principals concerning the authenticity of a message transmission. This general model shows that there are four basic tasks in designing a particular security service: 1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose. 2. Generate the secret information to be used with the algorithm. 7 Chapter 3 discusses a form of encryption, known as asymmetric encryption, in which only one of the two principals needs to have the secret information. Figure 1.5 Model for Network Security Information channel Security-related transformation Sender Secret information Message Message Secure message Secure message Recipient Opponent Trusted third party (e.g., arbiter, distributer of secret information) Security-related transformation Secret information 1.8 / A Model for Network Security 41 3. Develop methods for the distribution and sharing of the secret information. 4. Specify a protocol to be used by the two principals that make use of the security algorithm and the secret information to achieve a particular security service. Parts One and Two of this book concentrate on the types of security mechanisms and services that fit into the model shown in Figure 1.5. However, there are other security-related situations of interest that do not neatly fit this model but are considered in this book. A general model of these other situations is illustrated by Figure 1.6, which reflects a concern for protecting an information system from unwanted access. Most readers are familiar with the concerns caused by the existence of hackers who attempt to penetrate systems that can be accessed over a network. The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system. The intruder can be a disgruntled employee who wishes to do damage or a criminal who seeks to exploit computer assets for financial gain (e.g., obtaining credit card numbers or performing illegal money transfers). Another type of unwanted access is the placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application programs as well as utility programs, such as editors and compilers. Programs can present two kinds of threats: 1. Information access threats: Intercept or modify data on behalf of users who should not have access to that data. 2. Service threats: Exploit service flaws in computers to inhibit use by legitimate users. Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a disk that contains the unwanted logic concealed in otherwise useful software. They also can be inserted into a system across a network; this latter mechanism is of more concern in network security. The security mechanisms needed to cope with unwanted access fall into two broad categories (see Figure 1.6). The first category might be termed a gatekeeper function. It includes password-based login procedures that are designed to deny access to all but authorized users and screening logic that is designed to detect and reject worms, viruses, and other similar attacks. Once either an unwanted user Figure 1.6 Network Access Security Model Computing resources (processor, memory, I/O) Data Processes Software Internal security controls Information system Gatekeeper function Opponent —human (e.g., hacker) —software (e.g., virus, worm) Access channel 42 chapter 1 / Introduction or unwanted software gains access, the second line of defense consists of a variety of internal controls that monitor activity and analyze stored information in an attempt to detect the presence of unwanted intruders. These issues are explored in Part Three. 1.9 Standards Many of the security techniques and applications described in this book have been specified as standards. Additionally, standards have been developed to cover management practices and the overall architecture of security mechanisms and services. Throughout this book, we describe the most important standards in use or being developed for various aspects of cryptography and network security. Various organizations have been involved in the development or promotion of these standards. The most important (in the current context) of these organizations are as follows. ■ National Institute of Standards and Technology: NIST is a

U.S. federal agency that deals with measurement science, standards, and technology related to U.S. government use and to the promotion of U.S. private-sector innovation. Despite its national scope, NIST Federal Information Processing Standards (FIPS) and Special Publications (SP) have a worldwide impact. ■ Internet Society: ISOC is a professional membership society with worldwide organizational and individual membership. It provides leadership in addressing issues that confront the future of the Internet and is the organization home for the groups responsible for Internet infrastructure standards, including the Internet Engineering Task Force (IETF) and the Internet Architecture Board (IAB). These organizations develop Internet standards and related specifications, all of which are published as Requests for Comments (RFCs). A more detailed discussion of these organizations is contained in Appendix C. 1.10 Key Terms, Review Questions, and Problems Key Terms access control active attack authentication authenticity availability data confidentiality data integrity denial of service encryption integrity intruder masquerade nonrepudiation OSI security architecture passive attack replay security attacks security mechanisms security services traffic analysis 1.10 / Key Terms, Review Questions, and Problems 43 Review Questions 1.1 What is the OSI security architecture? 1.2 Briefly explain masquerade attack with an example. 1.3 What is the difference between security threats and attacks? 1.4 Why are passive attacks difficult to detect and active attacks difficult to prevent? 1.5 Identify the different security attacks prevented by the security mechanisms defined in X.800. 1.6 List and briefly define the fundamental security design principles. 1.7 Explain the difference between an attack surface and an attack tree. Problems 1.1 Consider an automated teller machine (ATM) in which users provide a personal identification number (PIN) and a card for account access. Give examples of confidentiality, integrity, and availability requirements associated with the system. In each case, indicate the degree of importance of the requirement. 1.2 Repeat Problem 1.1 for a telephone switching system that routes calls through a switching network based on the telephone number requested by the caller. 1.3 Consider a desktop publishing system used to produce documents for various organizations. a. Give an example of a type of publication for which confidentiality of the stored data is the most important requirement. b. Give an example of a type of publication in which data integrity is the most important requirement. c. Give an example in which system availability is the most important requirement. 1.4 For each of the following assets, assign a low, moderate, or high impact level for the loss of confidentiality, availability, and integrity, respectively. Justify your answers. a. A portal maintained by the Government to provide information regarding its departments and services. b. A hospital managing the medical records of its patients. c. A financial organization managing routine administrative information (not privacy-related information). d. An information system used for large acquisitions in a contracting organization that contains both sensitive, pre-solicitation phase contract information and routine administrative information. Assess the impact for the two data sets separately and the information system as a whole. e. The Examinations department of a University maintains examination particulars, such as question papers of forthcoming examinations, grades obtained, and examiner details. The University's administrative department maintains the students' attendance particulars and internal assessment results. Assess the impact for the two data sets separately and the information system as a whole. 1.5 Draw a matrix similar to Table 1.4 that shows the relationship between security attacks and mechanisms. 1.6 Draw a matrix similar to Table 1.4 that shows the relationship between security mechanisms and services. 1.7 Develop an attack tree for gaining access to customer account details from the database of a bank. 1.8 Consider a company whose operations are housed in two buildings on the same property; one building is headquarters, the other building contains network and computer services. The property is physically protected by a fence around the perimeter. The

only entrance to the property is through the fenced perimeter. In addition to the 44 chapter 1 / Introduction perimeter fence, physical security consists of a guarded front gate. The local networks are split between the Headquarters' LAN and the Network Services' LAN. Internet users connect to the Web server through a firewall. Dial-up users get access to a particular server on the Network Services' LAN. Develop an attack tree in which the root node represents disclosure of proprietary secrets. Include physical, social engineering, and technical attacks. The tree may contain both AND and OR nodes. Develop a tree that has at least 15 leaf nodes. 1.9 Read all of the classic papers cited in the Recommended Reading section for this chapter, available at the Author Web site at WilliamStallings.com/NetworkSecurity. The papers are available at box.com/NetSec6e. Compose a 500–1000 word paper (or 8 to 12 slide PowerPoint presentation) that summarizes the key concepts that emerge from these papers, emphasizing concepts that are common to most or all of the papers. 45 Part One: Cryptography Chapter Symmetric Encryption and Message Confidentiality 2.1 Symmetric Encryption Principles Cryptography Cryptanalysis Feistel Cipher Structure 2.2 Symmetric Block Encryption Algorithms Data Encryption Standard Triple DES Advanced Encryption Standard 2.3 Random and Pseudorandom Numbers The Use of Random Numbers TRNGs, PRNGs, and PRFs Algorithm Design 2.4 Stream Ciphers and RC4 Stream Cipher Structure The RC4 Algorithm 2.5 Cipher Block Modes of Operation Electronic Codebook Mode Cipher Block Chaining Mode Cipher Feedback Mode Counter Mode 2.6 Key Terms, Review Questions, and Problems 46 chapter 2 / Symmetric Encryption and Message Confidentiality Symmetric encryption, also referred to as conventional encryption, secret-key, or single-key encryption, was the only type of encryption in use prior to the development of public-key encryption in the late 1970s.1 It remains by far the most widely used of the two types of encryption. This chapter begins with a look at a general model for the symmetric encryption process; this will enable us to understand the context within which the algorithms are used. Then we look at three important block encryption algorithms: DES, triple DES, and AES. This is followed by a discussion of random and pseudorandom number generation. Next, the chapter introduces symmetric stream encryption and describes the widely used stream cipher RC4. Finally, we look at the important topic of block cipher modes of operation. 2.1 Symmetric Encryption Principles A symmetric encryption scheme has five ingredients (Figure 2.1): ■■ Plaintext: This is the original message or data that is fed into the algorithm as input. ■■ Encryption algorithm: The encryption algorithm performs various substitutions and transformations on the plaintext. ■■ Secret key: The secret key is also input to the algorithm. The exact substitutions and transformations performed by the algorithm depend on the key. 1 Public-key encryption was first described in the open literature in 1976; the National Security Agency (NSA) claims to have discovered it some years earlier. Learning Objectives After studying this chapter, you should be able to: ◆ Present an overview of the main concepts of symmetric cryptography. ◆ Explain the difference between cryptanalysis and brute-force attack. ◆ Summarize the functionality of DES. ◆ Present an overview of AES. ◆ Explain the concepts of randomness and unpredictability with respect to random numbers. ◆ Understand the differences among true random number generators, pseudorandom number generators, and pseudorandom functions. ◆ Present an overview of stream ciphers and RC4. ◆ Compare and contrast ECB, CBC, CFB, and counter modes of operation. 2.1 / Symmetric Encryption Principles 47 Figure 2.1 Simplified Model of Symmetric Encryption Plaintext input Y = E[K, X] X = D[K, Y] X K K Transmitted ciphertext Plaintext output Secret key shared by sender and recipient Secret key shared by sender and recipient Encryption algorithm (e.g., AES) Decryption algorithm (reverse of encryption algorithm) ■■ Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. ■■ Decryption algorithm:

This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the same secret key and produces the original plaintext. There are two requirements for secure use of symmetric encryption: 1. We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext. 2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable. It is important to note that the security of symmetric encryption depends on the secrecy of the key, not the secrecy of the algorithm. That is, it is assumed that it is impractical to decrypt a message on the basis of the ciphertext plus knowledge of the encryption/decryption algorithm. In other words, we do not need to keep the algorithm secret; we need to keep only the key secret. This feature of symmetric encryption is what makes it feasible for widespread use. The fact that the algorithm need not be kept secret means that manufacturers can and have developed low-cost chip implementations of data encryption algorithms. These chips are widely available and incorporated into a number of products. With the use of symmetric encryption, the principal security problem is maintaining the secrecy of the key. 48 chapter 2 / Symmetric Encryption and Message Confidentiality Cryptography Cryptographic systems are generically classified along three independent dimensions: 1. The type of operations used for transforming plaintext to ciphertext. All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element; and transposition, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (i.e., that all operations be reversible). Most systems, referred to as product systems, involve multiple stages of substitutions and transpositions. 2. The number of keys used. If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver each use a different key, the system is referred to as asymmetric, two-key, or public-key encryption. 3. The way in which the plaintext is processed. A block cipher processes the input one block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along. Cryptanalysis The process of attempting to discover the plaintext or key is known as cryptanalysis. The strategy used by the cryptanalyst depends on the nature of the encryption scheme and the information available to the cryptanalyst. Table 2.1 summarizes the various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst. The most difficult problem is presented when all that is available is the ciphertext only. In some cases, not even the encryption algorithm is known, but in general, we can assume that the opponent does know the algorithm used for encryption. One possible attack under these circumstances is the brute-force approach of trying all possible keys. If the key space is very large, this becomes impractical. Thus, the opponent must rely on an analysis of the ciphertext itself, generally applying various statistical tests to it. To use this approach, the opponent must have some general idea of the type of plaintext that is concealed, such as English or French text, an EXE file, a Java source listing, an accounting file, and so on. The ciphertext-only attack is the easiest to defend against because the opponent has the least amount of information to work with. In many cases, however, the analyst has more information. The analyst may be able to capture one or more plaintext messages as well as their encryptions. Or the analyst may know that certain plaintext patterns

will appear in a message. For example, a file that is encoded in the Postscript format always begins with the same pattern, or there may be a standardized header or banner to an electronic funds transfer message, and so on. All of these are examples of known plaintext. With this knowledge, the analyst may be able to deduce the key on the basis of the way in which the known plaintext is transformed. 2.1 / Symmetric Encryption Principles 49 Type of Attack Known to Cryptanalyst Ciphertext only ■ Encryption algorithm ■ Ciphertext to be decoded Known plaintext ■ Encryption algorithm ■ Ciphertext to be decoded ■ One or more plaintext–ciphertext pairs formed with the secret key Chosen plaintext ■ Encryption algorithm ■ Ciphertext to be decoded ■ Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key Chosen ciphertext ■ Encryption algorithm ■ Ciphertext to be decoded ■ Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key Chosen text ■ Encryption algorithm ■ Ciphertext to be decoded ■ Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key ■ Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key Table 2.1 Types of Attacks on Encrypted Messages Closely related to the known-plaintext attack is what might be referred to as a probable-word attack. If the opponent is working with the encryption of some general prose message, he or she may have little knowledge of what is in the message. However, if the opponent is after some very specific information, then parts of the message may be known. For example, if an entire accounting file is being transmitted, the opponent may know the placement of certain key words in the header of the file. As another example, the source code for a program developed by a corporation might include a copyright statement in some standardized position. If the analyst is able somehow to get the source system to insert into the system a message chosen by the analyst, then a chosen-plaintext attack is possible. In general, if the analyst is able to choose the messages to encrypt, the analyst may deliberately pick patterns that can be expected to reveal the structure of the key. Table 2.1 lists two other types of attack: chosen ciphertext and chosen text. These are less commonly employed as cryptanalytic techniques but are nevertheless possible avenues of attack. Only relatively weak algorithms fail to withstand a ciphertext-only attack. Generally, an encryption algorithm is designed to withstand a known-plaintext attack. An encryption scheme is computationally secure if the ciphertext generated by the scheme meets one or both of the following criteria: ■■ The cost of breaking the cipher exceeds the value of the encrypted information. ■■ The time required to break the cipher exceeds the useful lifetime of the information. 50 chapter 2 / Symmetric Encryption and Message Confidentiality Unfortunately, it is very difficult to estimate the amount of effort required to cryptanalyze ciphertext successfully. However, assuming there are no inherent mathematical weaknesses in the algorithm, then a brute-force approach is indicated. A brute-force attack involves trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained. On average, half of all possible keys must be tried to achieve success. That is, if there are x different keys, on average an attacker would discover the actual key after x/2 tries. It is important to note that there is more to a brute-force attack than simply running through all possible keys. Unless known plaintext is provided, the analyst must be able to recognize plaintext as plaintext. If the message is just plaintext in English, then the result pops out easily, although the task of recognizing English would have to be automated. If the text message has been compressed before encryption, then recognition is more difficult. And if the message is some more general type of data, such as a numerical file, and this has been compressed, the problem becomes even more difficult to automate. Thus, to supplement the brute-force approach, some degree of knowledge about the expected plaintext is needed, and some means

of automatically distinguishing plaintext from garble is also needed. Feistel Cipher Structure Many symmetric block encryption algorithms, including DES, have a structure first described by Horst Feistel of IBM in 1973 [FEIS73] and shown in Figure 2.2. The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K. The plaintext block is divided into two halves, LE0 and RE0. The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs LEi-1 and REi-1 derived from the previous round, as well as a subkey Ki derived from the overall K. In general, the subkeys Ki are different from K and from each other and are generated from the key by a subkey generation algorithm. In Figure 2.2, 16 rounds are used, although any number of rounds could be implemented. The right-hand side of Figure 2.2 shows the decryption process. All rounds have the same structure. A substitution is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking the exclusive-OR (XOR) of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey Ki . Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data. The Feistel structure is a particular example of the more general structure used by all symmetric block ciphers. In general, a symmetric block cipher consists of a sequence of rounds, with each round performing substitutions and permutations conditioned by a secret key value. The exact realization of a symmetric block cipher depends on the choice of the following parameters and design features. ■■ Block size: Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed. A block size of 128 bits is a reasonable trade-off and is nearly universal among recent block cipher designs. 2.1 / Symmetric Encryption Principles 51 Figure 2.2 Feistel Encryption and Decryption (16 rounds) Output (ciphertext) K1 LD0 = RE16 RD0 = LE16 LD2 = RE14 RD2 = LE14 LD14 = RE2 RD14 = LE2 LD16 = RE0 LD17 = RE0 RD16 = LE0 RD17 = LE0 RD1 LD1 = LE15 = RE15 RD15 LD15 = LE1 = RE1 Input (ciphertext) Output (plaintext) Round 1 K1 K2 K15 K16 K2 K15 K16 F LE0 RE0 Input (plaintext) LE1 RE1 LE2 RE2 F F LE14 RE14 LE15 RE15 LE16 RE16 LE17 RE17 F F F F F Round 2 Round 15 Round 16 Round 16 Round 15 Round 2 Round 1 ■■ Key size: Larger key size means greater security but may decrease encryption/ decryption speed. The most common key length in modern algorithms is 128 bits. ■■ Number of rounds: The essence of a symmetric block cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is from 10 to 16 rounds. 52 chapter 2 / Symmetric Encryption and Message Confidentiality ■■ Subkey generation algorithm: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis. ■■ Round function: Again, greater complexity generally means greater resistance to cryptanalysis. There are two other considerations in the design of a symmetric block cipher: ■■ Fast software encryption/decryption: In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern. ■■ Ease of analysis: Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength. DES, for example, does not have an easily analyzed functionality. Decryption with a symmetric block cipher is essentially the same as the encryption process. The rule is as follows: Use the ciphertext as input to the algorithm, but use the subkeys Ki in reverse order. That is, use Kn in the first round, Kn-1 in the second round, and so on until K1 is used in the last round. This is a nice feature, because it means we need not implement two different algorithms—one for encryption and one for decryption. 2.2 Symmetric Block Encryption Algorithms The most

commonly used symmetric encryption algorithms are block ciphers. A block cipher processes the plaintext input in fixed-sized blocks and produces a block of ciphertext of equal size for each plaintext block. This section focuses on the three most important symmetric block ciphers: the Data Encryption Standard (DES), triple DES (3DES), and the Advanced Encryption Standard (AES). Data Encryption Standard Until the introduction of the Advanced Encryption Standard in 2001, the most widely used encryption scheme was based on the Data Encryption Standard (DES) issued in 1977 as Federal Information Processing Standard 46 (FIPS 46) by the National Bureau of Standards, now known as the National Institute of Standards and Technology (NIST). The algorithm itself is referred to as the Data Encryption Algorithm (DEA).2 2 The terminology is a bit confusing. Until recently, the terms DES and DEA could be used interchangeably. However, the most recent edition of the DES document includes a specification of the DEA described here plus the triple DEA (3DES) described subsequently. Both DEA and 3DES are part of the Data Encryption Standard. Furthermore, until the recent adoption of the official term 3DES, the triple DEA algorithm was typically referred to as triple DES and written as 3DES. For the sake of convenience, we will use 3DES. 2.2 / Symmetric Block Encryption Algorithms 53 Description of the Algorithm The plaintext is 64 bits in length and the key is 56 bits in length; longer plaintext amounts are processed in 64-bit blocks. The DES structure is a minor variation of the Feistel network shown in Figure 2.2. There are 16 rounds of processing. From the original 56-bit key, 16 subkeys are generated, one of which is used for each round. The process of decryption with DES is essentially the same as the encryption process. The rule is as follows: Use the ciphertext as input to the DES algorithm, but use the subkeys Ki in reverse order. That is, use K16 on the first iteration, K15 on the second iteration, and so on until K1 is used on the 16th and last iteration. The Strength of DES Concerns about the strength of DES fall into two categories: concerns about the algorithm itself and concerns about the use of a 56-bit key. The first concern refers to the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm. Over the years, there have been numerous attempts to find and exploit weaknesses in the algorithm, making DES the moststudied encryption algorithm in existence. Despite numerous approaches, no one has so far succeeded in discovering a fatal weakness in DES.3 A more serious concern is key length. With a key length of 56 bits, there are 256 possible keys, which is approximately 7.2 * 1016 keys. Thus, on the face of it, a brute-force attack appears impractical. Assuming that on average half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher. However, the assumption of one encryption per microsecond is overly conservative. DES finally and definitively proved insecure in July 1998, when the Electronic Frontier Foundation (EFF) announced that it had broken a DES encryption using a special-purpose “DES cracker” machine that was built for less than $250,000. The attack took less than three days. The EFF has published a detailed description of the machine, enabling others to build their own cracker [EFF98]. And, of course, hardware prices will continue to drop as speeds increase, making DES virtually worthless. With current technology, it is not even necessary to use special, purpose-built hardware. Rather, the speed of commercial, off-the-shelf processors threaten the security of DES. A paper from Seagate Technology [SEAG08] suggests that a rate of one billion (109) key combinations per second is reasonable for today’s multicore computers. Recent offerings confirm this. Both Intel and AMD now offer hardwarebased instructions to accelerate the use of AES. Tests run on a contemporary multicore Intel machine resulted in an encryption rate of about half a billion [BASU12]. Another recent analysis suggests that with contemporary supercomputer technology, a rate of 1013 encryptions/s is reasonable [AROR12]. Considering these results, Table 2.2 shows how much time is required for a brute-force attack for various

key sizes. As can be seen, a single PC can break DES in about a year; if multiple PCs work in parallel, the time is drastically shortened. And today's supercomputers should be able to find a key in about an hour. Key sizes of 128 bits or greater are effectively unbreakable using simply a brute-force approach. Even if we managed to speed up the attacking system by a factor of 1 trillion (1012), it would still take over 100,000 years to break a code using a 128-bit key. 3 At least, no one has publicly acknowledged such a discovery. 54 chapter 2 / Symmetric Encryption and Message Confidentiality Figure 2.3 Triple DES E K1 D K2 E K3 P A B C (a) Encryption D K3 E K2 D K1 C B A P (b) Decryption Fortunately, there are a number of alternatives to DES, the most important of which are triple DES and AES, discussed in the remainder of this section. Triple DES Triple DES (3DES) was first standardized for use in financial applications in ANSI standard X9.17 in 1985. 3DES was incorporated as part of the Data Encryption Standard in 1999 with the publication of FIPS 46-3. 3DES uses three keys and three executions of the DES algorithm. The function follows an encrypt-decrypt-encrypt (EDE) sequence (Figure 2.3a): C = E(K3, D(K2, E(K1, P))) where C = ciphertext P = plaintext E[K, X] = encryption of X using key K D[K, Y] = decryption of Y using key K Table 2.2 Average Time Required for Exhaustive Key Search Key Size (bits) Cipher Number of Alternative Keys Time Required at 109 Decryptions/s Time Required at 1013 Decryptions/s 56 DES 256 ≈ 7.2 * 1016 255 ns = 1.125 years 1 hour 128 AES 2128 ≈ 3.4 * 1038 2127 ns = 5.3 * 1021 years 5.3 * 1017 years 168 Triple DES 2168 ≈ 3.7 * 1050 2167 ns = 5.8 * 1033 years 5.8 * 1029 years 192 AES 2192 ≈ 6.3 * 1057 2191 ns = 9.8 * 1040 years 9.8 * 1036 years 256 AES 2256 ≈ 1.2 * 1077 2255 ns = 1.8 * 1060 years 1.8 * 1056 years 2.2 / Symmetric Block Encryption Algorithms 55 Decryption is simply the same operation with the keys reversed (Figure 2.3b): P = D(K1, E(K2, D(K3, C))) There is no cryptographic significance to the use of decryption for the second stage of 3DES encryption. Its only advantage is that it allows users of 3DES to decrypt data encrypted by users of the older single DES: C = E(K1, D(K1, E(K1, P))) = E[K, P] With three distinct keys, 3DES has an effective key length of 168 bits. FIPS 46-3 also allows for the use of two keys, with K1 = K3; this provides for a key length of 112 bits. FIPS 46-3 includes the following guidelines for 3DES. ■■ 3DES is the FIPS-approved symmetric encryption algorithm of choice. ■■ The original DES, which uses a single 56-bit key, is permitted under the standard for legacy systems only. New procurements should support 3DES. ■■ Government organizations with legacy DES systems are encouraged to transition to 3DES. ■■ It is anticipated that 3DES and the Advanced Encryption Standard (AES) will coexist as FIPS-approved algorithms, allowing for a gradual transition to AES. It is easy to see that 3DES is a formidable algorithm. Because the underlying cryptographic algorithm is DEA, 3DES can claim the same resistance to cryptanalysis based on the algorithm as is claimed for DEA. Furthermore, with a 168-bit key length, brute-force attacks are effectively impossible. Ultimately, AES is intended to replace 3DES, but this process will take a number of years. NIST anticipates that 3DES will remain an approved algorithm (for U.S. government use) for the foreseeable future. Advanced Encryption Standard 3DES has two attractions that assure its widespread use over the next few years. First, with its 168-bit key length, it overcomes the vulnerability to brute-force attack of DEA. Second, the underlying encryption algorithm in 3DES is the same as in DEA. This algorithm has been subjected to more scrutiny than any other encryption algorithm over a longer period of time, and no effective cryptanalytic attack based on the algorithm rather than brute force has been found. Accordingly, there is a high level of confidence that 3DES is very resistant to cryptanalysis. If security were the only consideration, then 3DES would be an appropriate choice for a standardized encryption algorithm for decades to come. The principal drawback of 3DES is that the algorithm is relatively sluggish in software. The original DEA was designed for mid-1970s hardware implementation and does not produce efficient software code. 3DES, which has three times as many rounds as DEA, is

correspondingly slower. A secondary drawback is that both DEA and 3DES use a 64-bit block size. For reasons of both efficiency and security, a larger block size is desirable. Because of these drawbacks, 3DES is not a reasonable candidate for longterm use. As a replacement, NIST in 1997 issued a call for proposals for a new 56 chapter 2 / Symmetric Encryption and Message Confidentiality Advanced Encryption Standard (AES), which should have a security strength equal to or better than 3DES and significantly improved efficiency. In addition to these general requirements, NIST specified that AES must be a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192, and 256 bits. Evaluation criteria included security, computational efficiency, memory requirements, hardware and software suitability, and flexibility. In a first round of evaluation, 15 proposed algorithms were accepted. A second round narrowed the field to five algorithms. NIST completed its evaluation process and published a final standard (FIPS PUB 197) in November of 2001. NIST selected Rijndael as the proposed AES algorithm. The two researchers who developed and submitted Rijndael for the AES are both cryptographers from Belgium: Dr. Joan Daemen and Dr. Vincent Rijmen. Overview of the Algorithm AES uses a block length of 128 bits and a key length that can be 128, 192, or 256 bits. In the description of this section, we assume a key length of 128 bits, which is likely to be the one most commonly implemented. The input to the encryption and decryption algorithms is a single 128-bit block. In FIPS PUB 197, this block is depicted as a square matrix of bytes. This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix. Similarly, the 128-bit key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words: Each word is four bytes and the total key schedule is 44 words for the 128-bit key. The ordering of bytes within a matrix is by column. So, for example, the first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the in matrix, the second four bytes occupy the second column, and so on. Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the w matrix. The following comments give some insight into AES. 1. One noteworthy feature of this structure is that it is not a Feistel structure. Recall that in the classic Feistel structure, half of the data block is used to modify the other half of the data block, and then the halves are swapped. AES does not use a Feistel structure but processes the entire data block in parallel during each round using substitutions and permutation. 2. The key that is provided as input is expanded into an array of forty-four 32-bit words, w[i]. Four distinct words (128 bits) serve as a round key for each round. 3. Four different stages are used, one of permutation and three of substitution (Figure 2.4): ■■ Substitute bytes: Uses a table, referred to as an S-box,4 to perform a byteby-byte substitution of the block. ■■ Shift rows: A simple permutation that is performed row by row. 4 The term S-box, or substitution box, is commonly used in the description of symmetric ciphers to refer to a table used for a table-lookup type of substitution mechanism. 2.2 / Symmetric Block Encryption Algorithms 57 ■■ Mix columns: A substitution that alters each byte in a column as a function of all of the bytes in the column. ■■ Add round key: A simple bitwise XOR of the current block with a portion of the expanded key. 4. The structure is quite simple. For both encryption and decryption, the cipher begins with an Add Round Key stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages. Figure 2.5 depicts the structure of a full encryption round. Figure 2.4 AES Encryption and Decryption Add round key w[4, 7] Plaintext (16 bytes) Plaintext (16 bytes) Substitute bytes Expand key Shift rows Round 1 Mix columns Round 9 Round 10 Add round key • • • Substitute bytes Shift rows Mix columns Add round key Substitute bytes Shift rows Add round key Ciphertext (16 bytes) (a) Encryption Key (16 bytes) Add round key Inverse sub bytes Inverse shift rows Inverse mix cols Round 10 Round 9 Round 1 Add round key • • • Inverse sub bytes Inverse shift rows Inverse mix cols Add round key Inverse sub bytes

Inverse shift rows Add round key Ciphertext (16 bytes) (b) Decryption w[36, 39] w[40, 43] w[0, 3] 58 chapter 2 / Symmetric Encryption and Message Confidentiality S S S S S S S S S S S S S S S S SubBytes State State State State State ShiftRows MixColumns AddRoundKey M M M M r0 r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12 r13 r14 r15 Figure 2.5 AES Encryption Round 2.3 / Random and Pseudorandom Numbers 59 5. Only the Add Round Key stage makes use of the key. For this reason, the cipher begins and ends with an Add Round Key stage. Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security. 6. The Add Round Key stage by itself would not be formidable. The other three stages together scramble the bits, but by themselves, they would provide no security because they do not use the key. We can view the cipher as alternating operations of XOR encryption (Add Round Key) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on. This scheme is both efficient and highly secure. 7. Each stage is easily reversible. For the Substitute Byte, Shift Row, and Mix Columns stages, an inverse function is used in the decryption algorithm. For the Add Round Key stage, the inverse is achieved by XORing the same round key to the block, using the result that $A \oplus B \oplus B = A$. 8. As with most block ciphers, the decryption algorithm makes use of the expanded key in reverse order. However, the decryption algorithm is not identical to the encryption algorithm. This is a consequence of the particular structure of AES. 9. Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext. Figure 2.4 lays out encryption and decryption going in opposite vertical directions. At each horizontal point (e.g., the dashed line in the figure), State is the same for both encryption and decryption. 10. The final round of both encryption and decryption consists of only three stages. Again, this is a consequence of the particular structure of AES and is required to make the cipher reversible. 2.3 Random and Pseudorandom Numbers Random numbers play an important role in the use of encryption for various network security applications. We provide an overview in this section. The topic is examined in more detail in Appendix E. The Use of Random Numbers A number of network security algorithms based on cryptography make use of random numbers. For example, ■■ Generation of keys for the RSA public-key encryption algorithm (described in Chapter 3) and other public-key algorithms. ■■ Generation of a stream key for symmetric stream cipher (discussed in the following section). 60 chapter 2 / Symmetric Encryption and Message Confidentiality ■■ Generation of a symmetric key for use as a temporary session key. This function is used in a number of networking applications, such as Transport Layer Security (Chapter 5), Wi-Fi (Chapter 6), e-mail security (Chapter 7), and IP security (Chapter 8). ■■ In a number of key distribution scenarios, such as Kerberos (Chapter 4), random numbers are used for handshaking to prevent replay attacks. These applications give rise to two distinct and not necessarily compatible requirements for a sequence of random numbers: randomness and unpredictability. Randomness Traditionally, the concern in the generation of a sequence of allegedly random numbers has been that the sequence of numbers be random in some welldefined statistical sense. The following criteria are used to validate that a sequence of numbers is random. ■■ Uniform distribution: The distribution of bits in the sequence should be uniform; that is, the frequency of occurrence of ones and zeros should be approximately the same. ■■ Independence: No one subsequence in the sequence can be inferred from the others. Although there are well-defined tests for determining that a sequence of numbers matches a particular distribution, such as the uniform distribution, there is no such test to "prove" independence. Rather, a number of tests can be applied to demonstrate if a sequence does not exhibit independence. The general strategy is to apply a number of such tests until the confidence that independence exists is sufficiently strong. In the context of our discussion, the use of a sequence of numbers that appear statistically random often occurs in the design of

algorithms related to cryptography. For example, a fundamental requirement of the RSA public-key encryption scheme discussed in Chapter 3 is the ability to generate prime numbers. In general, it is difficult to determine if a given large number N is prime. A brute-force approach would be to divide N by every odd integer less than 2N. If N is on the order, say, of 10150 (a not uncommon occurrence in public-key cryptography), such a brute-force approach is beyond the reach of human analysts and their computers. However, a number of effective algorithms exist that test the primality of a number by using a sequence of randomly chosen integers as input to relatively simple computations. If the sequence is sufficiently long (but far, far less than 210150), the primality of a number can be determined with near certainty. This type of approach, known as randomization, crops up frequently in the design of algorithms. In essence, if a problem is too hard or time-consuming to solve exactly, a simpler, shorter approach based on randomization is used to provide an answer with any desired level of confidence. Unpredictability In applications such as reciprocal authentication and session key generation, the requirement is not so much that the sequence of numbers be statistically random but that the successive members of the sequence are unpredictable. With “true” random sequences, each number is statistically independent of other 2.3 / Random and Pseudorandom Numbers 61 numbers in the sequence and therefore unpredictable. However, as is discussed shortly, true random numbers are not always used; rather, sequences of numbers that appear to be random are generated by some algorithm. In this latter case, care must be taken that an opponent not be able to predict future elements of the sequence on the basis of earlier elements. TRNGs, PRNGs, and PRFs Cryptographic applications typically make use of algorithmic techniques for random number generation. These algorithms are deterministic and therefore produce sequences of numbers that are not statistically random. However, if the algorithm is good, the resulting sequences will pass many reasonable tests of randomness. Such numbers are referred to as pseudorandom numbers. You may be somewhat uneasy about the concept of using numbers generated by a deterministic algorithm as if they were random numbers. Despite what might be called philosophical objections to such a practice, it generally works. That is, under most circumstances, pseudorandom numbers will perform as well as if they were random for a given use. The phrase “as well as” is unfortunately subjective, but the use of pseudorandom numbers is widely accepted. The same principle applies in statistical application, in which a statistician takes a sample of a population and assumes that the results will be approximately the same as if the whole population were measured. Figure 2.6 contrasts a true random number generator (TRNG) with two forms of pseudorandom number generators. A TRNG takes as input a source that is effectively random; the source is often referred to as an entropy source. In essence, the entropy source is drawn from the physical environment of the computer Figure 2.6 Random and Pseudorandom Number Generators Conversion to binary Source of true randomness Random bit stream (a) TRNG TRNG = true random number generator PRNG = pseudorandom number generator PRF = pseudorandom function Deterministic algorithm Seed Pseudorandom bit stream (b) PRNG Deterministic algorithm Seed Pseudorandom value (c) PRF Contextspecic values 62 chapter 2 / Symmetric Encryption and Message Confidentiality and could include things such as keystroke timing patterns, disk electrical activity, mouse movements, and instantaneous values of the system clock. The source, or combination of sources, serves as input to an algorithm that produces random binary output. The TRNG may simply involve conversion of an analog source to a binary output. The TRNG may involve additional processing to overcome any bias in the source. In contrast, a PRNG takes as input a fixed value, called the seed, and produces a sequence of output bits using a deterministic algorithm. Typically, as shown in Figure 2.6, there is some feedback path by which some of the results of the algorithm are fed back as input as additional output bits are

produced. The important thing to note is that the output bit stream is determined solely by the input value or values, so that an adversary who knows the algorithm and the seed can reproduce the entire bit stream. Figure 2.6 shows two different forms of PRNGs, based on application. ■■ Pseudorandom number generator: An algorithm that is used to produce an open-ended sequence of bits is referred to as a PRNG. A common application for an open-ended sequence of bits is as input to a symmetric stream cipher, as discussed in the following section. ■■ Pseudorandom function (PRF): A PRF is used to produce a pseudorandom string of bits of some fixed length. Examples are symmetric encryption keys and nonces. Typically, the PRF takes as input a seed plus some context specific values, such as a user ID or an application ID. A number of examples of PRFs will be seen throughout this book. Other than the number of bits produced, there is no difference between a PRNG and a PRF. The same algorithms can be used in both applications. Both require a seed and both must exhibit randomness and unpredictability. Furthermore, a PRNG application may also employ context-specific input. Algorithm Design Cryptographic PRNGs have been the subject of much research over the years, and a wide variety of algorithms have been developed. These fall roughly into two categories: ■■ Purpose-built algorithms: These are algorithms designed specifically and solely for the purpose of generating pseudorandom bit streams. Some of these algorithms are used for a variety of PRNG applications; several of these are described in the next section. Others are designed specifically for use in a stream cipher. The most important example of the latter is RC4, described in the next section. ■■ Algorithms based on existing cryptographic algorithms: Cryptographic algorithms have the effect of randomizing input. Indeed, this is a requirement of such algorithms. For example, if a symmetric block cipher produced ciphertext that had certain regular patterns in it, it would aid in the process of cryptanalysis. Thus, cryptographic algorithms can serve as the core of PRNGs. Three 2.4 / Stream Ciphers and RC4 63 broad categories of cryptographic algorithms are commonly used to create PRNGs: —Symmetric block ciphers —Asymmetric ciphers —Hash functions and message authentication codes Any of these approaches can yield a cryptographically strong PRNG. A purpose-built algorithm may be provided by an operating system for general use. For applications that already use certain cryptographic algorithms for encryption or authentication, it makes sense to re-use the same code for the PRNG. Thus, all of these approaches are in common use. 2.4 Stream Ciphers and RC4 A block cipher processes the input one block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time as it goes along. Although block ciphers are far more common, there are certain applications in which a stream cipher is more appropriate. Examples are given subsequently in this book. In this section, we look at perhaps the most popular symmetric stream cipher, RC4. We begin with an overview of stream cipher structure, and then examine RC4. Stream Cipher Structure A typical stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time. Figure 2.7 is a representative diagram of stream cipher structure. In this structure, a key is input to a pseudorandom bit generator that produces a Figure 2.7 Stream Cipher Diagram Pseudorandom byte generator (key stream generator) Plaintext byte stream M Key K Key K k Plaintext byte stream M Ciphertext byte stream C ENCRYPTION Pseudorandom byte generator (key stream generator) DECRYPTION k 64 chapter 2 / Symmetric Encryption and Message Confidentiality stream of 8-bit numbers that are apparently random. The pseudorandom stream is unpredictable without knowledge of the input key and has an apparently random character. The output of the generator, called a keystream, is combined one byte at a time with the plaintext stream using the bitwise exclusive-OR (XOR) operation. For example, if the next byte generated by the generator is 01101100 and the next

plaintext byte is 11001100, then the resulting ciphertext byte is 11001100 plaintext ⊕ 01101100 key stream 10100000 ciphertext Decryption requires the use of the same pseudorandom sequence: 10100000 ciphertext ⊕ 01101100 key stream 11001100 plaintext [KUMA97] lists the following important design considerations for a stream cipher. 1. The encryption sequence should have a large period. A pseudorandom number generator uses a function that produces a deterministic stream of bits that eventually repeats. The longer the period of repeat, the more difficult it will be to do cryptanalysis. 2. The keystream should approximate the properties of a true random number stream as close as possible. For example, there should be an approximately equal number of 1s and 0s. If the keystream is treated as a stream of bytes, then all of the 256 possible byte values should appear approximately equally often. The more random-appearing the keystream is, the more randomized the ciphertext is, making cryptanalysis more difficult. 3. Note from Figure 2.7 that the output of the pseudorandom number generator is conditioned on the value of the input key. To guard against brute-force attacks, the key needs to be sufficiently long. The same considerations as apply for block ciphers are valid here. Thus, with current technology, a key length of at least 128 bits is desirable. With a properly designed pseudorandom number generator, a stream cipher can be as secure as block cipher of comparable key length. A potential advantage of a stream cipher is that stream ciphers that do not use block ciphers as a building block are typically faster and use far less code than do block ciphers. The example in this chapter, RC4, can be implemented in just a few lines of code. In recent years, this advantage has diminished with the introduction of AES, which is quite efficient in software. Furthermore, hardware acceleration techniques are now available for AES. For example, the Intel AES Instruction Set has machine instructions for one round of encryption and decryption and key generation. Using the hardware instructions results in speedups of about an order of magnitude compared to pure software implementations [XU10]. 2.4 / Stream Ciphers and RC4 65 One advantage of a block cipher is that you can reuse keys. In contrast, if two plaintexts are encrypted with the same key using a stream cipher, then cryptanalysis is often quite simple [DAWS96]. If the two ciphertext streams are XORed together, the result is the XOR of the original plaintexts. If the plaintexts are text strings, credit card numbers, or other byte streams with known properties, then cryptanalysis may be successful. For applications that require encryption/decryption of a stream of data (such as over a data-communications channel or a browser/Web link), a stream cipher might be the better alternative. For applications that deal with blocks of data (such as file transfer, e-mail, and database), block ciphers may be more appropriate. However, either type of cipher can be used in virtually any application. The RC4 Algorithm RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security. It is a variable key-size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation. Analysis shows that the period of the cipher is overwhelmingly likely to be greater than 10100 [ROBS95a]. Eight to sixteen machine operations are required per output byte, and the cipher can be expected to run very quickly in software. RC4 is used in the Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers. It is also used in the Wired Equivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA) protocol that are part of the IEEE 802.11 wireless LAN standard. RC4 was kept as a trade secret by RSA Security. In September 1994, the RC4 algorithm was anonymously posted on the Internet on the Cypherpunks anonymous remailers list. The RC4 algorithm is remarkably simple and quite easy to explain. A variable-length key of from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S, with elements S[0], S[1], . . . , S[255]. At all times, S contains a permutation of all 8-bit numbers from 0 through 255. For encryption and decryption, a byte k (see Figure 2.7) is generated from S by selecting one of the

255 entries in a systematic fashion. As each value of k is generated, the entries in S are once again permuted. Initialization of S To begin, the entries of S are set equal to the values from 0 through 255 in ascending order; that is, S[0] = 0, S[1] = 1, c , S[255] = 255. A temporary vector, T, is also created. If the length of the key K is 256 bytes, then K is transferred to T. Otherwise, for a key of length keylen bytes, the first keylen elements of T are copied from K, and then K is repeated as many times as necessary to fill out T. These preliminary operations can be summarized as: /* Initialization */ for i = 0 to 255 do S[i] = i; T[i] = K[i mod keylen]; 66 chapter 2 / Symmetric Encryption and Message Confidentiality Next we use T to produce the initial permutation of S. This involves starting with S[0] and going through to S[255] and, for each S[i], swapping S[i] with another byte in S according to a scheme dictated by T[i]: /* Initial Permutation of S */ j = 0; for i = 0 to 255 do j = (j + S[i] + T[i]) mod 256; Swap (S[i], S[j]); Because the only operation on S is a swap, the only effect is a permutation. S still contains all the numbers from 0 through 255. Stream Generation Once the S vector is initialized, the input key is no longer used. Stream generation involves cycling through all the elements of S[i] and, for each S[i], swapping S[i] with another byte in S according to a scheme dictated by the current configuration of S. After S[255] is reached, the process continues, starting over again at S[0]: /* Stream Generation */ i, j = 0; while (true) i = (i + 1) mod 256; j = (j + S[i]) mod 256; Swap (S[i], S[j]); t = (S[i] + S[j]) mod 256; k = S[t]; To encrypt, XOR the value k with the next byte of plaintext. To decrypt, XOR the value k with the next byte of ciphertext. Figure 2.8 illustrates the RC4 logic. Strength of RC4 A number of papers have been published analyzing methods of attacking RC4 (e.g., [KNUD98], [FLUH00], [MANT01]). None of these approaches is practical against RC4 with a reasonable key length, such as 128 bits. A more serious problem is reported in [FLUH01]. The authors demonstrate that the WEP protocol, intended to provide confidentiality on 802.11 wireless LAN networks, is vulnerable to a particular attack approach. In essence, the problem is not with RC4 itself but the way in which keys are generated for use as input to RC4. This particular problem does not appear to be relevant to other applications using RC4 and can be remedied in WEP by changing the way in which keys are generated. This problem points out the difficulty in designing a secure system that involves both cryptographic functions and protocols that make use of them. 2.4 / Stream Ciphers and RC4 67 255254253 4 3210 S T S (a) Initial state of S and T (b) Initial permutation of S Swap K T T[i] j = j + S[i] + T[i] t = S[i] + S[j] S[i] S[j] keylen i S (c) Stream generation Swap j = j + S[i] S[i] S[j] S[t] k i Figure 2.8 RC4 68 chapter 2 / Symmetric Encryption and Message Confidentiality 2.5 Cipher Block Modes of Operation A symmetric block cipher processes one block of data at a time. In the case of DES and 3DES, the block length is b = 64 bits; for AES, the block length is b = 128 bits. For longer amounts of plaintext, it is necessary to break the plaintext into b-bit blocks (padding the last block if necessary). To apply a block cipher in a variety of applications, five modes of operation have been defined by NIST (Special Publication 800-38A). The five modes are intended to cover virtually all of the possible applications of encryption for which a block cipher could be used. These modes are intended for use with any symmetric block cipher, including triple DES and AES. The most important modes are described briefly in the remainder of this section. Electronic Codebook Mode The simplest way to proceed is using what is known as electronic codebook (ECB) mode, in which plaintext is handled b bits at a time and each block of plaintext is encrypted using the same key. The term codebook is used because, for a given key, there is a unique ciphertext for every b-bit block of plaintext. Therefore, one can imagine a gigantic codebook in which there is an entry for every possible b-bit plaintext pattern showing its corresponding ciphertext. With ECB, if the same b-bit block of plaintext appears more than once in the message, it always produces the same ciphertext. Because of this, for lengthy messages, the ECB mode may not be secure. If the message is highly structured, it may be possible for a cryptanalyst to exploit

these regularities. For example, if it is known that the message always starts out with certain predefined fields, then the cryptanalyst may have a number of known plaintext–ciphertext pairs to work with. If the message has repetitive elements with a period of repetition a multiple of b bits, then these elements can be identified by the analyst. This may help in the analysis or may provide an opportunity for substituting or rearranging blocks. To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks. Cipher Block Chaining Mode In the cipher block chaining (CBC) mode (Figure 2.9), the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block. In effect, we have chained together the processing of the sequence of plaintext blocks. The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block. Therefore, repeating patterns of b bits are not exposed. For decryption, each cipher block is passed through the decryption algorithm. The result is XORed with the preceding ciphertext block to produce the plaintext block. To see that this works, we can write Cj = E(K, [Cj-1 ⊕ Pj]) 2.5 / Cipher Block Modes of Operation 69 where E[K, X] is the encryption of plaintext X using key K, and ⊕ is the exclusiveOR operation. Then D(K, Cj) = D(K, E(K, [Cj-1 ⊕ Pj])) D(K, Cj) = Cj-1 ⊕ Pj Cj-1 ⊕ D(K, Cj) = Cj-1 ⊕ Cj-1 ⊕ Pj = Pj which verifies Figure 2.9b. To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext. On decryption, the IV is XORed with the output of the decryption algorithm to recover the first block of plaintext. The IV must be known to both the sender and receiver. For maximum security, the IV should be protected as well as the key. This could be done by sending the IV using ECB encryption. One reason for protecting the IV is as follows: If an opponent is able to fool the receiver into using a different value for IV, then the opponent is able to invert selected bits in the first block of plaintext. To see this, consider the following: C1 = E(K, [IV ⊕ P1]) P1 = IV ⊕ D(K, C1) Figure 2.9 Cipher Block Chaining (CBC) Mode C1 P1 Encrypt IV K P2 C2 Encrypt K PN CN CN–1 Encrypt K (a) Encryption P1 C1 Decrypt IV K C2 P2 Decrypt K CN PN CN–1 Decrypt K (b) Decryption 70 chapter 2 / Symmetric Encryption and Message Confidentiality Now use the notation that X[j] denotes the jth bit of the b-bit quantity X. Then P1[i] = IV[i] ⊕ D(K, C1)[i] Then, using the properties of XOR, we can state P1[i]' = IV[i]' ⊕ D(K, C1)[i] where the prime notation denotes bit complementation. This means that if an opponent can predictably change bits in IV, the corresponding bits of the received value of P1 can be changed. Cipher Feedback Mode It is possible to convert any block cipher into a stream cipher by using the cipher feedback (CFB) mode. A stream cipher eliminates the need to pad a message to be an integral number of blocks. It also can operate in real time. Thus, if a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher. One desirable property of a stream cipher is that the ciphertext be of the same length as the plaintext. Thus, if 8-bit characters are being transmitted, each character should be encrypted using 8 bits. If more than 8 bits are used, transmission capacity is wasted. Figure 2.10 depicts the CFB scheme. In the figure, it is assumed that the unit of transmission is s bits; a common value is s = 8. As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext. First, consider encryption. The input to the encryption function is a b-bit shift register that is initially set to some initialization vector (IV). The leftmost (most significant) s bits of the output of the encryption function are XORed with the first unit of plaintext P1 to produce the first unit of ciphertext C1, which is then transmitted. In addition, the contents of the shift register are shifted left by s bits, and C1 is placed in the rightmost (least significant) s bits of the shift register. This process continues until all plaintext units have been encrypted. For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of

the encryption function to produce the plaintext unit. Note that it is the encryption function that is used, not the decryption function. This is easily explained. Let Ss(X) be defined as the most significant s bits of X. Then C1 = P1 ⊕ Ss[E(K, IV)] Therefore, P1 = C1 ⊕ Ss[E(K, IV)] The same reasoning holds for subsequent steps in the process. Counter Mode Although interest in the counter mode (CTR) has increased recently, with applications to ATM (asynchronous transfer mode) network security and IPSec (IP security), this mode was proposed early on (e.g., [DIFF79]). 2.5 / Cipher Block Modes of Operation 71 Figure 2.11 depicts the CTR mode. A counter equal to the plaintext block size is used. The only requirement stated in NIST Special Publication 800-38A is that the counter value must be different for each plaintext block that is encrypted. Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block (modulo 2b , where b is the block size). For encryption, the counter is encrypted and then XORed with the plaintext block to produce the ciphertext block; there is no chaining. For decryption, the same sequence of counter values is used, with each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block. Figure 2.10 s-bit Cipher Feedback (CFB) Mode C1 IV P1 Encrypt Select s bits Discard b–s bits K (a) Encryption CN–1 (b) Decryption s bits s bits s bits C2 P2 Encrypt Select s bits Discard b–s bits K s bits b–s bits s bits Shift register s bits CN PN Encrypt Select s bits Discard b–s bits K s bits b–s bits s bits Shift register P1 IV C1 Encrypt Select s bits Discard b–s bits K CN–1 s bits C2 s bits CN s bits s bits s bits P2 Encrypt Select s bits Discard b–s bits K b–s bits s bits Shift register b–s bits s bits Shift register s bits PN Encrypt Select s bits Discard b–s bits K 72 chapter 2 / Symmetric Encryption and Message Confidentiality [LIPM00] lists the following advantages of CTR mode. ■■ Hardware efficiency: Unlike the chaining modes, encryption (or decryption) in CTR mode can be done in parallel on multiple blocks of plaintext or ciphertext. For the chaining modes, the algorithm must complete the computation on one block before beginning on the next block. This limits the maximum throughput of the algorithm to the reciprocal of the time for one execution of block encryption or decryption. In CTR mode, the throughput is only limited by the amount of parallelism that is achieved. ■■ Software efficiency: Similarly, because of the opportunities for parallel execution in CTR mode, processors that support parallel features (such as aggressive pipelining, multiple instruction dispatch per clock cycle, a large number of registers, and SIMD instructions) can be effectively utilized. Figure 2.11 Counter (CTR) Mode (a) Encryption P1 C1 Counter 1 Encrypt K Counter 2 Counter N P2 PN C2 Encrypt K CN Encrypt K (b) Decryption C1 P1 Counter 1 Encrypt K Counter 2 Counter N C2 CN P2 Encrypt K PN Encrypt K 2.6 / Key Terms, Review Questions, and Problems 73 ■■ Preprocessing: The execution of the underlying encryption algorithm does not depend on input of the plaintext or ciphertext. Therefore, if sufficient memory is available and security is maintained, preprocessing can be used to prepare the output of the encryption boxes that feed into the XOR functions in Figure 2.11. When the plaintext or ciphertext input is presented, then the only computation is a series of XORs. Such a strategy greatly enhances throughput. ■■ Random access: The ith block of plaintext or ciphertext can be processed in random-access fashion. With the chaining modes, block Ci cannot be computed until the i - 1 prior block are computed. There may be applications in which a ciphertext is stored, and it is desired to decrypt just one block; for such applications, the random access feature is attractive. ■■ Provable security: It can be shown that CTR is at least as secure as the other modes discussed in this section. ■■ Simplicity: Unlike ECB and CBC modes, CTR mode requires only the implementation of the encryption algorithm and not the decryption algorithm. This matters most when the decryption algorithm differs substantially from the encryption algorithm, as it does for AES. In addition, the decryption key scheduling need not be implemented. 2.6 Key Terms, Review Questions, and Problems Review Questions 2.1 What is symmetric encryption? What are the two requirements

for secure use of symmetric encryption? 2.2 What is cryptanalysis? Summarize the various types of cryptanalytic attacks on encrypted messages. 2.3 List the parameters of a symmetric block cipher for greater security. 2.4 What is a block cipher? Name the important symmetric block ciphers. 2.5 Describe the data encryption algorithm for 64-bit length plaintext and 56-bit length key. 2.6 Describe the encryption and decryption of triple DES. 2.7 What are the advantages and disadvantages of triple DES? 2.8 List the important design criteria for a stream cipher. Key Terms Advanced Encryption Standard (AES) block cipher brute-force attack cipher block chaining (CBC) mode cipher feedback (CFB) mode ciphertext counter mode (CTR) cryptanalysis cryptography Data Encryption Standard (DES) decryption electronic codebook (ECB) mode encryption end-to-end encryption Feistel cipher key distribution keystream link encryption plaintext session key stream cipher subkey symmetric encryption triple DES (3DES) 74 chapter 2 / Symmetric Encryption and Message Confidentiality Problems 2.1 This problem uses a real-world example of a symmetric cipher, from an old U.S. Special Forces manual (public domain). The document, filename SpecialForces.pdf, is available at box.com/NetSec6e. a. Using the two keys (memory words) cryptographic and network security, encrypt the following message: Be at the third pillar from the left outside the lyceum theatre tonight at seven. If you are distrustful bring two friends. Make reasonable assumptions about how to treat redundant letters and excess letters in the memory words and how to treat spaces and punctuation. Indicate what your assumptions are. Note: The message is from the Sherlock Holmes novel The Sign of Four. b. Decrypt the ciphertext. Show your work. c. Comment on when it would be appropriate to use this technique and what its advantages are. 2.2 Consider a very simple symmetric block encryption algorithm in which 64-bit blocks of plaintext are encrypted using a 128-bit key. Encryption is defined as C = (P ⊕ K1) Ä K0 where C = ciphertext, K = secret key, K0 = leftmost 64 bits of K, K1 = rightmost 64 bits of K, ⊕ = bitwise exclusive OR, and Ä is addition mod 264. a. Show the decryption equation. That is, show the equation for P as a function of C, K0, and K1. b. Suppose an adversary has access to two sets of plaintexts and their corresponding ciphertexts and wishes to determine K. We have the two equations: C = (P ⊕ K1) Ä K0; C' = (P' ⊕ K1) Ä K0 First, derive an equation in one unknown (e.g., K0). Is it possible to proceed further to solve for K0? 2.3 Perhaps the simplest “serious” symmetric block encryption algorithm is the Tiny Encryption Algorithm (TEA). TEA operates on 64-bit blocks of plaintext using a 128-bit key. The plaintext is divided into two 32-bit blocks (L0, R0), and the key is divided into four 32-bit blocks (K0, K1, K2, K3). Encryption involves repeated application of a pair of rounds, defined as follows for rounds i and i + 1: Li = Ri-1 Ri = Li-1 Ä F(Ri-1, K0, K1, di) Li+1 = Ri Ri+1 = Li Ä F(Ri , K2, K3, di+1) where F is defined as F(M, Kj , Kk, di) = ((M 6 6 4) Ä Kj) ⊕ ((M W 5) Ä Kk) ⊕ (M Ä di) and where the logical shift of x by y bits is denoted by x 6 6 y, the logical right shift of x by y bits is denoted by x W y, and di is a sequence of predetermined constants. a. Comment on the significance and benefit of using the sequence of constants. b. Illustrate the operation of TEA using a block diagram or flow chart type of depiction. 2.6 / Key Terms, Review Questions, and Problems 75 c. If only one pair of rounds is used, then the ciphertext consists of the 64-bit block (L2, R2). For this case, express the decryption algorithm in terms of equations. d. Repeat part (c) using an illustration similar to that used for part (b). 2.4 Is the DES decryption the inverse of DES encryption? Justify your answer. 2.5 Consider a Feistel cipher composed of 14 rounds with block length 128 bits and key length 128 bits. Suppose that, for a given k, the key scheduling algorithm determines values for the first seven round keys, k1, k2, c, k8, and then sets k8 = k7, k9 = k6, k10 = k5, c, k14 = k1 Suppose you have a ciphertext c. Explain how, with access to an encryption oracle, you can decrypt c and determine m using just a single oracle query. This shows that such a cipher is vulnerable to a chosen plaintext attack. (An encryption oracle can be thought of as a device that, when given a plaintext, returns the

corresponding ciphertext. The internal details of the device are not known to you, and you cannot break open the device. You can only gain information from the oracle by making queries to it and observing its responses.) 2.6 For any block cipher, the fact that it is a nonlinear function is crucial to its security. To see this, suppose that we have a linear block cipher EL that encrypts 256-bit blocks of plaintext into 256-bit blocks of ciphertext. Let EL(k, m) denote the encryption of a 256-bit message m under a key k (the actual bit length of k is irrelevant). Thus, EL(k, [m1 ⊕ m2]) = EL(k, m1) ⊕ EL(k, m2) for all 256@bit patterns m1, m2 Describe how, with 256 chosen ciphertexts, an adversary can decrypt any ciphertext without knowledge of the secret key k. (A "chosen ciphertext" means that an adversary has the ability to choose a ciphertext and then obtain its decryption. Here, you have 256 plaintext–ciphertext pairs to work with, and you have the ability to choose the value of the ciphertexts.) 2.7 Suppose you have a true random bit generator where each bit in the generated stream has the same probability of being a 0 or 1 as any other bit in the stream and that the bits are not correlated; that is, the bits are generated from identical independent distribution. However, the bit stream is biased. The probability of a 1 is 0.5 - d and the probability of a 0 is 0.5 + d where 0 6 d 6 0.5. A simple deskewing algorithm is as follows: Examine the bit stream as a sequence of nonoverlapping pairs. Discard all 00 and 11 pairs. Replace each 01 pair with 0 and each 10 pair with 1. a. What is the probability of occurrence of each pair in the original sequence? b. What is the probability of occurrence of 0 and 1 in the modified sequence? c. What is the expected number of input bits to produce x output bits? d. Suppose that the algorithm uses overlapping successive bit pairs instead of nonoverlapping successive bit pairs. That is, the first output bit is based on input bits 1 and 2, the second output bit is based on input bits 2 and 3, and so on. What can you say about the output bit stream? 2.8 Another approach to deskewing is to consider the bit stream as a sequence of nonoverlapping groups of n bits each and output the parity of each group. That is, if a group contains an odd number of ones, the output is 1; otherwise the output is 0. a. Express this operation in terms of a basic Boolean function. b. Assume, as in the Problem 2.7, that the probability of a 1 is 0.5 - d. If each group consists of 2 bits, what is the probability of an output of 1? c. If each group consists of 3 bits, what is the probability of an output of 1? d. Generalize the result to find the probability of an output of 1 for input groups of n bits. 76 chapter 2 / Symmetric Encryption and Message Confidentiality 2.9 Is it appropriate to reuse keys in RC4? Why or why not? 2.10 RC4 has a secret internal state which is a permutation of all the possible values of the vector S and the two indices i and j. a. Using a straightforward scheme to store the internal state, how many bits are used? b. Suppose we think of it from the point of view of how much information is represented by the state. In that case, we need to determine how many different states there are, then take the log to the base 2 to find out how many bits of information this represents. Using this approach, how many bits would be needed to represent the state? 2.11 Alice and Bob agree to communicate privately via e-mail using a scheme based on RC4, but they want to avoid using a new secret key for each transmission. Alice and Bob privately agree on a 128-bit key k. To encrypt a message m consisting of a string of bits, the following procedure is used. 1. Choose a random 80-bit value v 2. Generate the ciphertext c = RC4(v } k) ⊕ m 3. Send the bit string (v } c) a. Suppose Alice uses this procedure to send a message m to Bob. Describe how Bob can recover the message m from (v } c) using k. b. If an adversary observes several values (v1 } c1), (v2 } c2), c transmitted between Alice and Bob, how can he or she determine when the same key stream has been used to encrypt two messages? 2.12 With the ECB mode, if there is an error in a block of the transmitted ciphertext, only the corresponding plaintext block is affected. However, in the CBC mode, this error propagates. For example, an error in the transmitted C1 (Figure 2.9) obviously corrupts P1 and P2. a. Are any blocks beyond P2 affected? b. Suppose that there is a bit error in the source

version of P1. Through how many ciphertext blocks is this error propagated? What is the effect at the receiver? 2.13 Is it possible to perform decryption operations in parallel on multiple blocks of ciphertext in CBC mode? How about encryption? 2.14 Why should the IV in CBC be protected? 2.15 CBC-Pad is a block cipher mode of operation used in the RC5 block cipher, but it could be used in any block cipher. CBC-Pad handles plaintext of any length. The ciphertext is longer than the plaintext by at most the size of a single block. Padding is used to assure that the plaintext input is a multiple of the block length. It is assumed that the original plaintext is an integer number of bytes. This plaintext is padded at the end by from 1 to bb bytes, where bb equals the block size in bytes. The pad bytes are all the same and set to a byte that represents the number of bytes of padding. For example, if there are 8 bytes of padding, each byte has the bit pattern 00001000. Why not allow zero bytes of padding? That is, if the original plaintext is an integer multiple of the block size, why not refrain from padding? 2.16 Padding may not always be appropriate. For example, one might wish to store the encrypted data in the same memory buffer that originally contained the plaintext. In that case, the ciphertext must be the same length as the original plaintext. A mode for that purpose is the ciphertext stealing (CTS) mode. Figure 2.12a shows an implementation of this mode. a. Explain how it works. b. Describe how to decrypt Cn-1 and Cn. 2.6 / Key Terms, Review Questions, and Problems 77 Figure 2.12 Block Cipher Modes for Plaintext not a Multiple of Block Size IV P1 C1 K K K K + + + + PN–2 CN–2 CN–3 Encrypt Encrypt Encrypt Encrypt Encrypt Encrypt (a) Ciphertext stealing mode (b) Alternative method Encrypt CN X PN–1 CN–1 PN 00...0 IV P1 (bb bits) C1 (bb bits) K K K K + + + + PN–2 (bb bits) CN–2 (bb bits) CN–3 Select leftmost j bits PN–1 (bb bits) CN–1 (bb bits) PN (j bits) CN (j bits) Encrypt 2.17 Figure 2.12b shows an alternative to CTS for producing ciphertext of equal length to the plaintext when the plaintext is not an integer multiple of the block size. a. Explain the algorithm. b. Explain why CTS is preferable to this approach illustrated in Figure 2.12b. 2.18 If a bit error occurs in the transmission of a ciphertext character in 8-bit CFB mode, how far does the error propagate? 78 Public-Key Cryptography and Message Authentication Chapter 3.1 Approaches to Message Authentication Authentication Using Conventional Encryption Message Authentication without Message Encryption 3.2 Secure Hash Functions Hash Function Requirements Security of Hash Functions Simple Hash Functions The SHA Secure Hash Function SHA-3 3.3 Message Authentication Codes HMAC MACs Based on Block Ciphers 3.4 Public-Key Cryptography Principles Public-Key Encryption Structure Applications for Public-Key Cryptosystems Requirements for Public-Key Cryptography 3.5 Public-Key Cryptography Algorithms The RSA Public-Key Encryption Algorithm Diffie–Hellman Key Exchange Other Public-Key Cryptography Algorithms 3.6 Digital Signatures Digital Signature Generation and Verification RSA Digital Signature Algorithm 3.7 Key Terms, Review Questions, and Problems 3.1 / Approaches to Message Authentication 79 In addition to message confidentiality, message authentication is an important network security function. This chapter examines three aspects of message authentication. First, we look at the use of message authentication codes and hash functions to provide message authentication. Then we look at public-key encryption principles and two specific public-key algorithms. These algorithms are useful in the exchange of conventional encryption keys. Then we look at the use of public-key encryption to produce digital signatures, which provides an enhanced form of message authentication. 3.1 Approaches to Message Authentication Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message authentication. A message, file, document, or other collection of data is said to be authentic when it is genuine and comes from its alleged source. Message authentication is a procedure that allows communicating parties to verify that received messages are authentic.1 The two important

aspects are to verify that the contents of the message have not been altered and that the source is authentic. We may also wish to verify a message's timeliness (it has not been artificially delayed and replayed) and sequence relative to other messages flowing between two parties. All of these concerns come under the category of data integrity as described in Chapter 1. 1 For simplicity, for the remainder of this chapter, we refer to message authentication. By this we mean both authentication of transmitted messages and of stored data (data authentication). Learning Objectives After studying this chapter, you should be able to: ◆◆ Define the term message authentication code. ◆◆ List and explain the requirements for a message authentication code. ◆◆ Explain why a hash function used for message authentication needs to be secured. ◆◆ Understand the differences among preimage resistant, second preimage resistant, and collision resistant properties. ◆◆ Understand the operation of SHA-512. ◆◆ Present an overview of HMAC. ◆◆ Present an overview of the basic principles of public-key cryptosystems. ◆◆ Explain the two distinct uses of public-key cryptosystems. ◆◆ Present an overview of the RSA algorithm. ◆◆ Define Diffie–Hellman key exchange. ◆◆ Understand the man-in-the-middle attack. 80 chapter 3 / Public-Key Cryptography and Message Authentication Authentication Using Conventional Encryption It would seem possible to perform authentication simply by the use of symmetric encryption. If we assume that only the sender and receiver share a key (which is as it should be), then only the genuine sender would be able to encrypt a message successfully for the other participant, provided the receiver can recognize a valid message. Furthermore, if the message includes an error-detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper. If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit. In fact, symmetric encryption alone is not a suitable tool for data authentication. To give one simple example, in the ECB mode of encryption, if an attacker reorders the blocks of ciphertext, then each block will still decrypt successfully. However, the reordering may alter the meaning of the overall data sequence. Although sequence numbers may be used at some level (e.g., each IP packet), it is typically not the case that a separate sequence number will be associated with each b-bit block of plaintext. Thus, block reordering is a threat. Message Authentication without Message Encryption In this section, we examine several approaches to message authentication that do not rely on encryption. In all of these approaches, an authentication tag is generated and appended to each message for transmission. The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination. Because the approaches discussed in this section do not encrypt the message, message confidentiality is not provided. As was mentioned, message encryption by itself does not provide a secure form of authentication. However, it is possible to combine authentication and confidentiality in a single algorithm by encrypting a message plus its authentication tag. Typically, however, message authentication is provided as a separate function from message encryption. [DAVI89] suggests three situations in which message authentication without confidentiality is preferable: 1. There are a number of applications in which the same message is broadcast to a number of destinations. Two examples are notification to users that the network is now unavailable and an alarm signal in a control center. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity. Thus, the message must be broadcast in plaintext with an associated message authentication tag. The responsible system performs authentication. If a violation occurs, the other destination systems are alerted by a general alarm. 2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis with