our warnings about not using in-house cryptographic algorithms or a brand-new encryption technology that has not been publicly reviewed and analyzed. It may promise speed and security or low cost, but remember that only algorithms that withstood documented attacks are worthy of serious use—others should be treated as unproven technology, not ready for prime time.

Also, be careful before using specific software that a government recommends. For example, Russia mandates use of certain approved software for strong encryption. It has been mentioned that the government certifies all such software after behind-the-scenes key escrow. To operate in Russia, a business may not have any choice in this matter, but knowing that the government could compromise the encryption may allow the business to adopt other safeguards.

References

1. Data Encryption Standard (DES): http://www.itl.nist.gov/fipspubs/fip46-2.htm.
2. Specialized DES cracking computer: http://www.eff.org/descracker.html.
3. Advanced Encryption Standard (AES): http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.
4. Bruce Schneier, *Applied Cryptography, 2nd edition*.
5. Weak DES keys: http://www.ietf.org/rfc/rfc2409.txt, Appendix A.
6. AES selection report: http://csrc.nist.gov/encryption/aes/round2/r2report.pdf.
7. Rijndael developer's site: http://www.esat.kuleuven.ac.be/~rijmen/rijndael/.
8. Rijndael technical overview: http://www.baltimore.com/devzone/aes/tech_overview.html.
9. Rijndael technical overview: http://www.sans.org/infosecFAQ/encryption/mathematics. htm.
10. PKZIP encryption weakness: http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/1994/CS/CS0842.ps.gz.
11. Diffie and Hellman paper on Public Key Crypto: http://cne.g.,mu.edu/modules/acmpkp/security/texts/NEWDIRS.PDF.

12. RSA algorithm: http://www.rsasecurity.com/rsalabs/rsa_algorithm/index.html.
13. Paper on elliptic curve cryptography: ftp://ftp.rsasecurity.com/pub/ctryptobytes/crypto1n2.pdf.
14. Attacks on RSA: http://crypto.stanford.edu/~dabo/abstracts/RSAattack-survey.html.
15. SSL 3.0 protocol: http://www.netscape.com/eng/ssl3/draft302.txt.
16. TLS 1.0 protocol: http://www.ietf.org/rfc/rfc2246.txt.
17. International encryption regulations: http://cwis.kub.nl/~frw/people/koops/lawsurvy. htm.
18. IETF PKI working group documents: http://www.ietf.org/html.charters/pkix-charter.html.
19. Kerberos documentation collection: http://web.mit.edu/kerberos/www/.
20. Kerberos issues in Windows 2000: http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html#ntbroken.
21. Secure Hash Standard (SHS): http://www.itl.nist.gov/fipspubs/fip180-1.htm.
22. Improved SHS draft: http://csrc.nist.gov/encryption/shs/dfips-180-2.pdf.
23. Digital Signature Standard (DSS): http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf.
24. *USA Today* story on steganography: http://www.usatoday.com/life/cyber/tech/2001-02-05-binladen. htm#more.
25. Steganography study: http://www.citi.umich.edu/techreports/reports/citi-tr-01-11.pdf.