Symmetric Cryptography

You've learned the basic concepts underlying symmetric key cryptography, asymmetric key cryptography, and hashing functions. In the following sections, we'll take an in-depth look at several common symmetric cryptosystems: the Data Encryption Standard (DES), Triple DES (3DES), International Data Encryption Algorithm (IDEA), Blowfish, Skipjack, and the Advanced Encryption Standard (AES).

Data Encryption Standard

The U.S. government published the Data Encryption Standard in 1977 as a proposed standard cryptosystem for all government communications. Because of flaws in the algorithm, cryptographers and the federal government no longer consider DES secure. It is widely believed that intelligence agencies routinely decrypt DES-encrypted information. DES was superseded by the Advanced Encryption Standard in December 2001. It is still important to understand DES because it is the building block of Triple DES (3DES), a strong encryption algorithm discussed in the next section.

DES is a 64-bit block cipher that has five modes of operation: Electronic Code Book (ECB) mode, Cipher Block Chaining (CBC) mode, Cipher Feedback (CFB) mode, output feedback (OFB) mode, and Counter (CTR) mode. These modes are explained in the following sections. All of the DES modes operate on 64 bits of plaintext at a time to generate 64-bit blocks of ciphertext. The key used by DES is 56 bits long.

DES uses a long series of exclusive OR (XOR) operations to generate the ciphertext. This process is repeated 16 times for each encryption/decryption operation. Each repetition is commonly referred to as a *round* of encryption, explaining the statement that DES performs 16 rounds of encryption.

As mentioned, DES uses a 56-bit key to drive the encryption and decryption process. However, you may read in some literature that DES uses a 64-bit key. This is not an inconsistency—there's a perfectly logical explanation. The DES specification calls for a 64-bit key. However, of those 64 bits, only 56 actually contain keying information. The remaining 8 bits are supposed to contain parity information to ensure that the other 56 bits are accurate. In practice, however, those parity bits are rarely used. You should commit the 56-bit figure to memory.

Electronic Code Book Mode

Electronic Code Book (ECB) mode is the simplest mode to understand and the least secure. Each time the algorithm processes a 64-bit block, it simply encrypts the block using the chosen secret key. This means that if the algorithm encounters the same block multiple times, it will produce the same encrypted block. If an enemy were eavesdropping on the communications, they could simply build a "code book" of all the possible encrypted values. After a sufficient number of blocks were gathered, cryptanalytic techniques could be used to decipher some of the blocks and break the encryption scheme.

This vulnerability makes it impractical to use ECB mode on all but the shortest transmissions. In everyday use, ECB is used only for exchanging small amounts of data, such as keys and parameters used to initiate other DES modes as well as the cells in a database.

Cipher Block Chaining Mode

In Cipher Block Chaining (CBC) mode, each block of unencrypted text is XORed with the block of ciphertext immediately preceding it before it is encrypted using the DES algorithm. The decryption process simply decrypts the ciphertext and reverses the XOR operation. CBC implements an IV and XORs it with the first block of the message, producing a unique output every time the operation is performed. The IV must be sent to the recipient, perhaps by tacking the IV onto the

front of the completed ciphertext in plain form or by protecting it with ECB mode encryption using the same key used for the message. One important consideration when using CBC mode is that errors propagate—if one block is corrupted during transmission, it becomes impossible to decrypt that block and the next block as well.

Cipher Feedback Mode

Cipher Feedback (CFB) mode is the streaming cipher version of CBC. In other words, CFB operates against data produced in real time. However, instead of breaking a message into blocks, it uses memory buffers of the same block size. As the buffer becomes full, it is encrypted and then sent to the recipients. Then the system waits for the next buffer to be filled as the new data is generated before it is in turn encrypted and then transmitted. Other than the change from preexisting data to real-time data, CFB operates in the same fashion as CBC. It uses an IV, and it uses chaining.

Output Feedback Mode

In output feedback (OFB) mode, DES operates in almost the same fashion as it does in CFB mode. However, instead of XORing an encrypted version of the previous block of ciphertext, DES XORs the plaintext with a seed value. For the first encrypted block, an initialization vector is used to create the seed value. Future seed values are derived by running the DES algorithm on the previous seed value. The major advantages of OFB mode are that there is no chaining function and transmission errors do not propagate to affect the decryption of future blocks.

Counter Mode

DES that is run in Counter (CTR) mode uses a stream cipher similar to that used in CFB and OFB modes. However, instead of creating the seed value for each encryption/decryption operation from the results of the previous seed values, it uses a simple counter that increments for each operation. As with OFB mode, errors do not propagate in CTR mode.

 .CTR mode allows you to break an encryption or decryption operation into multiple independent steps. This makes CTR mode well suited for use in parallel computing.

Triple DES

As mentioned in previous sections, the Data Encryption Standard's (DES) 56-bit key is no longer considered adequate in the face of modern cryptanalytic techniques and supercomputing power. However, an adapted version of DES, Triple DES (3DES), uses the same algorithm to produce a more secure encryption.

There are four versions of 3DES. The first simply encrypts the plaintext three times, using three different keys: K_1, K_2, and K_3. It is known as DES-EEE3 mode (the *E*s indicate that there are three encryption operations, whereas the numeral 3 indicates that three different keys are used). DES-EEE3 can be expressed using the following notation, where `E(K,P)` represents the encryption of plaintext *P* with key *K*:

```
E(K1,E(K2,E(K3,P)))
```

DES-EEE3 has an effective key length of 168 bits.

The second variant (DES-EDE3) also uses three keys but replaces the second encryption operation with a decryption operation.

```
E(K1,D(K2,E(K3,P)))
```

The third version of 3DES (DES-EEE2) uses only two keys, K_1 and K_2, as follows:

```
E(K1,E(K2,E(K1,P)))
```

The fourth variant of 3DES (DES-EDE2) also uses two keys but uses a decryption operation in the middle.

```
E(K1,D(K2,E(K1,P)))
```

Both the third and fourth variants have an effective key length of 112 bits.


Technically, there is a fifth variant of 3DES, DES-EDE1, which uses only one cryptographic key. However, it results in the same algorithm as standard DES, which is unacceptably weak for most applications. It is provided only for backward-compatibility purposes.

These four variants of 3DES were developed over the years because several cryptologists put forth theories that one variant was more secure than the others. However, the current belief is that all modes are equally secure.


Take some time to understand the variants of 3DES. Sit down with a pencil and paper and be sure you understand the way each variant uses two or three keys to achieve stronger encryption.


This discussion raises an obvious question—what happened to Double DES (2DES)? You'll read in Chapter 7 that Double DES was tried but quickly abandoned when it was proven that an attack existed that rendered it no more secure than standard DES.

International Data Encryption Algorithm

The International Data Encryption Algorithm (IDEA) block cipher was developed in response to complaints about the insufficient key length of the DES algorithm. Like DES, IDEA operates on 64-bit blocks of plaintext/ciphertext. However, it begins its operation with a 128-bit key. This key is broken up in a series of operations into 52 16-bit

subkeys. The subkeys then act on the input text using a combination of XOR and modulus operations to produce the encrypted/decrypted version of the input message. IDEA is capable of operating in the same five modes used by DES: ECB, CBC, CFB, OFB, and CTR.

All of this material on key length block size and the number of rounds of encryption may seem dreadfully boring; however, it's important material, so be sure to brush up on it while preparing for the exam.

The IDEA algorithm was patented by its Swiss developers. However, the patent expired in 2012, and it is now available for unrestricted use. One popular implementation of IDEA is found in Phil Zimmerman's popular Pretty Good Privacy (PGP) secure email package. Chapter 7 covers PGP in further detail.

Blowfish

Bruce Schneier's Blowfish block cipher is another alternative to DES and IDEA. Like its predecessors, Blowfish operates on 64-bit blocks of text. However, it extends IDEA's key strength even further by allowing the use of variable-length keys ranging from a relatively insecure 32 bits to an extremely strong 448 bits. Obviously, the longer keys will result in a corresponding increase in encryption/decryption time. However, time trials have established Blowfish as a much faster algorithm than both IDEA and DES. Also, Mr. Schneier released Blowfish for public use with no license required. Blowfish encryption is built into a number of commercial software products and operating systems. A number of Blowfish libraries are also available for software developers.

Skipjack

The Skipjack algorithm was approved for use by the U.S. government in Federal Information Processing Standard (FIPS) 185, the Escrowed Encryption Standard (EES). Like many block ciphers, Skipjack operates on 64-bit blocks of text. It uses an 80-bit key and supports

the same four modes of operation supported by DES. Skipjack was quickly embraced by the U.S. government and provides the cryptographic routines supporting the Clipper and Capstone encryption chips.

However, Skipjack has an added twist—it supports the escrow of encryption keys. Two government agencies, the National Institute of Standards and Technology (NIST) and the Department of the Treasury, hold a portion of the information required to reconstruct a Skipjack key. When law enforcement authorities obtain legal authorization, they contact the two agencies, obtain the pieces of the key, and are able to decrypt communications between the affected parties.

Skipjack and the Clipper chip were not embraced by the cryptographic community at large because of its mistrust of the escrow procedures in place within the U.S. government.

Rivest Cipher 5 (RC5)

Rivest Cipher 5, or RC5, is a symmetric algorithm patented by Rivest–Shamir–Adleman (RSA) Data Security, the people who developed the RSA asymmetric algorithm. RC5 is a block cipher of variable block sizes (32, 64, or 128 bits) that uses key sizes between 0 (zero) length and 2,040 bits. RC5 is an improvement on an older algorithm called RC2 that is no longer considered secure. RSA also developed a new algorithm, RC6, that built upon RC5, but it has not been widely adopted.

RC5 is the subject of brute-force cracking attempts. A large-scale effort leveraging massive community computing resources cracked a message encrypted using RC5 with a 64-bit key, but this effort took more than four years to crack a single message.

Advanced Encryption Standard

In October 2000, the National Institute of Standards and Technology announced that the Rijndael (pronounced “rhine-doll”) block cipher

had been chosen as the replacement for DES. In November 2001, NIST released FIPS 197, which mandated the use of AES/Rijndael for the encryption of all sensitive but unclassified data by the U.S. government.

The AES cipher allows the use of three key strengths: 128 bits, 192 bits, and 256 bits. AES only allows the processing of 128-bit blocks, but Rijndael exceeded this specification, allowing cryptographers to use a block size equal to the key length. The number of encryption rounds depends on the key length chosen:

- 128-bit keys require 10 rounds of encryption.
- 192-bit keys require 12 rounds of encryption.
- 256-bit keys require 14 rounds of encryption.

Twofish

The Twofish algorithm developed by Bruce Schneier (also the creator of Blowfish) was another one of the AES finalists. Like Rijndael, Twofish is a block cipher. It operates on 128-bit blocks of data and is capable of using cryptographic keys up to 256 bits in length.

Twofish uses two techniques not found in other algorithms:

Prewhitening involves XORing the plaintext with a separate subkey before the first round of encryption.

Postwhitening uses a similar operation after the 16th round of encryption.

AES is just one of the many symmetric encryption algorithms you need to be familiar with. Table 6.2 lists several common and well-known symmetric encryption algorithms along with their block size and key size.


The information in Table 6.2 is great fodder for CISSP exam questions. Take care to memorize it before sitting for the exam.

TABLE 6.2 Symmetric memorization chart

Name	Block size	Key size
Advanced Encryption Standard (AES)	128	128, 192, 256
Rijndael	Variable	128, 192, 256
Blowfish (often used in SSH)	64	32–448
Data Encryption Standard (DES)	64	56
IDEA (used in PGP)	64	128
Rivest Cipher 2 (RC2)	64	128
Rivest Cipher 5 (RC5)	32, 64, 128	0–2,040
Skipjack	64	80
Triple DES (3DES)	64	112 or 168
Twofish	128	1–256

Symmetric Key Management

Because cryptographic keys contain information essential to the security of the cryptosystem, it is incumbent upon cryptosystem users and administrators to take extraordinary measures to protect the security of the keying material. These security measures are collectively known as *key management practices*. They include safeguards surrounding the creation, distribution, storage, destruction, recovery, and escrow of secret keys.

Creation and Distribution of Symmetric Keys

As previously mentioned, one of the major problems underlying symmetric encryption algorithms is the secure distribution of the secret keys required to operate the algorithms. The three main methods used to exchange secret keys securely are offline distribution,

public key encryption, and the Diffie–Hellman key exchange algorithm.

Offline Distribution The most technically simple method involves the physical exchange of key material. One party provides the other party with a sheet of paper or piece of storage media containing the secret key. In many hardware encryption devices, this key material comes in the form of an electronic device that resembles an actual key that is inserted into the encryption device. However, every offline key distribution method has its own inherent flaws. If keying material is sent through the mail, it might be intercepted. Telephones can be wiretapped. Papers containing keys might be inadvertently thrown in the trash or lost.

Public Key Encryption Many communicators want to obtain the speed benefits of secret key encryption without the hassles of key distribution. For this reason, many people use public key encryption to set up an initial communications link. Once the link is successfully established and the parties are satisfied as to each other's identity, they exchange a secret key over the secure public key link. They then switch communications from the public key algorithm to the secret key algorithm and enjoy the increased processing speed. In general, secret key encryption is thousands of times faster than public key encryption.

Diffie-Hellman In some cases, neither public key encryption nor offline distribution is sufficient. Two parties might need to communicate with each other, but they have no physical means to exchange key material, and there is no public key infrastructure in place to facilitate the exchange of secret keys. In situations like this, key exchange algorithms like the Diffie–Hellman algorithm prove to be extremely useful mechanisms.


.Secure RPC (S-RPC) employs Diffie–Hellman for key exchange.

About the Diffie–Hellman Algorithm

The Diffie–Hellman algorithm represented a major advance in the state of cryptographic science when it was released in 1976. It's still in use today. The algorithm works as follows:

1. The communicating parties (we'll call them Richard and Sue) agree on two large numbers: p (which is a prime number) and g (which is an integer) such that $1 < g < p$.
2. Richard chooses a random large integer r and performs the following calculation:

   $R = g^r \bmod p$

3. Sue chooses a random large integer s and performs the following calculation:

   $S = g^s \bmod p$

4. Richard sends R to Sue and Sue sends S to Richard.
5. Richard then performs the following calculation:

   $K = S^r \bmod p$

6. Sue then performs the following calculation:

   $K = R^s \bmod p$

At this point, Richard and Sue both have the same value, K, and can use this for secret key communication between the two parties.

Storage and Destruction of Symmetric Keys

Another major challenge with the use of symmetric key cryptography is that all of the keys used in the cryptosystem must be kept secure. This includes following best practices surrounding the storage of encryption keys:

- Never store an encryption key on the same system where encrypted data resides. This just makes it easier for the attacker!
- For sensitive keys, consider providing two different individuals

with half of the key. They then must collaborate to re-create the entire key. This is known as the principle of *split knowledge* (discussed earlier in this chapter).

When a user with knowledge of a secret key leaves the organization or is no longer permitted access to material protected with that key, the keys must be changed, and all encrypted materials must be reencrypted with the new keys. The difficulty of destroying a key to remove a user from a symmetric cryptosystem is one of the main reasons organizations turn to asymmetric algorithms, as discussed in Chapter 7.

Key Escrow and Recovery

Cryptography is a powerful tool. Like most tools, it can be used for a number of beneficent purposes, but it can also be used with malicious intent. To gain a handle on the explosive growth of cryptographic technologies, governments around the world have floated ideas to implement key escrow systems. These systems allow the government, under limited circumstances such as a court order, to obtain the cryptographic key used for a particular communication from a central storage facility.

There are two major approaches to key escrow that have been proposed over the past decade.

Fair Cryptosystems In this escrow approach, the secret keys used in a communication are divided into two or more pieces, each of which is given to an independent third party. Each of these pieces is useless on its own but may be recombined to obtain the secret key. When the government obtains legal authority to access a particular key, it provides evidence of the court order to each of the third parties and then reassembles the secret key.

Escrowed Encryption Standard This escrow approach provides the government with a technological means to decrypt ciphertext. This standard is the basis behind the Skipjack algorithm discussed earlier in this chapter.

It's highly unlikely that government regulators will ever overcome the legal and privacy hurdles necessary to implement key escrow on a

widespread basis. The technology is certainly available, but the general public will likely never accept the potential government intrusiveness it facilitates.

Cryptographic Lifecycle

With the exception of the one-time pad, all cryptographic systems have a limited life span. Moore's law, a commonly cited trend in the advancement of computing power, states that the processing capabilities of a state-of-the-art microprocessor will double approximately every two years. This means that, eventually, processors will reach the amount of strength required to simply guess the encryption keys used for a communication.

Security professionals must keep this cryptographic lifecycle in mind when selecting an encryption algorithm and have appropriate governance controls in place to ensure that the algorithms, protocols, and key lengths selected are sufficient to preserve the integrity of a cryptosystem for however long it is necessary to keep the information it is protecting secret. Security professionals can use the following algorithm and protocol governance controls:

- Specifying the cryptographic algorithms (such as AES, 3DES, and RSA) acceptable for use in an organization
- Identifying the acceptable key lengths for use with each algorithm based on the sensitivity of information transmitted
- Enumerating the secure transaction protocols (such as SSL and TLS) that may be used

For example, if you're designing a cryptographic system to protect the security of business plans that you expect to execute next week, you don't need to worry about the theoretical risk that a processor capable of decrypting them might be developed a decade from now. On the other hand, if you're protecting the confidentiality of information that could be used to construct a nuclear bomb, it's virtually certain that you'll still want that information to remain secret 10 years in the future!

Summary

Cryptographers and cryptanalysts are in a never-ending race to develop more secure cryptosystems and advanced cryptanalytic techniques designed to circumvent those systems.

Cryptography dates back as early as Caesar and has been an ongoing topic for study for many years. In this chapter, you learned some of the fundamental concepts underlying the field of cryptography, gained a basic understanding of the terminology used by cryptographers, and looked at some historical codes and ciphers used in the early days of cryptography.

This chapter also examined the similarities and differences between symmetric key cryptography (where communicating parties use the same key) and asymmetric key cryptography (where each communicator has a pair of public and private keys).

We then analyzed some of the symmetric algorithms currently available and their strengths and weaknesses. We wrapped up the chapter by taking a look at the cryptographic lifecycle and the role of algorithm/protocol governance in enterprise security.

The next chapter expands this discussion to cover contemporary public key cryptographic algorithms. Additionally, some of the common cryptanalytic techniques used to defeat both types of cryptosystems will be explored.