Chapter 6
Cryptography and Symmetric Key Algorithms

THE CISSP EXAM TOPICS COVERED IN THIS CHAPTER INCLUDE:

- ✓ **Domain 2: Asset Security**
  - 2.5 Determine data security controls
    - 2.5.1 Understand data states
- ✓ **Domain 3: Security Architecture and Engineering**
  - 3.5 Assess and mitigate the vulnerabilities of security architectures, designs, and solution elements
    - 3.5.4 Cryptographic systems
  - 3.9 Apply cryptography
    - 3.9.1 Cryptographic lifecycle (e.g., key management, algorithm selection)
    - 3.9.2 Cryptographic methods (e.g., symmetric, asymmetric, elliptic curves)
    - 3.9.6 Nonrepudiation
    - 3.9.7 Integrity (e.g. hashing)

Cryptography provides confidentiality, integrity, authentication, and nonrepudiation for sensitive information while it is stored (at rest), traveling across a network (in transit), and existing in memory (in use). Cryptography is an extremely important security technology that is embedded in many of the controls used to protect

information from unauthorized visibility and use.

Over the years, mathematicians and computer scientists have developed a series of increasingly complex cryptographic algorithms designed to increase the level of protection provided to data. While cryptographers spent time developing strong encryption algorithms, hackers and governments alike devoted significant resources to undermining them. This led to an “arms race” in cryptography and resulted in the development of the extremely sophisticated algorithms in use today.

This chapter looks at the history of cryptography, the basics of cryptographic communications, and the fundamental principles of private key cryptosystems. The next chapter continues the discussion of cryptography by examining public key cryptosystems and the various techniques attackers use to defeat cryptography.

Historical Milestones in Cryptography

Since the earliest civilizations, human beings have devised various systems of written communication, ranging from ancient hieroglyphics written on cave walls to flash storage devices stuffed with encyclopedias full of information in modern English. As long we've been communicating, we've used secretive means to hide the true meaning of those communications from prying eyes. Ancient societies used a complex system of secret symbols to represent safe places to stay during times of war. Modern civilizations use a variety of codes and ciphers to facilitate private communication between individuals and groups. In the following sections, you'll look at the evolution of modern cryptography and several famous attempts to covertly intercept and decipher encrypted communications.

Caesar Cipher

One of the earliest known cipher systems was used by Julius Caesar to communicate with Cicero in Rome while he was conquering Europe. Caesar knew that there were several risks when sending messages—one of the messengers might be an enemy spy or might be ambushed while en route to the deployed forces. For that reason, Caesar developed a cryptographic system now known as the *Caesar cipher*. The system is extremely simple. To encrypt a message, you simply shift each letter of the alphabet three places to the right. For example, *A* would become *D*, and *B* would become *E*. If you reach the end of the alphabet during this process, you simply wrap around to the beginning so that *X* becomes *A*, *Y* becomes *B*, and *Z* becomes *C*. For this reason, the Caesar cipher also became known as the ROT3 (or Rotate 3) cipher. The Caesar cipher is a substitution cipher that is mono-alphabetic.

Although the Caesar cipher uses a shift of 3, the more general shift cipher uses the same algorithm to shift any number of characters desired by the user. For example, the ROT12 cipher

would turn an *A* into an *M*, a *B* into an *N*, and so on.

Here's an example of the Caesar cipher in action. The first line contains the original sentence, and the second line shows what the sentence looks like when it is encrypted using the Caesar cipher.

```
THE DIE HAS BEEN CAST
WKH GLH KDV EHHQ FDVW
```

To decrypt the message, you simply shift each letter three places to the left.

Although the Caesar cipher is easy to use, it's also easy to crack. It's vulnerable to a type of attack known as *frequency analysis*. The most common letters in the English language are *E*, *T*, *A*, *O*, *N*, *R*, *I*, *S*, and *H*. An attacker seeking to break a Caesar-style cipher encoding a message that was written in English merely needs to find the most common letters in the encrypted text and experiment with substitutions of these common letters to help determine the pattern.

American Civil War

Between the time of Caesar and the early years of the United States, scientists and mathematicians made significant advances beyond the early ciphers used by ancient civilizations. During the American Civil War, Union and Confederate troops both used relatively advanced cryptographic systems to secretly communicate along the front lines because each side was tapping into the telegraph lines to spy on the other side. These systems used complex combinations of word substitutions and transposition (see the section "Ciphers," later in this chapter, for more details) to attempt to defeat enemy decryption efforts. Another system used widely during the Civil War was a series of flag signals developed by army doctor Albert J. Myer.

Photos of many of the items discussed in this chapter are available at www .nsa.gov/about/cryptologic_heritage/museum.

Ultra vs. Enigma

Americans weren't the only ones who expended significant resources in the pursuit of superior code-making machines. Prior to World War II, the German military-industrial complex adapted a commercial code machine nicknamed Enigma for government use. This machine used a series of three to six rotors to implement an extremely complicated substitution cipher. The only possible way to decrypt the message with contemporary technology was to use a similar machine with the same rotor settings used by the transmitting device. The Germans recognized the importance of safeguarding these devices and made it extremely difficult for the Allies to acquire one.

The Allied forces began a top-secret effort known by the code name Ultra to attack the Enigma codes. Eventually, their efforts paid off when the Polish military successfully reconstructed an Enigma prototype and shared their findings with British and American cryptology experts. The Allies, led by Alan Turing, successfully broke the Enigma code in 1940, and historians credit this triumph as playing a significant role in the eventual defeat of the Axis powers. The story of the Allies' effort to crack the Enigma has been popularized in famous films including *U-571* and *The Imitation Game*.

The Japanese used a similar machine, known as the Japanese Purple Machine, during World War II. A significant American attack on this cryptosystem resulted in breaking the Japanese code prior to the end of the war. The Americans were aided by the fact that Japanese communicators used very formal message formats that resulted in a large amount of similar text in multiple messages, easing the cryptanalytic effort.

Cryptographic Basics

The study of any science must begin with a discussion of some of the fundamental principles upon which it is built. The following sections lay this foundation with a review of the goals of cryptography, an overview of the basic concepts of cryptographic technology, and a look at the major mathematical principles used by cryptographic systems.

Goals of Cryptography

Security practitioners use cryptographic systems to meet four fundamental goals: confidentiality, integrity, authentication, and nonrepudiation. Achieving each of these goals requires the satisfaction of a number of design requirements, and not all cryptosystems are intended to achieve all four goals. In the following sections, we'll examine each goal in detail and give a brief description of the technical requirements necessary to achieve it.

Confidentiality

Confidentiality ensures that data remains private in three different situations: when it is at rest, when it is in transit, and when it is in use.

Confidentiality is perhaps the most widely cited goal of cryptosystems—the preservation of secrecy for stored information or for communications between individuals and groups. Two main types of cryptosystems enforce confidentiality.

- *Symmetric cryptosystems* use a shared secret key available to all users of the cryptosystem.
- *Asymmetric cryptosystems* use individual combinations of public and private keys for each user of the system. Both of these concepts are explored in the section "Modern Cryptography" later in this chapter.

.The concept of protecting data at rest and data in transit is

often covered on the CISSP exam. You should also know that data in transit is also commonly called data *on the wire*, referring to the network cables that carry data communications.

When developing a cryptographic system for the purpose of providing confidentiality, you must think about three different types of data.

- *Data at rest*, or stored data, is that which resides in a permanent location awaiting access. Examples of data at rest include data stored on hard drives, backup tapes, cloud storage services, USB devices, and other storage media.
- *Data in motion*, or data on the wire, is data being transmitted across a network between two systems. Data in motion might be traveling on a corporate network, a wireless network, or the public internet.
- *Data in use* is data that is stored in the active memory of a computer system where it may be accessed by a process running on that system.

Each of these situations poses different types of confidentiality risks that cryptography can protect against. For example, data in motion may be susceptible to eavesdropping attacks, whereas data at rest is more susceptible to the theft of physical devices. Data in use may be accessed by unauthorized processes if the operating system does not properly implement process isolation.

Integrity

Integrity ensures that data is not altered without authorization. If integrity mechanisms are in place, the recipient of a message can be certain that the message received is identical to the message that was sent. Similarly, integrity checks can ensure that stored data was not altered between the time it was created and the time it was accessed. Integrity controls protect against all forms of alteration, including intentional alteration by a third party attempting to insert false information, intentional deletion of portions of the data, and unintentional alteration by faults in the transmission process.

Message integrity is enforced through the use of encrypted message digests, known as *digital signatures*, created upon transmission of a message. The recipient of the message simply verifies that the message's digital signature is valid, ensuring that the message was not altered in transit. Integrity can be enforced by both public and secret key cryptosystems. This concept is discussed in detail in Chapter 7, "PKI and Cryptographic Applications." The use of cryptographic hash functions to protect file integrity is discussed in Chapter 21, "Malicious Code and Application Attacks."

Authentication

Authentication verifies the claimed identity of system users and is a major function of cryptosystems. For example, suppose that Bob wants to establish a communications session with Alice and they are both participants in a shared secret communications system. Alice might use a challenge-response authentication technique to ensure that Bob is who he claims to be.

Figure 6.1 shows how this challenge-response protocol would work in action. In this example, the shared-secret code used by Alice and Bob is quite simple—the letters of each word are simply reversed. Bob first contacts Alice and identifies himself. Alice then sends a challenge message to Bob, asking him to encrypt a short message using the secret code known only to Alice and Bob. Bob replies with the encrypted message. After Alice verifies that the encrypted message is correct, she trusts that Bob himself is truly on the other end of the connection.

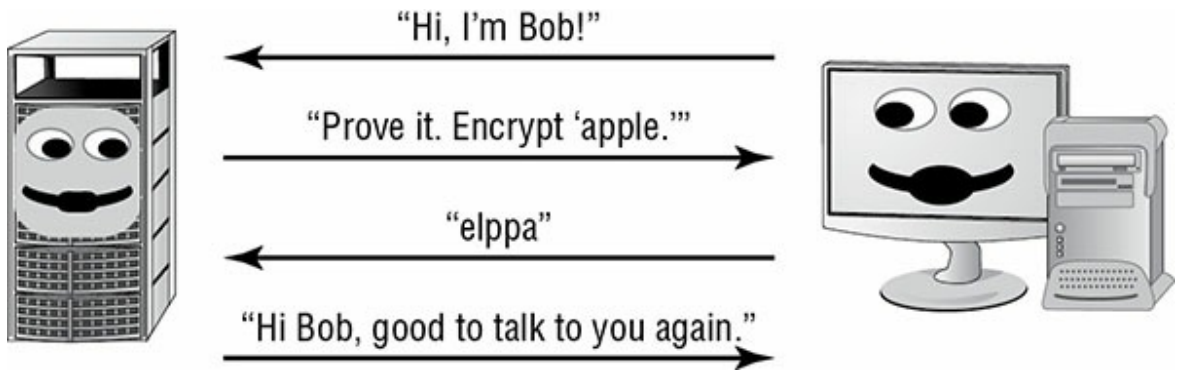

FIGURE 6.1 Challenge-response authentication protocol

Nonrepudiation

Nonrepudiation provides assurance to the recipient that the message was originated by the sender and not someone masquerading as the sender. It also prevents the sender from claiming that they never sent the message in the first place (also known as *repudiating* the message). Secret key, or symmetric key, cryptosystems (such as simple substitution ciphers) do not provide this guarantee of nonrepudiation. If Jim and Bob participate in a secret key communication system, they can both produce the same encrypted message using their shared secret key. Nonrepudiation is offered only by public key, or asymmetric, cryptosystems, a topic discussed in greater detail in Chapter 7.

Cryptography Concepts

As with any science, you must be familiar with certain terminology before studying cryptography. Let's take a look at a few of the key terms used to describe codes and ciphers. Before a message is put into a coded form, it is known as a *plaintext* message and is represented by the letter P when encryption functions are described. The sender of a message uses a cryptographic algorithm to *encrypt* the plaintext message and produce a *ciphertext* message, represented by the letter *C*. This message is transmitted by some physical or electronic means to the recipient. The recipient then uses a predetermined algorithm to decrypt the ciphertext message and retrieve the plaintext version. (For an illustration of this process, see Figure 6.3 later in this chapter.)

All cryptographic algorithms rely on *keys* to maintain their security. For the most part, a key is nothing more than a number. It's usually a very large binary number, but it's a number nonetheless. Every algorithm has a specific *key space*. The key space is the range of values that are valid for use as a key for a specific algorithm. A key space is defined by its *bit size*. Bit size is nothing more than the number of binary bits (0s and 1s) in the key. The key space is the range between the key that has all 0s and the key that has all 1s. Or to state it another way, the key space is the range of numbers from 0 to 2^n, where n is the bit size of the key. So, a 128-bit key can have a value from 0 to 2^{128}

(which is roughly $3.40282367 \times 10^{38}$, a very big number!). It is absolutely critical to protect the security of secret keys. In fact, all of the security you gain from cryptography rests on your ability to keep the keys used private.

The Kerchoff Principle

All cryptography relies on algorithms. An *algorithm* is a set of rules, usually mathematical, that dictates how enciphering and deciphering processes are to take place. Most cryptographers follow the Kerchoff principle, a concept that makes algorithms known and public, allowing anyone to examine and test them. Specifically, the *Kerchoff principle* (also known as Kerchoff's assumption) is that a cryptographic system should be secure even if everything about the system, except the key, is public knowledge. The principle can be summed up as "The enemy knows the system."

A large number of cryptographers adhere to this principle, but not all agree. In fact, some believe that better overall security can be maintained by keeping both the algorithm and the key private. Kerchoff's adherents retort that the opposite approach includes the dubious practice of "security through obscurity" and believe that public exposure produces more activity and exposes more weaknesses more readily, leading to the abandonment of insufficiently strong algorithms and quicker adoption of suitable ones.

As you'll learn in this chapter and the next, different types of algorithms require different types of keys. In private key (or secret key) cryptosystems, all participants use a single shared key. In public key cryptosystems, each participant has their own pair of keys. Cryptographic keys are sometimes referred to as *cryptovariables*.

The art of creating and implementing secret codes and ciphers is known as *cryptography*. This practice is paralleled by the art of *cryptanalysis*—the study of methods to defeat codes and ciphers.

Together, cryptography and cryptanalysis are commonly referred to as *cryptology*. Specific implementations of a code or cipher in hardware and software are known as *cryptosystems*. Federal Information Processing Standard (FIPS) 140–2, "Security Requirements for Cryptographic Modules," defines the hardware and software requirements for cryptographic modules that the federal government uses.

Be sure to understand the meanings of the terms in this section before continuing your study of this chapter and the following chapter. They are essential to understanding the technical details of the cryptographic algorithms presented in the following sections.

Cryptographic Mathematics

Cryptography is no different from most computer science disciplines in that it finds its foundations in the science of mathematics. To fully understand cryptography, you must first understand the basics of binary mathematics and the logical operations used to manipulate binary values. The following sections present a brief look at some of the most fundamental concepts with which you should be familiar.

Boolean Mathematics

Boolean mathematics defines the rules used for the bits and bytes that form the nervous system of any computer. You're most likely familiar with the decimal system. It is a base 10 system in which an integer from 0 to 9 is used in each place and each place value is a multiple of 10. It's likely that our reliance on the decimal system has biological origins— human beings have 10 fingers that can be used to count.

Boolean math can be very confusing at first, but it's worth the investment of time to learn how logical functions work. You need to understand these concepts to truly understand the inner

workings of cryptographic algorithms.

Similarly, the computer's reliance upon the Boolean system has electrical origins. In an electrical circuit, there are only two possible states—on (representing the presence of electrical current) and off (representing the absence of electrical current). All computation performed by an electrical device must be expressed in these terms, giving rise to the use of Boolean computation in modern electronics. In general, computer scientists refer to the on condition as a *true* value and the off condition as a *false* value.

Logical Operations

The Boolean mathematics of cryptography uses a variety of logical functions to manipulate data. We'll take a brief look at several of these operations.

AND

The AND operation (represented by the ∧ symbol) checks to see whether two values are both true. The truth table that follows illustrates all four possible outputs for the AND function. Remember, the AND function takes only two variables as input. In Boolean math, there are only two possible values for each of these variables, leading to four possible inputs to the AND function. It's this finite number of possibilities that makes it extremely easy for computers to implement logical functions in hardware. Notice in the following truth table that only one combination of inputs (where both inputs are true) produces an output value of true:

X	**Y**	**X ∧ Y**
0	0	0
0	1	0
1	0	0
1	1	1

Logical operations are often performed on entire Boolean words rather than single values. Take a look at the following example:

```
X: 0 1 1 0 1 1 0 0
Y: 1 0 1 0 0 1 1 1
_____________________
X ∧ Y: 0 0 1 0 0 1 0 0
```

Notice that the AND function is computed by comparing the values of X and Y in each column. The output value is true only in columns where both X and Y are true.

OR

The OR operation (represented by the ∨ symbol) checks to see whether at least one of the input values is true. Refer to the following truth table for all possible values of the OR function. Notice that the only time the OR function returns a false value is when both of the input values are false:

X	**Y**	**X ∨ Y**
0	0	0
0	1	1
1	0	1
1	1	1

We'll use the same example we used in the previous section to show you what the output would be if X and Y were fed into the OR function rather than the AND function:

```
X: 0 1 1 0 1 1 0 0
Y: 1 0 1 0 0 1 1 1
_____________________
X ∨ Y: 1 1 1 0 1 1 1 1
```

NOT

The NOT operation (represented by the ~ or ! symbol) simply reverses the value of an input variable. This function operates on only one variable at a time. Here's the truth table for the NOT function:

X	**~X**
0	1

1	0

In this example, you take the value of X from the previous examples and run the NOT function against it:

```
X: 0 1 1 0 1 1 0 0
_____________________________
~X: 1 0 0 1 0 0 1 1
```

Exclusive OR

The final logical function you'll examine in this chapter is perhaps the most important and most commonly used in cryptographic applications—the exclusive OR (XOR) function. It's referred to in mathematical literature as the XOR function and is commonly represented by the ⊕ symbol. The XOR function returns a true value when only one of the input values is true. If both values are false or both values are true, the output of the XOR function is false. Here is the truth table for the XOR operation:

X	**Y**	**X ⊕ Y**
0	0	0
0	1	1
1	0	1
1	1	0

The following operation shows the X and Y values when they are used as input to the XOR function:

```
X: 0 1 1 0 1 1 0 0
Y: 1 0 1 0 0 1 1 1
_____________________________
X ⊕ Y: 1 1 0 0 1 0 1 1
```

Modulo Function

The *modulo* function is extremely important in the field of cryptography. Think back to the early days when you first learned division. At that time, you weren't familiar with decimal numbers and compensated by showing a remainder value each time you performed

a division operation. Computers don't naturally understand the decimal system either, and these remainder values play a critical role when computers perform many mathematical functions. The modulo function is, quite simply, the remainder value left over after a division operation is performed.

The modulo function is just as important to cryptography as the logical operations are. Be sure you're familiar with its functionality and can perform simple modular math.

The modulo function is usually represented in equations by the abbreviation *mod*, although it's also sometimes represented by the % operator. Here are several inputs and outputs for the modulo function:

```
8 mod 6 = 2
6 mod 8 = 6
10 mod 3 = 1
10 mod 2 = 0
32 mod 8 = 0
```

We'll revisit this function in Chapter 7 when we explore the RSA public key encryption algorithm (named after Rivest, Shamir, and Adleman, its inventors).

One-Way Functions

A *one-way function* is a mathematical operation that easily produces output values for each possible combination of inputs but makes it impossible to retrieve the input values. Public key cryptosystems are all based on some sort of one-way function. In practice, however, it's never been proven that any specific known function is truly one way. Cryptographers rely on functions that they believe are one way, but it's always possible that they might be broken by future cryptanalysts.

Here's an example. Imagine you have a function that multiplies three numbers together. If you restrict the input values to single-digit numbers, it's a relatively straightforward matter to reverse-engineer this function and determine the possible input values by looking at the numerical output. For example, the output value 15 was created by

using the input values 1, 3, and 5. However, suppose you restrict the input values to five-digit prime numbers. It's still quite simple to obtain an output value by using a computer or a good calculator, but reverse-engineering is not quite so simple. Can you figure out what three prime numbers were used to obtain the output value 10,718,488,075,259? Not so simple, eh? (As it turns out, the number is the product of the prime numbers 17,093; 22,441; and 27,943.) There are actually 8,363 five-digit prime numbers, so this problem might be attacked using a computer and a brute-force algorithm, but there's no easy way to figure it out in your head, that's for sure!

Nonce

Cryptography often gains strength by adding randomness to the encryption process. One method by which this is accomplished is through the use of a nonce. A *nonce* is a random number that acts as a placeholder variable in mathematical functions. When the function is executed, the nonce is replaced with a random number generated at the moment of processing for one-time use. The nonce must be a unique number each time it is used. One of the more recognizable examples of a nonce is an initialization vector (IV), a random bit string that is the same length as the block size and is XORed with the message. IVs are used to create unique ciphertext every time the same message is encrypted using the same key.

Zero-Knowledge Proof

One of the benefits of cryptography is found in the mechanism to prove your knowledge of a fact to a third party without revealing the fact itself to that third party. This is often done with passwords and other secret authenticators.

The classic example of a *zero-knowledge proof* involves two individuals: Peggy and Victor. Peggy knows the password to a secret door located inside a circular cave, as shown in Figure 6.2. Victor would like to buy the password from Peggy, but he wants Peggy to prove that she knows the password before paying her for it. Peggy doesn't want to tell Victor the password for fear that he won't pay later. The zero-knowledge proof can solve their dilemma.

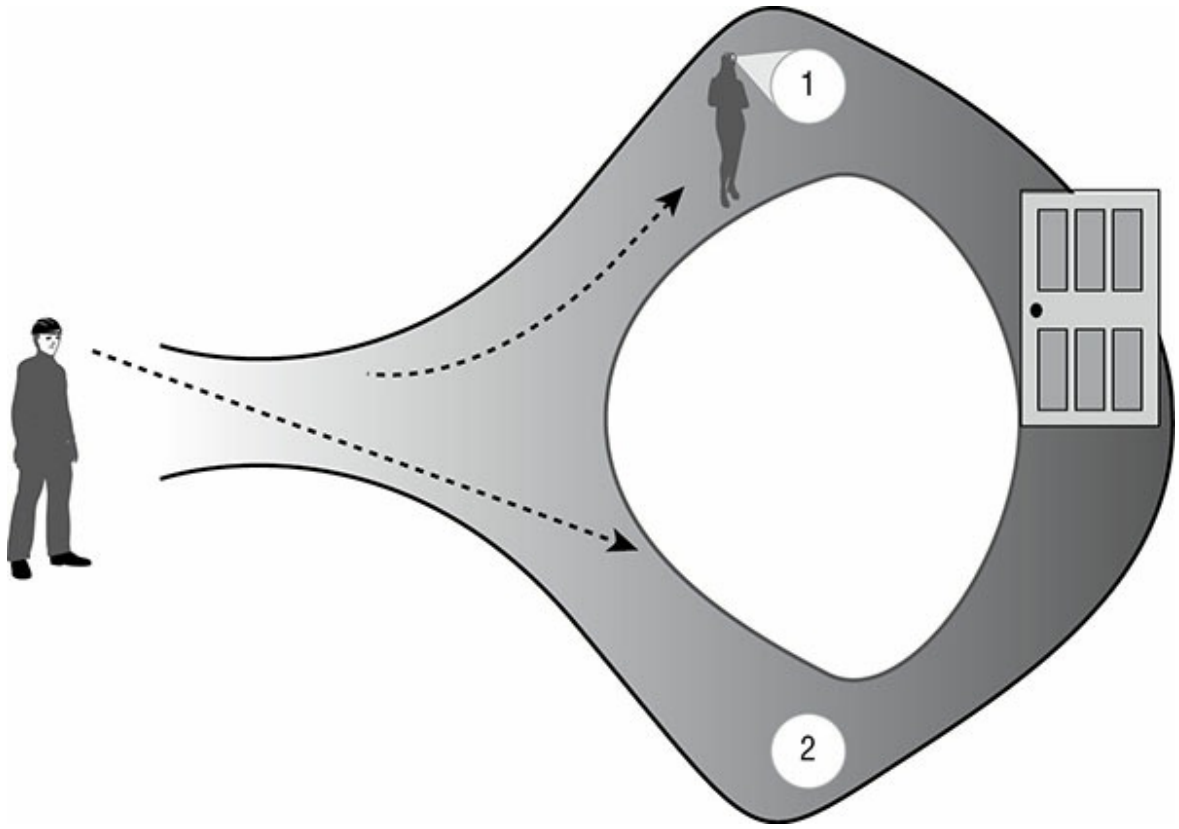

FIGURE 6.2 The magic door

Victor can stand at the entrance to the cave and watch Peggy depart down the path. Peggy then reaches the door and opens it using the password. She then passes through the door and returns via path 2. Victor saw her leave down path 1 and return via path 2, proving that she must know the correct password to open the door.

Split Knowledge

When the information or privilege required to perform an operation is divided among multiple users, no single person has sufficient privileges to compromise the security of an environment. This separation of duties and two-person control contained in a single solution is called *split knowledge*. The best example of split knowledge is seen in the concept of *key escrow*. Using key escrow, cryptographic keys, digital signatures, and even digital certificates can be stored or backed up in a special database called the *key escrow database*. In the event a user loses or damages their key, that key can be extracted from

the backup. However, if only a single key escrow recovery agent exists, there is opportunity for fraud and abuse of this privilege. *M of N Control* requires that a minimum number of agents (*M*) out of the total number of agents (*N*) work together to perform high-security tasks. So, implementing three of eight controls would require three people out of the eight with the assigned work task of key escrow recovery agent to work together to pull a single key out of the key escrow database (thereby also illustrating that *M* is always less than or equal to *N*).

Work Function

You can measure the strength of a cryptography system by measuring the effort in terms of cost and/or time using a *work function* or work factor. Usually the time and effort required to perform a complete brute-force attack against an encryption system is what the work function represents. The security and protection offered by a cryptosystem is directly proportional to the value of the work function/factor. The size of the work function should be matched against the relative value of the protected asset. The work function need be only slightly greater than the time value of that asset. In other words, all security, including cryptography, should be cost effective and cost efficient. Spend no more effort to protect an asset than it warrants, but be sure to provide sufficient protection. Thus, if information loses its value over time, the work function needs to be only large enough to ensure protection until the value of the data is gone.

In addition to understanding the length of time that the data will have value, security professionals selecting cryptographic systems must also understand how emerging technologies may impact cipher-cracking efforts. For example, researchers may discover a flaw in a cryptographic algorithm next year that renders information protected with that algorithm insecure. Similarly, technological advancements in cloud-based parallel computing and quantum computing may make brute-force efforts much more feasible down the road.

Ciphers

Cipher systems have long been used by individuals and governments