



## **Cryptography's Role in Securing the Information Society**

Kenneth W. Dam and Herbert S. Lin, Editors,  
Committee to Study National Cryptography Policy,  
National Research Council

ISBN: 0-309-52254-4, 720 pages, 6 x 9, (1996)

**This PDF is available from the National Academies Press at:**  
<http://www.nap.edu/catalog/5131.html>

Visit the [National Academies Press](http://www.nap.edu) online, the authoritative source for all books from the [National Academy of Sciences](http://www.nap.edu), the [National Academy of Engineering](http://www.nap.edu), the [Institute of Medicine](http://www.nap.edu), and the [National Research Council](http://www.nap.edu):

- Download hundreds of free books in PDF
- Read thousands of books online for free
- Explore our innovative research tools – try the “[Research Dashboard](#)” now!
- [Sign up](#) to be notified when new books are published
- Purchase printed books and selected PDF files

**Thank you for downloading this PDF. If you have comments, questions or just want more information about the books published by the National Academies Press, you may contact our customer service department toll-free at 888-624-8373, [visit us online](#), or send an email to [feedback@nap.edu](mailto:feedback@nap.edu).**

**This book plus thousands more are available at <http://www.nap.edu>.**

Copyright © National Academy of Sciences. All rights reserved.

Unless otherwise indicated, all materials in this PDF File are copyrighted by the National Academy of Sciences. Distribution, posting, or copying is strictly prohibited without written permission of the National Academies Press. [Request reprint permission for this book](#).

# C

## A Brief Primer on Cryptography

This appendix provides a brief primer on cryptography, but it is necessary to understand from the start that cryptography is not a “silver bullet” for information security. For example, a network may be insecure in the sense that it is easy for an adversary to obtain information that is flowing on the network. End users may use very strong cryptography to protect this information. But if sufficiently motivated and skilled, adversaries may well attempt to penetrate the systems attached to the network, where they can obtain the information in the clear. Or they may be able to bribe a system operator to obtain it for them. Nevertheless cryptography still has value under these circumstances, because it forces the adversary to alter his or her attack and expend greater effort to obtain information; furthermore, the use of cryptography will foil some adversaries who are not motivated or skilled enough to develop alternative attacks.

### C.1 A VERY SHORT HISTORY OF CRYPTOGRAPHY

For most of its history, cryptography was more an art than a science and was devoted primarily to keeping messages and records secret. To be sure, mathematical techniques for cryptanalysis and engineering skills for building devices for encryption and decryption played important roles, but cryptography itself did not rest on a firm mathematical foundation.

The scientific basis for modern cryptography was established in 1949 with the development of information theory by Claude Shannon, who determined for the first time a mathematically rigorous basis for defining

a “perfect” encryption system that could be made impenetrable, even in principle, to an adversary with unlimited resources. Based on this work, secret-key cryptography (defined below) blossomed, with the most public work in this area being the Data Encryption Standard promulgated in 1975 by the National Bureau of Standards (now the National Institute of Standards and Technology). The second major revolution occurred in 1976 with the first discussion in the open literature of asymmetric cryptography, inspired by a landmark paper of Whitfield Diffie and Martin Hellman.<sup>1</sup>

## C.2 CAPABILITIES ENABLED BY CRYPTOGRAPHY

Cryptography can help to ensure the integrity of data (i.e., that data retrieved or received are identical to data originally stored or sent), to authenticate specific parties (i.e., that the purported sender or author of a message is indeed its real sender or author), to facilitate nonrepudiation, and to preserve the confidentiality of information that may have come improperly into the possession of unauthorized parties.

To understand how cryptographic methods span a range of communication and storage needs, consider the general problem of sending a private message from Party A to Party B. Centuries ago, such a process was accomplished by Party A writing a letter containing his or her signature (authentication). The letter was sealed inside a container to prevent accidental disclosure (confidential transmission). If Party B received the container with a broken seal, it meant that the letter had been disclosed or altered and Party B would take appropriate actions (data integrity). Otherwise, Party B would verify Party A’s signature and read the message. In the information era, each of the steps remains essentially the same, except that automated tools perform most of the work and are explained below.

### C.2.1 Ensuring the Integrity of Data

Digital information is transmitted (or stored) so that it can be received (or retrieved). For two reasons, it is possible that the information received or retrieved might differ from the original information transmitted or stored:

1. A technical problem may inadvertently alter one or more of the bits of information in question. No digital transmission-receiving or stor-

---

<sup>1</sup>Whitfield Diffie and Martin Hellman, “New Directions in Cryptography,” *IEEE Transactions on Information Theory*, Volume IT-22, IEEE Press, New York, 1976, pp. 644-654.

age and retrieval system is perfect—every now and then, with a frequency depending on the particular characteristics of the technology used in the system and the environment in which it operates, a “1” will be received or retrieved when a “0” is sent or stored and vice versa.

2. A third party may deliberately alter one or more of the bits of information in question. For example, a proposal by Vendor A to a buyer may offer to undertake a task for \$100,000. Vendor B, competing for the same contract but wishing to charge \$150,000, may intercept the digital transmission of Vendor A’s proposal and deliberately alter the \$100,000 to \$300,000. Thus, the buyer would be presented with information that falsely understated the cost-effectiveness of Vendor A and would award the contract to Vendor B.

In some cases, the alteration of one or even many bits may not render the information received or retrieved useless (e.g., if the bits constitute a digital representation of a photograph). However, for other purposes (e.g., the transmission of a software program), even a one-bit difference between what was received and what was transmitted could make all the difference in the world.

It is therefore desirable to ensure that any alterations, whether inadvertent or deliberate, can be detected if they occur. An integrity lock or integrity check is a quantity derived algorithmically from the bits that constitute the information being transmitted or stored and appended to it for the purpose of ensuring that the information received or retrieved is identical to the information being transmitted or stored.

Cryptography is relevant to integrity checks that are intended to detect deliberate alterations. In such cases, the integrity check (also known as a message authenticator) must use a process that involves information unknown to potential penetrators; that is, it has parts that are secret and known only to the communicating parties (usually a secret key).<sup>2</sup>

In the example of Party A sending a message to Party B, Party A attaches a number called a cryptographic checksum that is generated

---

<sup>2</sup>Protecting against inadvertent alterations is the job of error-correcting codes (e.g., cyclic redundancy checks, Reed-Solomon codes, parity checks). However, for error correction, the process and techniques in question will be known commonly to all users. Thus, if a message is protected only by error-correcting codes, a person could use this knowledge to construct revised error checks that would conceal deliberate alterations. The use of error-correcting codes does not ensure integrity against intended subversions of the transmission.

Note: Although under some circumstances (e.g., for limited numbers of presumably random and inadvertent alterations), an error-correcting code can correct bits that have been changed.

Generally, a cryptography-based approach is much more sensitive to changes in the message than usual error-correction schemes; that is, cryptography provides assurances both of confidentiality and of message integrity.

### BOX C.1 Checksums and Hashes

Checksums were originally used to detect errors in stored or transmitted data. The simplest checksum is a single bit that is the XOR of all message bits. This can detect single errors, but not double errors. Most error-detecting codes add more complex checksums (often called CRCs, for cyclic redundancy checks) to detect much larger numbers of errors.

For nonmalicious errors owing to physical error phenomena, such checksums are fine. But when an opponent might try to corrupt data, a cryptographically secure checksum is needed—one that will detect random errors and prevent malicious errors. For example, one of the federal standards relating to DES describes a message authentication code (MAC), which is formed by enciphering the message under a secret key known only to authorized parties and adding the last 64 bits (or less if that suffices) of ciphertext as a MAC. Clearly, another authorized user can validate the MAC by enciphering the message (which is sent in the clear—not enciphered—since only authentication is needed in this application) and comparing the computed MAC with the received MAC. An opponent who does not know the secret key has as much trouble computing a modified MAC to go with a corrupted version of the data as in breaking DES. (If enciphering and deciphering are mathematically equivalent, it is just as hard to encipher without the key as to decipher without the key.)

A hash function is a pseudorandom function that is shorter than its input. Originally used in searching and sorting algorithms, where it did not need any cryptographic properties, a one-way hash function is useful for digital signatures because the hash can be signed by the public-key cryptographic system, rather than the much longer message. “One-way” means that it is easy to compute  $H(\text{message})$  but computationally infeasible to compute any inverse image of a given value  $H$ —or if one inverse image is known (e.g., if a spoofer who has intercepted a message knows the message and  $H(\text{message})$ ), it is computationally infeasible to find another inverse image. (There are many inverse images since  $H$  is a compressive function.) The one-way property is needed because the signature is valid not only for the message signed, but also for any other message with the same hash value.

In short, a cryptographic checksum depends on a secret key known to the authorized transmitter and receiver, whereas a one-way hash value can be computed by anyone. The hash value is then acted on by the secret key in an asymmetric cryptographic system to produce a digital signature.

algorithmically from specific characteristics of the message (e.g., the letters and numbers in the message; see Box C.1. Party B can use the same algorithm to compute the checksum of the message received and compare it to the checksum sent, and if they match, Party B can be confident of the message's data integrity.

### C.2.2 Authentication of Users

In many communications systems, it is quite important to establish the clear and unquestionable identity of the communicating parties; the

process of establishing and verifying the identity of a party is referred to as *authentication*.<sup>3</sup>

Authentication is based on something that the proper party would know, would have, or would be. For example, a specially designed electronic ring might be owned or worn only by individuals authorized to wear it.<sup>4</sup> A secret password might be regarded as being known only to a certain party. Another approach based on secret knowledge involves one party challenging another party to answer certain questions that could be answered correctly only by one with the proper identity. In banking circles, a customer's mother's maiden name is often an authenticator, since it is assumed that such a fact would not be readily known to an impersonator (but contemporary dossier-quality databases of personal information significantly weaken the assurance). In telephony, humans frequently authenticate each other merely by the recognized sound of a voice or simply by the fact that a certain individual answers the telephone when the number alleged to be that individual's number is dialed. Vendors accepting credit cards use handwritten signatures to authenticate identities on the assumption that only the proper holder of a credit card can sign the credit card charge slip in a way that matches the signature on the card. Stronger authentication methods often involve hardware—a tangible object or artifact—that must be associated with authorized users and that is not easily duplicated. (The ultimate "hardware" involved might

---

<sup>3</sup>Authentication is almost always a requirement for authorized access (to use a system; to read, write, modify, or destroy data; to run a software process; to make use of computer or communications resources). Access rights have qualifications that are often called privileges (e.g., access to data might include some or all of the privileges of read, write, modify, destroy). Similarly, not all users of a system will be accorded free run of all the software in the system (i.e., their privileges will be restricted). A system electronically accessing another without human intervention typically will not be entitled to all of the latter's data and/or software privileges. For example, one system might be authorized, or privileged, to ask for only a certain kind of data (e.g., only the cash value of point-of-sale information will be exchanged with bankcard authorization systems).

For some security requirements, authentication by itself may be sufficient. For example, the "security" of the credit card industry is based primarily on authentication mechanisms, not secrecy mechanisms. People routinely recite their credit card numbers to anyone and everyone who wants to be paid for services or products. Some degree of security is provided by using other information, such as the expiration date or mother's maiden name, to authenticate the person using that number. Such authentication is performed only when the risk of fraud is above a given level (e.g., the purchase of an expensive item or too much credit card activity in a given period of time). Secrecy mechanisms are used, for the most part, to prevent eavesdroppers from getting one's card number, but most fraud is conducted without eavesdropping. For example, cards are stolen, numbers are stolen from vendor databases, and merchant employees copy numbers.

Authentication and confidentiality are complementary tools for supporting information security, as the discussion above and that in Box C.2 make clear.

<sup>4</sup>"Dick Tracy Eat Your Heart Out," *New York Times*, September 4, 1995, p. 38.

well be biometric in nature: a person's handprint, a fingerprint, or a retinal pattern.) Of course, except in the case of biometric identifiers, all authentication systems can be compromised if the secret or the hardware token belonging only to the proper party is passed on to unauthorized parties.<sup>5</sup>

Even though such mechanisms are not perfect, they are routinely used to conduct personal and business interactions, and most of those communications use nonprivate channels. As we move toward an electronic economy, conducted over wide-scale communication networks, it becomes increasingly important to develop stronger authentication mechanisms that prevent wrongdoers of all types from being able to access remote resources without proper authorization. The reason is that electronic commerce disconnects consumers and suppliers from the physical mechanisms that help curb fraud. For example, vendors accepting credit cards for face-to-face transactions check the signature on the card (or else accept liability for not performing the check). Mail-order and telephone vendors use the mailing address of a customer to help authenticate a purchase; large orders are generally not sent to an address different from the credit card billing address, unless extra steps are taken to ensure authenticity. However, when such mechanisms are not available, electronic commerce will require strong cryptography-based mechanisms that will help to establish the identity of the consumer.

Thus, it is the goal of most communications system designers to provide strong authenticity of the communicating parties. It should be noted, however, that in some cases (such as telephone Caller ID services), the communicating party may wish to be anonymous (or pseudonymous) for good reason and system designers must take this into account.

Cryptography-based authentication could also help to deal with the problem of controlling the secondary use of data collected from individuals (described in Chapter 1). For example, a requirement to include the source of personal data (e.g., the original party to which an individual discloses personal data) with the person-identified information at the time of disclosure would help the individual keep track of how such information is subsequently used. Such a requirement could be enforced through the use of a digital signature belonging to the data source being bound to the personal information before it is disseminated.<sup>6</sup>

---

<sup>5</sup>For this reason, very strong authentication requires hardware components that can be in the possession of only one person at a time. Nevertheless, software-based authentication has many advantages (such as ease of deployment and perhaps lower costs) that may prove decisive against the lower levels of confidence that are possible with such methods. Software-based authentication is better than nothing, and the decisions regarding medium will be made on the basis of business needs for differing levels of confidence.

<sup>6</sup>Personal communication, Marc Rotenberg, Electronic Privacy Information Center, Washington, D.C., March 10, 1995.

Finally, good authentication mechanisms can facilitate the generation of reliable audit trails that allow the investigation of possible wrongdoing. Such mechanisms have high value when many individuals are in a position to compromise the same sensitive data.

### C.2.3 Nonrepudiation

The authentication of a user and the integrity of a message sent by a user are two different concepts. For example, being assured of a message's integrity does not in itself assure the receiver that its purported sender did in fact send it.

*Nonrepudiation* is a cryptographic capability that combines techniques for ensuring user authentication and message integrity in such a way that the signer of a message cannot plausibly deny that it was he who created it or claim that the message received was not in fact the message sent. In other words, nonrepudiation protects against impersonation and denial of creation.

Digital signatures typically are used to provide nonrepudiation. A digital signature is a piece of information derived from both information known only to the sender (Party A) and the exact text of the message sent.<sup>7</sup> On the basis of information freely available to the sender (Party A), the receiver (Party B), and any evildoer, Party B can check the digital signature of the message allegedly sent by Party A against the message actually received. If nothing improper has occurred,

- Party B can be assured that Party A was in fact the sender;
- Party B can be assured that the message received was actually the one Party A sent; and
- If Party A ever denies having sent the message, Party B can prove that Party A did.<sup>8</sup>

Again, if the secrets on which authentication is based are compromised, a valid signature does not mean that a message was actually sent

---

<sup>7</sup>Although the entire message could be run through an encryption process and some part of the result used as the digital signature, in normal practice only a digest of the message is subject to this process to conserve both time and computer resources. The digest is created by an algorithm (usually known as a secure hash algorithm) that shortens a message of any length into a result that is of fixed and known length. This result (the digest or hash) is constructed in such a way that the likelihood that different original data items will produce identical digests is very small.

<sup>8</sup>Strictly speaking, this is true only if an asymmetric authentication system is used. An authentication system based on symmetric cryptography and secret keys can protect only against third-party forgeries, but not against the case in which Party B forges a message, claiming it to be from Party A. The reason is that Party A and Party B (but not a third party) both know a common secret key.

by the person who would normally be associated with that signature. If Party A gives his or her secret to Party C (e.g., a secretary) and Party C uses Party A's secret to send a message, that message is indistinguishable from one actually sent by Party A. Moreover, anyone receiving a message signed by Party A has a right to expect that Party A sent it and to take action on that basis. For example, if Party A completes an electronic message to buy certain items, that party will digitally sign the message in preparation for sending it. However, if Party A's attention is diverted during this time, Party C might actually send the message to a different supplier. This different supplier can verify that the message was signed by an authorized individual (Party A) and has every right to conclude that a valid purchase order has been received.

Finally, nonrepudiation often includes a time element; for example, one must be able to prove not only that he or she directed a stockbroker to buy 1,000 shares of Company XYZ at \$30 per share, but also when the order was placed. Note that if the date and time are part of the message being signed, then the sender also cannot repudiate that date and time at which he or she signed the message. A greater degree of confidence that the date and time are in fact correct can be provided by secure date/time stamps.<sup>9</sup>

#### C.2.4 Preservation of Confidentiality<sup>10</sup>

It is inherent and assumed in most communications system design that communications between parties should be controlled in such a way that unintended access by others is prohibited. There are three common

---

<sup>9</sup>Secure date/time stamping is a technique involving a trusted third party to certify the creation of a document at a given time. Conceptually, a digital document is mailed to this trusted third party, who provides a date/time stamp and a digital signature of the document-stamp combination. If the date/time stamp is inserted correctly by the third party, the digital signature of that party ensures that the document did indeed exist at the time and date in the document. More discussion of this concept can be found in Barry Cipra, "Electronic Time-Stamping: The Notary Public Goes Digital," *Science*, Volume 261(5118), July 9, 1993, pp. 162-163, and on-line at <http://www.surety.com>.

<sup>10</sup>Note that in this section (and throughout this report unless otherwise stated explicitly), the term "confidentiality" (or, synonymously, secrecy) applies to data in a technical sense. There is another sense in which the term "confidentiality" is often used that refers to a policy context—the assertion that data are sensitive and must be protected from unauthorized parties. In a policy sense, confidentiality can be accomplished by techniques based entirely on access control and authorization—individuals without proper authorization are not permitted to view confidential data.

Thus, the distinction can be made between data that *are* confidential (i.e., on policy grounds, a person's AIDS/HIV status may be confidential data; the law recognizes the confidentiality of communications between lawyer and client, husband and wife, priest and parishioner) and data that are *made* confidential by the technical means described in this section.

methods of gaining confidentiality of communications: physical security, obfuscation, and encryption.

In the case of physical security, the communicator relies on the fact that the attacker will have a very difficult time physically penetrating the communications media or devices, or that it will be too costly for an attacker to do so. An example of this is an optical fiber, a medium that is inherently difficult to tap into without being intrusive to the communication.

In the case of obfuscation, the communicator relies upon the fact that communicated information is so well hidden in some surrounding container that it will be difficult for an attacker to recognize and thus retrieve it. An example of this is steganography, in which data can be hidden in things such as photographs.<sup>11</sup>

Finally, with encryption, one communicating party encodes information by using an agreed-upon coding method; the information is transmitted to its destination; then the other communicating party decodes the information. In this case, the communicator is relying on the fact that for someone other than the intended recipient, it will be very difficult to break the code or discover a secret that the code depends on, such as a key.

When used for preserving confidentiality, cryptography enables the system designer to separate the security of a message from the security of the medium used to transmit that message. Since some of the most useful and least expensive media in use today are insecure (e.g., wireless communications), such separation has obvious advantages. Even the most sophisticated cryptography today requires some keeping of secrets, but a properly implemented cryptography system reduces the problem of keeping messages secret to the problem of keeping secret a much smaller key, thereby simplifying the security problem.

Note that confidentiality and authentication are tied closely together, as discussed in Box C.2. Furthermore, systems that provide strong authentication capabilities and those that provide strong confidentiality can serve a similar purpose under some circumstances. For example, confidentiality provided by cryptography can keep hackers from learning a credit card number that is sent over the Internet, while authentication provided by cryptography can keep hackers from using that credit card number once they get it.<sup>12</sup>

---

<sup>11</sup>A simple example: most black-and-white pictures rendered in digital form use at most  $2^{16}$  (65,536) shades of gray, because the human eye is incapable of distinguishing any more shades. Each element of a digitized black-and-white photo would then be associated with 16 bits of information about what shade of gray should be used. If a picture were digitized with 24 bits of gray scale, the last 8 bits could be used to convey a concealed message that would never appear except for someone who knew to look for it. The digital size of the picture would be 50% larger than it would have to be, but no one but the creator of the image would know.

<sup>12</sup>Of course, the problem is that, in practice, many uses of credit card numbers do not

### BOX C.2 Dependence of Confidentiality on Authentication

Confidentiality in electronic communications is not possible without authentication. Suppose that Party A and Party B want to communicate in such a way that Party C cannot eavesdrop, but that no authentication is performed. One might conjecture a system that selects a random session key, without telling Party A and Party B, and then encrypts everything communicated between them. Unfortunately, such a system is not confidential because Party C could place himself between Party A and Party B, relaying all information between both parties (or only the information Party C wanted to pass). This is possible because it was assumed that no authentication existed. That is, by assumption the system cannot distinguish among Party A, Party B, or Party C, and neither can any of the parties involved.

In practice, there are numerous mechanisms that seemingly provide a sufficient level of authentication for business or personal communications. For example, people routinely "authenticate" the person on the other end of a telephone call by recognizing the voice. Unfortunately, this still does not provide the necessary foundation for a secure telephone system. For example, Party C can simply listen to the conversation that he or she is relaying between Party A and Party B, without participating. (This scenario illustrates an illicit form of call forwarding; Party C rigs the telephone system to be called when Party A dials Party B's number, and Party C automatically dials Party B when a call from Party A is received.) Since the telephone system has no authentication, by assumption, Party C's scheme cannot be prevented even if Party A recognizes Party B's voice (which is a very strong end-to-end authentication mechanism).

Similarly, one might assume that the telephone system itself does not allow the type of tampering that Party C needs to place himself between Party A and Party B. In other words, the telephone system is designed in such a way that when Party A dials Party B's number, the call is routed directly to Party B's telephone. This arrangement is characteristic of most telephone systems today. However, its success depends on the ability of the telephone system to authenticate the maintainers of the system. Although the population of valid system users is smaller than the population of telephone users, the former is still relatively large (more than a few people), and history has shown that wide-ranging networks are difficult, if not impossible, to secure without strong authentication mechanisms.

For a communications system to be confidential, the system itself must authenticate the end users. Only then can it exchange the secret information needed to establish a confidential connection between those users. Authentication is a necessary, but not sufficient, condition for confidentiality.

---

require strong authentication (e.g., telephone orders), even if security procedures are intended to minimize the incidence of fraud in such orders (e.g., not shipping an order to an address other than the billing address on the credit card). If every use of a credit card required cryptographic authentication, revealing a credit card number to the world would not have much significance.

### C.3 BASIC CONSTRUCTS OF CRYPTOGRAPHY

Cryptography began the science of keeping information secret from those not authorized to see it. In this classical application (called encryption in this report), cryptography has been used for thousands of years. Today, cryptographic methods help solve critical information-age problems, including those of data confidentiality (keeping data private), data integrity (ensuring that data retrieved or received are identical to data originally stored or sent), and subject authentication (ensuring that the purported sender or author of a message is indeed its real sender or author). Box C.3 contains some additional applications of cryptography.

In general, cryptographic systems involve the following:

- *The message to be sent* (usually known as the plaintext); for example, a sentence written in English. The plaintext is the message that Party A composes for reading by Party B. All plaintext messages can be represented as numbers (e.g., by using 00 for A, 01 for B, and so on, with 26 for space, 27 for comma, 28 for period, 29 for semicolon, and 30 for question mark).
- *The ciphertext* (the gibberish that results from encryption) that anyone can see without compromising the plaintext message.
- *An encryption algorithm* (a series of mathematical steps) that Party A uses, in combination with an encryption key, to generate the ciphertext.
- *A decryption algorithm* that Party B uses, in combination with a decryption key, to retrieve the plaintext from the ciphertext.

One of the simplest encryption schemes is the following: for every letter in the plaintext message (represented by a number), add 1 to obtain the corresponding ciphertext message letter. The encryption algorithm is simple addition, with an encryption key of 1.<sup>13</sup> The same encryption algorithm could be employed using a different encryption key (i.e., a number other than 1). The corresponding decryption algorithm is subtraction, with a decryption key of 1.

One of the fundamental goals of cryptographic research is to develop algorithms that can be used effectively within a specific system and that are difficult to “crack.” (A more precise definition of “difficult” is presented in the next section.) A second goal, pursued under the label of

---

<sup>13</sup>In general, such schemes “wrap” at the end of the alphabet, so that 30 (originally question mark) is mapped back to the start of the alphabet (in this case A). Thus, the complete cipher is A becomes B, B becomes C, . . . Y becomes Z, Z becomes space, space becomes comma, comma becomes period, period becomes semicolon, semicolon becomes question mark, and question mark becomes A. If, as in our example, the alphabet has 31 characters, this wrap would be known as “mod 31.”

### BOX C.3 Additional Capabilities Enabled by Cryptography

Cryptographic techniques allow a wide variety of other capabilities, including the following:

- *Secret sharing.* Cryptography enables the division of a secret among  $m$  people in a way that any  $k$  people can reconstruct the secret (for  $k$  less than or equal to  $m$ ), but also in such a way that any combinations of fewer than  $k$  people have no information at all about the secret.
- *Verifiable secret sharing.* A stronger form of secret sharing enables any of the  $k$  people to verify that he or she has indeed received a real part of the secret.
- *Secure function evaluation.* Cryptography enables a function to be evaluated publicly with multiple arguments in such a way that none of the holders of each argument has any knowledge about what the others are holding. One application is electronic voting in such a way that the winner of a vote can be known without forcing any individual to reveal how he or she voted.

These capabilities are less commonly discussed than the fundamental capabilities of enabling confidentiality, signature, and authentication. However, other applications in the future may well rest on them.

cryptanalytic research, is to develop methods and techniques for trying to read messages that have been encrypted by algorithms that may or may not be known to the cryptanalyst.

In symmetric cryptography (or, equivalently, secret-key or private-key cryptography), the encryption key is the same as the decryption key; thus, message privacy depends on the key being kept secret. A major problem faced by Party A is how to inform Party B of the key that is being used. The Data Encryption Standard (DES) is an example of a secret-key cryptographic system.

In one-time pad cryptographic systems, a key is used once and then discarded; the key must be as long as the message. Because an eavesdropper is faced with a constantly changing key pattern that is impossible to break even with exhaustive search, a one-time pad is provably unbreakable provided the key is secure. However, one-time pad systems are difficult to use, and keeping the key secure poses a big problem for key management.

In asymmetric (or, equivalently, public-key) cryptographic systems, the encryption key is different from the decryption key. Message privacy depends only on the decryption key being kept secret. The encryption key can even be published and disseminated widely, so that anyone can encrypt messages. Only the recipient Party B needs the decryption key (which is specific to that party), and Party B never needs to share it with anyone (since only he or she should be able to read messages encrypted

for transmission to Party B). The RSA algorithm is a very popular algorithm at the heart of many asymmetric cryptographic systems. (Box C.4 provides more details on the mathematics of asymmetric cryptography.)

In a key-escrow cryptographic system, the decryption key is made available to parties not directly involved in a given communication—but only under certain circumstances (e.g., under judicial warrant). However, without a complete decryption key, these other parties should be unable to decipher protected communications. The Clipper initiative is a key-escrow proposal for secure telephone communications advanced by the Clinton Administration and described in Chapter 5.

Key management is an integral aspect of all cryptographic systems, which entails (1) the generation of appropriate keys and (2) the distribution of such keys only to the proper parties. Proper and efficient key management is quite complex and is needed to ensure the confidentiality, integrity, and authenticity of the keys used for encryption and decryption. For example, in a symmetric cryptographic system, each user must establish his or her own secret key to use with every other party with whom communication is desired. Thus, for a system of  $N$  users who wish to be able to communicate securely with each other, the number of secret keys that the parties (taken all together) must manage and keep secret is  $N(N - 1)/2$  (i.e., the number of pairs possible with  $N$  parties). When  $N$  is small, the key exchange problem can be handled by personal meetings,

#### BOX C.4

##### The Mathematics of Asymmetric Cryptography

Asymmetric cryptography is based on the putative existence of one-way functions: mathematical functions that are easy to compute but hard to undo. There is no mathematical proof that such functions exist, but there are functions that to date have resisted all attempts to make them easy to undo. One such function is multiplication (its inverse—factoring). It is computationally easy to multiply two prime integers, but in general it is computationally difficult to factor the product. (Computational ease and difficulty refer to the computational resources that are required to perform the task.)

An asymmetric cryptographic system based on factoring would regard the product of the two prime integers as the public key and the two prime integers as the private key. The public key can be made known—once it is known, all of the information about the private key is known in principle too, but it would simply take too long to attempt to compute it.

What does “too long” mean? If the public-key and private-key pair is well chosen, and if in fact multiplication does represent a true one-way function, it means that under no foreseeable circumstances could enough computational power be assembled to perform the factoring in a time shorter than the age of the universe.

Alas, factoring is not provably “hard,” and a variety of techniques have been used in the last decade to drive down the time needed to perform factoring.

but when  $N$  is large, face-to-face meetings as a method for key exchange are impractical.

In many ways, the key management problem is conceptually the same as the cryptographic problem of keeping messages secure, although in practice the key management system usually handles a smaller volume of data, and therefore different methods can be used.<sup>14</sup> Asymmetric cryptographic systems greatly reduce, but do not eliminate, the problem of key distribution. For example, people using an asymmetric cryptographic system can (in principle) publish their public keys in the equivalent of a telephone book that can be distributed freely. Each user must keep track of only  $N - 1$  keys (and can even keep them in a public place with no security), and he or she needs to keep secret only one piece of information—the user's own private key. Note also that the need for face-to-face meetings is eliminated.

Another approach to managing cryptographic keys that does not use asymmetric cryptography is the use of a key distribution center (KDC). A KDC is a trusted agent that knows each user's master key. This master key is employed to exchange session keys for use by users in direct communication. The advantages over link encryption are that only one node is vulnerable to attack (the KDC) and that the users can converse directly (after the initial connection protocol in which both must communicate with the KDC to set up the exchange of session keys).

Note that key management for data communications is very different than for data storage. When encrypted data are communicated, parties have incentives to keep the relevant key only for the duration of the transmission and to eliminate it permanently once the transmission is complete (typically measured in seconds or minutes). When encrypted data are stored, the storing party has a great deal of incentive to retain the key as long as the data may be important (perhaps years or decades).<sup>15</sup>

---

<sup>14</sup>The primary exception to this rule is that keys in a one-time pad are as large as the message itself; thus, the key management system for a one-time pad must be as efficient as the cryptographic system itself.

<sup>15</sup>One practical qualifier is important. Another constraint on data storage entirely apart from encryption is the fact that archived data must in general be copied periodically and rewritten in order to ensure that the then-current generation of technology will be able to access it. For example, in the early days of desktop computing (10 years ago), many computers used 8-inch floppy disks. Today, it is difficult to find an 8-inch floppy disk drive, and data stored on an 8-inch floppy disk would be inaccessible without such a drive. The careful archivist would have to copy the data from the 8-inch floppies to newer media, such as 5 1/4-inch floppies or CD-ROMs. When storage technologies become more capable and widespread (leading to the obsolescence of today's CD-ROM drives), the same copying and rewriting procedure will have to be followed.

*Footnote continues on next page.*

## C.4 ATTACKS ON CRYPTOGRAPHIC SYSTEMS

A cryptographic system involves an encryption algorithm, a decryption algorithm, and keys for encryption and decryption. Although the precise boundaries between algorithm and key are fuzzy, for practical purposes the algorithm can be regarded as whatever in the mathematics is difficult to change, whereas the key is whatever is easy to change.

A basic assumption of cryptographic security is that an eavesdropper knows the relevant decryption algorithm. The algorithm may (or may not) be a public one, but the history of all information secrecy suggests that the best-kept secrets eventually leak. The use of an easily changed key thus enables the continued use of an algorithm that is known to an eavesdropper. (Any added security that results from the fact that an eavesdropper may not in fact know the algorithm is simply a bonus.) Put differently, the security of a cryptographic system should be judged by the security provided by the key alone, even if one attempts to keep the algorithm secret.

To compromise a message, an eavesdropper has two alternatives: to obtain the message in plaintext before it has been encrypted (or after it has been decrypted) or to obtain the ciphertext and decipher it without knowing all that the recipient knows about the decryption algorithm and key. (For the purposes of this appendix, the term "compromise" refers to an eavesdropper intercepting and being able to read the secret message; other types of compromise such as preventing Party B from receiving the message or deliberately garbling it so that even Party B cannot read it are not addressed here.) Although cryptography and cryptanalysis are concerned primarily with the latter, an eavesdropper does not particularly care what methods may be used to obtain the plaintext of a message. Thus, Party A and Party B must ensure that all elements of their commu-

---

Given that periodic rewriting is necessary (e.g., every 10 years), it is natural to ask if it should be the originally encrypted data or the unencrypted-and-then-reencrypted data that should be rewritten. There are advantages and disadvantages to both. Rewriting the originally encrypted data means that it does not need to be decrypted, thus improving possible losses of confidentiality. On the other hand, it also means that the key management system contemporaneous with the originally encrypted data must be preserved for future use. (Specifically, the key management system is responsible for maintaining a record of the key used to encrypt the data so that it can be decrypted later.) Preserving the key management system has many of the same problems associated with it that preserving the older storage media poses. If the choice is made to rewrite unencrypted-and-then-reencrypted data, then the originally encrypted data must be decrypted, which opens another channel for loss of confidentiality.

Different institutions will make this trade-off in different ways, but if the choice is made to rewrite the unencrypted-and-then-reencrypted data, then the time that the original key must be preserved is the time between data rewritings, which may be much shorter than the time the data is of interest.

nications system are secure; if Party A uses a secretary to encrypt the message and the secretary sells the message to an enemy agent, the best encryption scheme in the world does not matter. Similarly, if the eavesdropper is able to intercept the decryption key (e.g., because it was transmitted on an insecure channel or because it too was sold by the secretary), secret messages transmitted with the lost key are vulnerable to compromise. (The fact that Party B must have the decryption key to decrypt the message and that somehow the information identifying the decryption key must be transmitted from Party A to Party B is at the heart of the key interception problem.)

Still, it is often the case that the only real alternative for an eavesdropper is to try to decipher an intercepted ciphertext. How difficult is it for an eavesdropper to accomplish such a task?

The difficulty of cryptanalysis depends on two factors: the size of the key and the mathematical structure of the algorithm itself. Key size determines how long it would take to cryptanalyze the ciphertext by brute force—trying all possible keys with a given decryption algorithm until the (meaningful) plaintext appears.<sup>16</sup> With a sufficiently long key, even

---

<sup>16</sup>Strictly speaking, this statement is true for symmetric cryptography involving algorithms such as DES with a larger key and a few other minor modifications to make it stronger. With asymmetric cryptography, the difficulty of the problem rests in knowing the computational effort needed to invert certain functions (e.g., factoring). For more discussion, see Section C.5. For these reasons, the comment of Edgar Allen Poe (Edgar Allen Poe, *The Gold-Bug*, Creative Education Inc., Mankato, Minn., 1990, p. 63) that “it may well be doubted whether human ingenuity can construct an enigma of the kind which human ingenuity may not, by proper application, resolve” is exactly wrong—there is every reason to believe that it is possible to devise an impenetrable cipher. (The one-time pad is such an example.)

Quantitatively, the effort to encipher and decipher (i.e., with knowledge of the key) in conventional systems is almost independent of the key size (and sublinear in any event). For example, both RC2 and RC4 have key initialization routines that take the variable-length key and expand it into a larger “expanded key” used in the encryption and decryption process. Since the key initialization is done only once per key, it adds a fixed overhead, which is negligible in most applications because the expanded key is used to encrypt large amounts of data before the key is changed. Cryptanalysis, on the other hand, appears to be exponential in the key size ( $2^b$ , where  $b$  is the number of bits).

The bottom line is that cryptanalysis grows exponentially in  $b$ , while enciphering and deciphering grow at worst linearly in  $b$ —a very nice work factor for the cryptographer, but an awful situation for the cryptanalyst.

Asymmetric cryptographic systems are more complex. The best-known algorithms provide cryptanalytic attacks that grow as  $\exp[c \cdot b^{1/3} \cdot \ln(b)^{2/3}]$  (where  $c$  is a constant equal to approximately 1.7) while enciphering and deciphering grow as  $b^3$ .

Finally, one important operational caveat for both asymmetric and symmetric systems is that one must be able to recognize the output as meaningful before one can know that the key just tested was indeed correct. When the plaintext is an English sentence, it is possible to look at the resulting sentence and recognize it for what it is. However, if the “plaintext” is in fact a computer program or an image file, it may be much more difficult to recognize the output as being correct.

an eavesdropper with very extensive computing resources would have to take a very long time (longer than the age of the universe) to test all possible combinations. On the other hand, practical considerations related to implementation issues may force a trade-off between overall security (of which key size is one element) and cost.<sup>17</sup>

---

<sup>17</sup>A relevant issue is that computers can be expected to grow more powerful over time, although there are fundamental limits on computational capability imposed by the structure of the universe (e.g., nothing travels faster than light in vacuum, and the number of atoms in the universe available to build computers is large but finite). Thus, the minimum key size needed to protect a message against a very powerful opponent will grow as computers become more powerful, although it is certainly possible to choose a key size that will be adequate for protecting against exhaustive search for all time.

Thus, although it is true that dramatic reductions in the cost of computing (or equivalently, increases in computational power) have occurred in the past four decades, it does not follow that such reductions in cost or increases in power can continue indefinitely. The commonplace belief or instinct that they can continue indefinitely is simply wrong.

What is true is that fundamental limits to computation have not yet been reached and will not be reached for a long time, but this is a result of the fact that early computational devices were so far from the fundamental limits of computation that many orders of magnitude improvement have been possible. Two illustrative calculations demonstrate the fact that there are practical limits:

1. *A limit based on the energy output of the sun.* All real computations consume energy. On the basis of standard thermodynamics and statistical mechanics, the energy cost of an irreversible computation must be on the order of  $kT$ , where  $T$  is the ambient temperature (on an absolute Kelvin scale) and  $k$  is Boltzmann's constant (equal to  $1.4 \times 10^{-23}$  joules per degree Kelvin). The sun's power output is approximately  $3.86 \times 10^{26}$  watts; thus, its total energy output over its expected lifetime of 10 billion years ( $3 \times 10^{17}$  seconds) is about  $10^{44}$  joules. Assume an ambient temperature of  $T = 10^{-6}$  degrees, which will then impose an energy cost per operation of  $1.4 \times 10^{-29}$  joules per operation. Thus, the number of computational operations possible using the entire energy output of the sun is given by energy output divided by the energy cost per operation, or about  $10^{73}$  operations. If only one operation were necessary to test a key (in practice, hundreds are necessary), then it would take  $10^{73}$  operations to test a key of 73 decimal digits (which is equivalent to about 250 binary bits). For reference, the number of atoms in the solar system is about  $10^{60}$ .

2. *A limit based on the mass of Earth.* The mass of Earth is about  $6 \times 10^{24}$  kg. A proton mass is  $1.6 \times 10^{-27}$  kg, so that Earth contains about  $4 \times 10^{51}$  protons. Assume one proton per computer, and that each computer can perform one operation in the time that it takes light to cross its diameter (i.e.,  $10^{-15}$  meters divided by  $3 \times 10^{10}$  meters per second, or  $1/3 \times 10^{-25}$  seconds). Each computer can thus perform  $3 \times 10^{25}$  operations per second. If all of these computers work in parallel, they can perform  $4 \times 10^{51} \times 3 \times 10^{25}$  operations per second, or  $10^{77}$  operations per second. The age of the universe is on the order of 10 billion years, or  $3 \times 10^{17}$  seconds. Thus, an earthful of proton-sized computers can perform  $3 \times 10^{94}$  operations in the age of the universe. With the assumptions made before, this corresponds to a key size of 95 decimal digits, or about 320 bits.

Both of these calculations demonstrate that it is clearly possible to specify a key size large enough to guarantee that an attack based on exhaustive search will *never* be feasible, regardless of advances in conventional computational hardware or algorithms. (The qualification to "conventional" computing is for quantum computing, discussed in Section C.6.6.)

Nevertheless, because the cost of brute-force cryptanalysis doubles for every bit that is added to the length of a key, there is a broad consensus among cryptographers that it is possible today to encrypt data very inexpensively in a way that would be unbreakable through brute force in the foreseeable future, regardless of advances in computing technology that could be used for cryptanalysis. Put differently, for some sufficiently long key length, the possibility of brute-force cryptanalysis can be ruled out categorically for all time. In practice, "sufficiently long" may turn out to be a key length as short as 168 bits. (Of course, this analysis does not address those operational situations, encountered from time to time, in which the time required to encrypt plaintext must be kept to a minimum; in such situations, the operational performance requirements of a system may preclude the use of such a long key. Nevertheless, in many situations, the operational requirements are not quite so critical, and the system implementer can use very long key lengths without an impact on performance.)

The algorithm itself may provide an alternative to exhaustive search: a weakness in the algorithm, if exploited by an opponent, may categorically rule out certain keys, thereby reducing the number of keys that need to be tested. Such weaknesses may be introduced deliberately (resulting in a "trapdoor" that allows someone with such knowledge to decipher a message secretly) or may be accidental (perhaps as the result of insufficient analysis).

Several attack scenarios are possible for the eavesdropper:

- *Ciphertext only.* If the eavesdropper has only the intercepted ciphertext and nothing else, it may well be impossible to recover the plaintext. This is the least advantageous for the eavesdropper; however, judgments about the security of a system should not be made on the basis of this assumption since Party A and Party B may not know that this condition obtains.

- *Known plaintext.* The eavesdropper may have the intercepted ciphertext (call it C1) and some other ciphertext (C2), as well as the plaintext (P2) corresponding to C2. (For example, C2 may be known to be an encrypted press release that is then published by Party A the day after interception.) If the eavesdropper has reason to believe that C1 (the ciphertext of the message of interest) has been produced by the same algorithm and key, he or she may be able to derive the decryption key with much less work than by exhaustive search or the case in which only C1 is available (i.e., the ciphertext-only case).

- *Chosen plaintext.* A variant of the known plaintext attack is the *chosen plaintext* attack, in which the eavesdropper has been able to insert words of his or her own into P2. (An attack of this sort characterized U.S.

Navy intelligence just before the Battle of Midway in World War II.<sup>18</sup>) By controlling the plaintext, the work of the eavesdropper is eased significantly (because test cases can be generated easily, among other things).

Once the eavesdropper learns the decryption key, he or she can decipher easily any subsequent message that uses that key.

Note that much of the public debate about the ease or difficulty of breaking an encryption scheme is carried out in terms of an “ideal” implementation of a given algorithm. However, in actual practice, cryptanalysts (those trying to break encryption schemes) exploit weaknesses in the way an algorithm is implemented in practice. For example, the protection afforded by algorithm X may require the use of random numbers. However, it may turn out that the way in which system A implements algorithm X does not use true random numbers but rather numbers with a predictable sequence (e.g., consecutive numbers, or even worse, a fixed number such as zero or one).<sup>19</sup> A cryptanalyst who suspects that this might be true about someone who uses system A to protect communications may be able to exploit it and therefore reduce by orders of magnitude the effort required to decipher those communications. Put differently, any cryptographic system that relies on keys has a built-in vulnerability with respect to the key. The encryption may be virtually invulnerable, but the key is always vulnerable. Even if the key is ultimately divided between multiple parties, the place at which the key is generated is always a potential vulnerability.

Strong cryptography refers to cryptographic systems that are very difficult to break. Eavesdroppers with large amounts of time, money, and computing expertise (e.g., national governments) are in a much better position to break cryptographic systems of a given strength than are those with more limited resources (e.g., individuals or corporations). Organized crime may also be in a good position to obtain cryptanalytic intelli-

---

<sup>18</sup>To confirm a cryptanalytic solution, U.S. codebreakers asked the American garrison at Midway to report over an open and unsecured channel a shortage of fresh water. The Japanese, monitoring this channel, reported two days later that “AF” was experiencing a shortage of fresh water, thus confirming that “AF” was indeed the Japanese code designation for Midway. See David Kahn, *The Codebreakers: The Story of Secret Writing*, MacMillan, New York, 1967, p. 569.

<sup>19</sup>An analogy will illustrate. Computer users must often “sign on” to their computers using a secret password that cannot be guessed easily. However, it is quite common to find computer users who use passwords such as their name or some easily remembered (and therefore easily guessed) word. A person (or a computer) trying to guess passwords is obviously in a much better position if the search can be limited to all eight-character words in the dictionary and all proper names (analogous to numbers with a predictable sequence), rather than all possible combinations of eight characters (analogous to random numbers).

gence because it is able to bring large sums of money to bear on the problem if the results are worth even more.

An interesting technical question is the extent to which it is possible to build very strong cryptographic systems with no algorithmic weaknesses whose decryption keys are sufficiently large to preclude exhaustive search as an effective method of decryption. If such systems are possible, a user of such systems can, by definition, be assured that no eavesdropper can break that encryption system.

Finally, the role of operational errors in the penetration of even well-designed cryptographic systems should not be underestimated. Penetration is often possible because the user of the cryptographic system has made a mistake that compromises its security. One example that has recently come to light is the successful decryption of certain messages sent by Soviet agents in the United States regarding nuclear weapons and the U.S. nuclear program at the end of World War II. Soviet agents used a one-time pad; when used properly, a one-time pad is known with mathematical certainty to be impenetrable (as described above). However, a one-time pad is based on the idea that a certain sequence of random numbers serving as the encryption key to a message will never be used more than once. For some time, Soviet agents used a particular one-time pad to encode messages, and American analysts were unable to decipher them. However, the time came when Soviet agents began to *reuse* numbers from the one-time pad, and American cryptanalysts were able to make substantial headway in deciphering them.<sup>20</sup>

## C.5 ELEMENTS OF CRYPTOGRAPHIC SECURITY

To keep eavesdroppers from compromising secret messages, security experts may take several approaches. By far the most common approach is to change the key frequently, although in practice the problem of key distribution may complicate matters considerably if a private-key system is used. A less frequent (though still common) technique is to encrypt a message multiple times through the same algorithm with different keys; an approach based on multiple encryption using DES has been proposed as an alternative to the Skipjack encryption-decryption algorithm. (Skipjack is the name of the algorithm on which the Clipper chip is based.)

Security experts may also attempt to keep the algorithm secret. Keeping algorithms secret has advantages and disadvantages. The advantage is that when an algorithm is kept secret, fewer people have the opportu-

---

<sup>20</sup>George Johnson, "The Spies' Code and How It Broke," *New York Times*, Week in Review, July 16, 1995, p. 16.

nity to learn its potential weaknesses; thus, information about its weaknesses can be less widespread should any have been overlooked. In addition, keeping algorithms secret is a way to keep out of the public domain information on what constitutes a good algorithm. The disadvantage is the flip side to the same coin—when fewer people can learn its weaknesses, an algorithm may have vulnerabilities that go undetected by its users and, thus, may be vulnerable to clandestine compromise.

Finally, in principle, it is possible to vary the algorithm as well. However, it is very difficult to develop a trusted algorithm, with the result that algorithms are changed rarely, and the number of useful algorithms is much smaller than the number of keys possible when even a small key is used.

To summarize, the fundamental question in evaluating cryptographic systems is how long the system as a whole takes to become obsolete or whether it will defy obsolescence. The algorithms and techniques for key generation and management are important, but it is a mistake to focus exclusively on these matters. A cryptographic system may well become obsolete in a given environment even though its mathematical foundations remain sound. Extending the time to obsolescence may be desirable and necessary, but no system can be extended indefinitely. The continual evolution of cryptographic techniques and the use of redundant systems are as important to security as the mathematical correctness of an algorithm and the size of an encryption key.

## C.6 EXPECTED LIFETIMES OF CRYPTOGRAPHIC SYSTEMS

Because of the rapidly decreasing cost of computation, cryptographic systems that cost \$1 billion to break in 1945 can be broken for approximately \$10 today. In the same way, today's cryptographic systems should have large safety margins to protect against future advances in technology.

The need for safety margins varies, depending on the data entrusted to the cryptographic system. Press releases, encrypted during transmission for later release, typically need at most a few days of secrecy. Medical records, on the other hand, can have privacy time constants on the order of 50 years. Because a company or governmental agency typically uses a single cryptographic system to protect all of its data, ideally the system should have a safety margin commensurate with the longest privacy time constant encountered.<sup>21</sup>

Symmetric cryptographic systems allow large safety margins at low

---

<sup>21</sup>See also footnote 15.

cost. Asymmetric cryptographic systems have a more pronounced relationship between cost and safety margin, so that it is harder to achieve large safety margins with asymmetric systems. Even with conventional systems, where 50-year safety margins appear possible and cost-effective, national security and related export considerations may prevent their use.

### C.6.1 Background

The need for safety margins stems from two general categories of technological advances: those due to improvements in computation and those due to breakthroughs in cryptanalytic attacks.

Safety margins needed to protect against improvements in computation are easier to predict because there is a steady trend, manifest since the 1930s, that is expected to continue for the next few decades and probably beyond, in which the cost of computation has decreased by an order of magnitude (i.e., a factor of 10) every 5 to 7 years (e.g., Moore's law predicts, so far fairly accurately, that microprocessor speed doubles every 18 months, equivalent to a factor of 10 every 5 years). Consequently, a computation that costs \$1 billion today may cost only \$10 in 50 years.<sup>22</sup> Since some of the information entrusted to cryptographic systems has a privacy time constant of 50 years and more (e.g., medical records should be private for at least the duration of the patient's life), it is seen that a significant safety margin is needed.

Advances of the second type, breakthroughs in cryptanalysis and related techniques, are much harder to predict. In the case of symmetric cryptographic systems, there is little public literature to use as a guide, but asymmetric cryptographic systems offer some data points, and so they are treated first.

### C.6.2 Asymmetric Cryptographic Systems

The asymmetric cryptographic systems in primary use today base their security on the difficulty of two related computational problems: factoring integers and finding discrete logarithms.<sup>23</sup> Factoring is used as

---

<sup>22</sup>This assumes that Moore's law will continue to hold. Today, the technology of silicon electronics does not run up against fundamental physical constraints, but whether the Moore's law trend will continue to hold for 50 years is open to debate. Most experts suggest that for a decade or two, it will probably remain valid.

<sup>23</sup>Although it is not needed to understand what follows, these two problems can be explained easily. In factoring, one is given an integer, for example 493, and asked to find all prime factors. Since  $493 = 17 \times 29$ , the answer here is "17 and 29." In discrete logarithms, one is given  $a$ ,  $n$ , and  $y$  in the equation " $a^x$  modulo  $n = y$ " and asked to find  $x$ . For example, a solution to " $2^x$  modulo  $11 = 10$ " is  $x = 5$ . To see this, note that  $2^5 = 32$  and  $32$  modulo  $11 = 10$ .

the example in what follows because it is the more studied of these two problems.

For many years, the progress of factoring was measured by the progress in factoring what are known as the Fermat numbers, denoted by  $F_n$ , the  $n$ th Fermat number. The  $n$ th Fermat number is  $2^{(2^n)} + 1$  (where  $\wedge$  denotes exponentiation). Hence,  $F_2 = 2^4 + 1 = 17$ ,  $F_3 = 2^8 + 1 = 257$ , etc. In general,  $F_n$  is an  $n + 1$  bit number. Increasing  $n$  by 1 doubles the size of the number to be factored, when measured in bits. In recent history,  $F_7$  was factored in 1970,  $F_8$  in 1980,  $F_9$  in 1990, and  $F_{10}$  in 1995.

It is interesting to note that  $F_9$  was factored by an algorithm that is much faster for "special" numbers such as  $2^{(2^n)} + 1$  than for "general" numbers used in asymmetric cryptographic systems. This was not true of earlier factoring methods. Hence, although the history of the Fermat numbers can be used to illustrate the history of factoring as applied to asymmetric cryptographic systems, the future does not allow that correspondence. (More precisely,  $F_8$  was factored with a method that is not applicable to breaking asymmetric cryptographic systems. However, another factoring method that is applicable to breaking asymmetric cryptographic systems was being tested at about the time and would have been successful in factoring  $F_8$  in either 1980 or the next year.) Also, the factoring of  $F_9$  involved a large network of workstations, connected over the Internet and using idle time. This networking reduced by several orders of magnitude the time needed to undertake the relevant computations. Table C.1 provides a historical record of the factoring of "nonspecial" numbers.

Some of the advances in factoring Fermat numbers were due to the decreasing cost of computation, which fell by approximately a factor of 100 in each 10-year period. However, most of the improvement was due to breakthroughs in factoring algorithms. For example, the continued fraction method, used successfully to factor  $F_7$  in 1970, would have taken approximately a million times as much effort to factor  $F_8$ , or 10,000 times as long in 1980, given the factor-of-100 speedup in computers. In contrast,

TABLE C.1 A History of Factoring

Year	Size of Number Factored
1964	20 decimal digits (66 bits)
1974	45 decimal digits (149 bits)
1984	71 decimal digits (236 bits)
1994	129 decimal digits (429 bits)

SOURCE: Andrew Odlyzko, "The Future of Integer Factorization," *Cryptobytes*, RSA Laboratories, Redwood City, Calif., Volume 1(2), Summer 1995, p. 5.

the quadratic sieve, developed in the late 1970s, cut the required computation by a factor of roughly 100,000 when compared to continued fractions. Qualitatively similar numbers apply to the improvements that allowed  $F_9$  to be factored in 1990.

The data points from the factoring of Fermat numbers give an estimate that key size (the size of the number to be factored) must double every 10 years to keep up with improvements in factoring. This, in turn, implies that key size must have a safety factor of 32 to be secure over 50 years (five periods of 10 years, resulting in a key size increase of  $2^5 = 32$ ). This estimate is very approximate, and probably conservative, because the development of asymmetric cryptography gave a tremendous impetus to the study of factoring. Mathematicians working in what had been one of the purest of pure areas of mathematics, with little attendant funding, could suddenly point to immense commercial benefits from their work. Also,  $F_7$  and  $F_8$  were factored on single-processor machines, whereas the factorization of  $F_9$  made use of the idle time on a network of approximately 700 workstations scattered around the world, and such an advance in computational power can come only once. A less conservative estimate would therefore be to assume at least one, and probably two, additional breakthroughs that double the size of the numbers that can be factored.

The above discussion points to a need for a safety factor of 2 to 32 in the length of the key for asymmetric cryptographic systems, an admittedly large range of uncertainty. This ambiguity is sometimes eliminated by real-world considerations. If, for example, there are significant idle computational resources available for public-key computations and they can be done in background mode without delaying current communications, then a safety margin of a factor of 32 in key size is entirely reasonable and should be used. On the other hand, if the computation to use a factor-of-four margin in key size results in unacceptable delay, one might use a factor-of-two margin, or no safety margin at all, particularly if the data has low value and a short privacy time constant. Export considerations also might limit key size, but in these latter cases users need to be aware of the danger to their communications, so that they do not trust valuable data to a system with an inappropriately low safety margin.

Today, factoring 512-bit numbers is extremely difficult, while factoring 1,024-bit numbers is computationally impossible. By using 512 bits as a reasonable security level for asymmetric cryptographic systems whose data must be secret only in the immediate future, a safety margin of 2 (the minimal indicated) would dictate the use of 1,024-bit numbers, while a safety margin of 32 (a much more conservative and safer value) would lead to roughly 16-kilobit numbers. The public-key algorithms in use today have a cost of computation that grows with  $b^3$ , where  $b$  is the num-

ber of bits. Hence, a safety margin of a factor of 32 in key size requires an increase in cost of  $32^3$  (more than 30,000), an uneconomic situation in most applications. At the larger bit sizes, more efficient arithmetic methods can be used that might reduce the growth curve to approximately  $b^2$ , but even  $32^2 = 1,024$  is a larger cost penalty than most users will be willing to pay.

### C.6.3 Conventional Cryptographic Systems

DES is the most widely studied conventional cryptographic system, and so it is used here for illustrative purposes in assessing the security levels needed in such systems. The best known *practical* method for breaking DES is exhaustive search of all  $2^{56}$  possible keys. The correct key can be recognized because it decipheres intercepted ciphertext into meaningful plaintext. In 1977 Diffie and Hellman estimated the cost of exhaustive search at \$10,000 per key.<sup>24</sup> Their estimate would scale to a cost of at most \$100 per solution in 1994, 14 years (two periods of 7 years) later.

This figure of \$100 per solution is also supported by recent work of Wiener.<sup>25</sup> Using commonly available components, Wiener estimated that he could build exhaustive search machines for \$1 million each, which could produce a DES key every 3.5 hours. Amortizing machine cost over 5 years results in a cost of \$80 per solution. Although this estimate neglects costs such as interest, design, maintenance, electricity, etc., these additional costs do not affect the estimated cost because it is only a "ballpark" estimate, accurate to at best a factor of two. More accurate estimates are not needed because of the rapidly decreasing cost of computation: an error by a factor of two is erased in 1 to 2 years. These numbers might make it seem that DES reached the end of its useful life some time ago. That is partly true and partly false for the reasons explained below.

The approximately \$100 cost per solution assumes an opponent is willing to invest several million dollars in the design and production of exhaustive search cryptanalytic machines. Exhaustive search on general-purpose computers is much more expensive, costing on the order of \$10 million per solution. Hence, DES is insecure against opponents who can afford to build special-purpose cryptanalytic machines, have enough problems to keep them fully loaded (idle time increases the cost per solution), and have access to modern integrated circuit technology. National intelligence organizations within the developed world meet all of these

---

<sup>24</sup>Whitfield Diffie and Martin Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," *Computer*, June 1977, pp. 74-84.

<sup>25</sup>M.J. Wiener, "Efficient DES Key Search," TR-244, School of Computer Science, Carleton University, Ottawa, Canada, May 1994; presented at the Rump Session of Crypto '93.

criteria. National intelligence organizations in less developed nations and organized crime possess the budget, but export restrictions and other government controls on cryptographic technology raise the question of whether they could purchase the required technology. Large corporations also pose a potential threat to DES, but the difficulty of hiding a several million dollar budget plus government controls make them less likely threats. Hence, DES is relatively secure today against industrial espionage, extremely insecure against major foreign powers, and questionable against lesser foreign powers and organized crime (which has no qualms about hiding budgets or conspiracies). DES's useful lifetime against commercial adversaries is on the order of 15 years, which could bring the \$10 million per solution on general-purpose hardware down to \$10,000 per solution, an amount that many individuals could afford.

Advances in cryptanalysis could speed the obsolescence of DES, but there are few historical data on which to base such an estimate. Prudence would dictate doubling the key size over what is indicated by current algorithms, especially since exhaustive search has been assumed in the above analysis. The frequently proposed triple-DES, which uses three DES devices in series with three different keys, more than meets this requirement and does not require any new standards. It does, however, meet with real-world problems since even single-DES is subject to U.S. Munitions List controls.

Unlike asymmetric cryptographic systems, the cost of increasing the key size of DES, or of most other conventional cryptographic systems, is minimal. Again, for illustrative purposes, DES has a 56-bit key that is expanded into a 768-bit pseudokey for use by the algorithm. Aside from the increased storage required, a 768-bit key could be used with a minimal penalty in the speed of computation. Since storing the 56-bit key consumes less than 10% of DES's required storage, doubling the key size results in at most a 10% increase in encryption-decryption cost.<sup>26</sup>

---

<sup>26</sup>In fact, a small increase in encryption time would occur, because if the DES algorithm is adapted to use a larger key size, it would also be advisable to increase the number of rounds (iterations), thus increasing the encryption-decryption time. For example, obtaining the full benefit of a 128-bit DES key would require approximately doubling the number of rounds, with an attendant doubling of computational time. Although this increase in time would be a problem in some applications, in many others it would not (e.g., telephone line communications where speeds are relatively slow). In any event, the rate of increase of computational time (as security is increased) is much slower in symmetric systems such as DES than in asymmetric systems.

### C.6.4 Timing Attacks

A different type of attack against a number of cryptographic systems has been developed by Paul C. Kocher, an independent consultant.<sup>27</sup> Kocher's attack differs from traditional cryptanalysis in that it needs additional information on the time required by each encryption, decryption, or signing. However, it often works even when only known ciphertext is available. Although such an attack is harder to mount than a ciphertext-only attack (see definitions above), in most applications it appears comparable in difficulty to obtaining known plaintext and in most applications is no harder than mounting a chosen text attack. While the attack can thus be mounted in only a small fraction of cases, good cryptographic practice requires treating such attacks seriously. Good business practice also dictates this approach because it takes only one large loss to result in a loss of confidence or money.

Kocher's attack makes use of his insightful observation that the computation time of many systems depends in a predictable manner on the first bit of the secret key. By computing the two running times (when that bit is 0 and when it is 1) for a large number of observed computations and correlating them with the observed computation time, the attacker can make a good guess on the first bit of the secret key. If this guess is correct, the computation time depends in a predictable manner on the second bit of the secret key, which can be attacked in like manner, etc. Any errors in early decisions result in poor correlations that signal the error and invite revisiting the decision.

Although his results are very recent and therefore somewhat preliminary, Kocher has estimated that on the order of 1,000 computation times are sufficient to attack many software implementations of DES, RC5, RSA, Diffie-Hellman, and the Digital Storage Standard (DSS). He is investigating the applicability to other systems as well.

One obvious fix to this problem is to implement fixed-time-length encryptions to conceal variations in the encryption times. Of course, such a fix would also run counter to the often-present desire to minimize computational delay.

---

<sup>27</sup>See Paul Kocher, *Cryptanalysis of Diffie-Hellman, RSA, DSS, and Other Systems Using Timing Attacks*, Stanford, Calif., December 7, 1995; available on-line at <http://www.cryptography.com/timingattack.html>. A popular account of this attack is found in John Markoff, "Secure Digital Transactions Just Got a Little Less Secure," *New York Times*, December 11, 1995, p. A1.

### C.6.5 Skipjack/Clipper/EES

The Skipjack encryption algorithm used in the Escrow Encryption Standard (EES; "Clipper") has an 80-bit key size. Since the algorithm itself is classified and made available only in silicon, exhaustive search cannot be contemplated by other than the U.S. government until the algorithm is reverse-engineered. Many deem that likely to happen within 5 to 10 years, perhaps even sooner at foreign national intelligence organizations. Alternatively, the algorithm may have to be divulged to such organizations if EES is to become international in scope—a prerequisite to its being widely used in this country since so much business is international in nature. For all these reasons, in what follows, the prudent assumption is made that the algorithm is known to an adversary.

Since Skipjack has a key size that is 24 bits larger than DES, exhaustive search takes  $2^{24}$  (16 million) times as long and costs 16 million times as much. The \$100-per-solution cost of DES thus scales to approximately \$1 billion per solution. (Although 16 million times \$100 equals \$1.6 billion, the use of more than an order-of-magnitude estimate would give a misleading impression of the accuracy of these estimates.) Skipjack is thus immune to exhaustive search for some time to come. If a cost of \$1 million per solution is used as ending the utility of a system, Skipjack's key size has a safety factor of 1,000, which will be erased in 15 to 21 years because of the decreasing cost of computation (three periods of 5 to 7 years).

If Skipjack is considered usable even at \$1,000 per solution, that adds another 15 to 20 years to its useful life, for a total of 30 to 40 years. The figure of \$1 million per solution is appropriate since some data will be worth that much to an opponent. Again, any cryptanalytic improvements over exhaustive search would decrease the lifetime of Skipjack. In summary, Skipjack's key size possesses a larger margin of safety than single-encryption DES, but that margin is smaller than would be dictated by purely economic and technical considerations. (As with DES, increasing the key size of Skipjack does not greatly increase the computation cost.)

### C.6.6 A Warning

When issues related to potential weaknesses are raised, the argument is often made that when a system becomes weak, it can be replaced by a stronger one. The implied question is, Why use more security now than is needed? Although this argument makes sense for some cryptographic applications, in many cases it is wrong, given that a standard is intended for universal use.

The argument is correct for applications—such as tactical military or commercial plans—in which an opponent gains value only by cryptanalyzing the system soon after the data have been encrypted. But strategic plans, as well as medical records and many other forms of individual and corporate data, have long privacy time constants. When the old cryptographic system for such data is in danger of compromise, it does not help to reencrypt the data in a new, stronger cryptographic system: an opponent who has recorded and stored the data encrypted in the old system can attack the old, weaker cryptographic system used to encrypt the stored data.

### C.6.7 Quantum and DNA Computing<sup>28</sup>

Two recent computing proposals may fundamentally alter the above analysis. Shor has proposed using quantum computing to factor integers.<sup>29</sup> Although such computing requires technology far beyond that available today, if it could be implemented, it would reduce factoring and discrete logs to easy problems and kill the currently most popular public-key cryptographic systems. Quantum computing is still embryonic, and it is not clear whether it will be practical.

Quantum computing is computing that is based on the properties of quantum mechanical systems. In classical computing, a bit is either 0 or 1. However, a fundamental property of quantum mechanical systems (such as single quantum particles) is that they can exist in a “superposition” of states, fractionally both 0 and 1. A properly coupled set of  $L$  quantum bits (or “qubits”) can hold not just one value out of the total  $N = 2^L$  possible values, but can in principle contain *all* such values simultaneously. If logical operations are now performed—and the laws of quantum mechanics do allow such operations—then computations can be performed simultaneously and in parallel on all the represented numbers.

Using these concepts, Shor was able to find a quantum algorithm that can, in principle, find the prime factors of a number  $N$  in a time propor-

---

<sup>28</sup>Material in this section is based on two JASON reports, one on quantum computing called *Boundaries of Computing*, and the second called *DNA Computing* (A. Despain et al., *Boundaries of Computing*, JASON Study Report JSR-95-115, MITRE Corporation, McLean, Va., September 19, 1995; N. Lewis and P. Weinberger, *DNA Computing*, JASON Study Report JSR-95-116, MITRE Corporation, McLean, Va., September 12, 1995). A lay exposition of quantum computing is contained in Seth Lloyd, “Quantum-Mechanical Computers,” *Scientific American*, October 1995, pp. 140-145.

<sup>29</sup>Peter Shor, “Algorithms for Quantum Computation: Discrete Logarithms and Factoring,” in Shafi Goldwasser (ed.), *35th Annual Symposium on Foundations of Computer Science: Proceedings*, IEEE Computer Press, New York, 1994.

tional to  $L$ , the number of bits of that number, raised to some power (i.e., in polynomial time). No factoring algorithm implementable on a classical computer is known that can factor a number with so few steps; all known classical factoring algorithms are at best barely subexponential in the number of bits. Quantitatively, given the number  $N$  and  $L = \log_2 N$ , the quantum algorithm can factor  $N$  in a time proportional to  $L^k$ , where  $k$  is some number; all known classical algorithms give times that are worse than this time.

It must be emphasized that it is not known today how to build a quantum computer that could execute a quantum algorithm. Indeed, while individual qubits have been created and manipulated in the laboratory, no basic circuit has yet been constructed for a quantum computation, let alone a full-up computer.<sup>30</sup> It has been estimated that a quantum computer that could solve cryptographically interesting problems would have a minimum of about  $10^{11}$  quantum logic gates.

Nor is it known how broad is the class of number-theoretic problems that can be speeded up with a quantum computer. Shor's factoring algorithm makes a very special use of the fast Fourier transform as a key step. It is possible that some other computationally difficult problems on which cryptographic systems could be based are not susceptible to this trick and are equally hard for quantum computers. This is a fascinating and lively area of current research.

DNA computing is another recently described paradigm for massively parallel computation. The basic idea is that DNA strands in a test tube can be used to encode all possible answers to a given problem, such as a cryptanalytic solution to a given piece of ciphertext encoded with a known algorithm. Biochemical techniques are known for sorting out different strands of DNA; these techniques are logically equivalent to the execution of an algorithm to obtain only the strands of DNA that represent the correct answer(s) to the problem. The power of DNA computing lies in the ability to prepare and sort through a compilation of all possible answers to problems of a given computational complexity.

A small computational problem has indeed been solved by the use of a DNA computer.<sup>31</sup> This successful demonstration puts DNA computing on a much firmer foundation than quantum computing. However, DNA computing does not fundamentally change the hard nature of cryptanalytic problems, such as factoring or breaking DES; it merely changes the cost of the computation. At this time, it is not clear if DNA computing for

---

<sup>30</sup>See David DiVincenzo, "Quantum Computation," *Science*, Volume 270(5234), October 13, 1995, pp. 255-261.

<sup>31</sup>Leonard Adelman, "Molecular Computation of Solutions to Combinatorial Problems," *Science*, Volume 266, November 11, 1994, pp. 1021-1024.

cryptanalysis will be more or less expensive than electronic computing. If DNA cryptanalytic machines can be built more cheaply than electronic ones, they will require those concerned with information security to adopt larger safety margins in their encryption schemes (e.g., larger keys) than they previously envisioned.

An approach has been described for using DNA computing to break DES that would require about 4 months of reaction time and 1 gram of DNA to succeed.<sup>32</sup> Since current laboratory techniques use only micrograms, or at most milligrams, of DNA, actually implementing this approach today would probably be a multimillion dollar project, and it would reveal only a single DES key.

More relevant to the future is the fact that the amount of DNA required is exponential in the size of the problem. That is, attempting the decryption problem on a message encoded with a 57-bit key would require twice the amount of DNA required for the comparable decryption problem with a 56-bit key. An 80-bit decryption (required for Skipjack) would require 16 million grams (16 tons) of DNA. Thus, over the long run, it does not appear that even the massively parallel nature of DNA computing will be able to overcome the ease with which key sizes can be increased.

### C.6.8 Elliptic Curve Cryptographic Systems

Variants of the RSA and Diffie-Hellman asymmetric cryptographic systems have been proposed that use elliptic curves instead of modular multiplication as the fundamental group operation. Today the elliptic curve variants have the advantage that the best-known algorithms for cryptanalyzing them have computational requirements that grow exponentially in the size of the modulus, as opposed to subexponential behavior for RSA and Diffie-Hellman. If this exponential behavior continues to hold, asymmetric cryptographic systems can have significant safety margins, comparable to those obtainable with conventional cryptographic systems, without undue economic or time cost to legitimate users. Caution is warranted, however, since the elliptic curve systems are fairly recent and therefore not nearly as well studied as RSA and Diffie-Hellman.

### C.6.9 Quantum Cryptography

Certain techniques based on fundamental quantum mechanical properties of physical systems can be used to perform key exchange between two parties that have never met, who share no a priori secret information,

---

<sup>32</sup>See Dan Boneh, Christopher Dunworth, and Richard J. Lipton, *Breaking DES Using a Molecular Computer*, Technical Report CS-TR-489-95, Princeton University, Princeton, N.J., 1995.

to enable them to communicate in absolute privacy.<sup>33</sup> In particular, the laws of quantum mechanics allow two particles (such as photons of light in a fiber-optic cable) to be put in a state of "entangled" information. In such a state, any measurement of one of the particles necessarily disturbs the entanglement. Thus, eavesdropping on a quantum channel used to communicate a key will inevitably be detected by the intended recipient of the key, at which point a new key can be transmitted.

A working quantum cryptography apparatus has been developed, although the sending and receiving mechanisms are only 30 centimeters apart. The creators of this apparatus<sup>34</sup> believe that nothing in principle limits the technique from being used over much greater distances. At the same time, they note that quantum key distribution must compete with classical techniques for key exchange, which are much cheaper over long distances.

---

<sup>33</sup>The description in this subsection is taken from Charles Bennett et al., "Quantum Cryptography," *Scientific American*, Volume 267(4), October 1992, pp. 50-57.

<sup>34</sup>Bennett et al., "Quantum Cryptography," 1992.