Static Password Token	The device contains a password that is physically hidden (not visible to the possessor) but that is transmitted for each authentication.
Synchronous Dynamic Password Token	A timer is used to rotate through various combinations produced by a cryptographic algorithm.
Temporal-based access control	Restricting access to systems and/or data based upon time.
Temporal role-based access control	Restricting access to systems and/or data based on both time and role.
Trust Path	A series of trust relationships that authentication requests must follow between domains.
View-based access control (VBAC)	Restricting access to data, typically within databases by manipulating the output "views" of a database search.
World Wide Web	A central information space and repository for documents and other resources.

SSCP Chapter 2 Quiz

Tiempo permitido

ilimitado (tiempo estimado requerido: 2:00:00)

Intentos

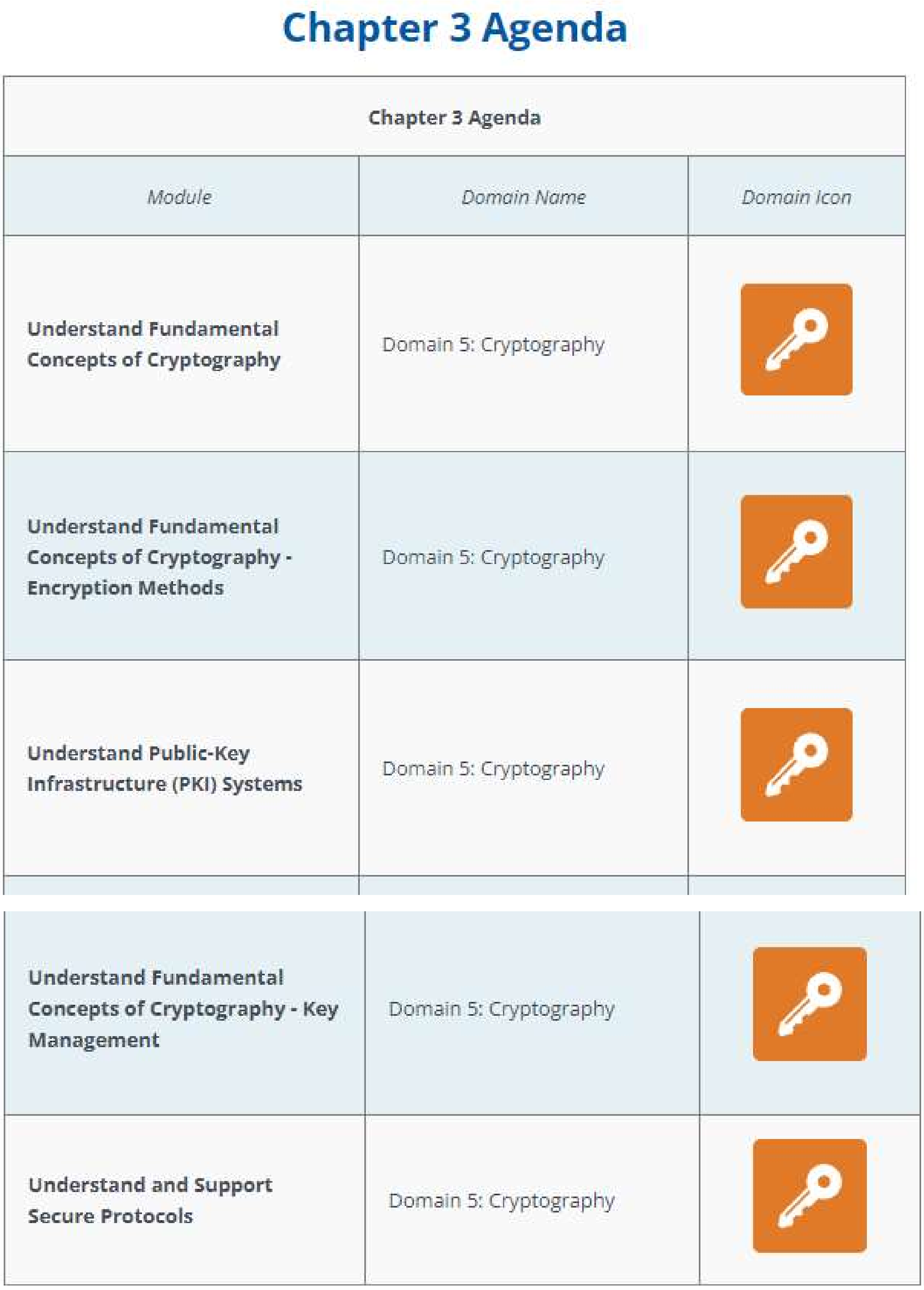
Permitido - ilimitado, Terminado - 1

Instrucciones

Antes de enviar el cuestionario, tendrá la oportunidad de volver a las preguntas que quizás se haya perdido o que aún no haya respondido.
Puede enviar las respuestas de su cuestionario en cualquier momento.

Haga clic en "Iniciar prueba" para comenzar Intento 2.

Chapter 3: Cryptography



Chapter 3 Agenda

Chapter 3 Agenda		
Module	*Domain Name*	*Domain Icon*
Understand Fundamental Concepts of Cryptography	Domain 5: Cryptography	🔑
Understand Fundamental Concepts of Cryptography - Encryption Methods	Domain 5: Cryptography	🔑
Understand Public-Key Infrastructure (PKI) Systems	Domain 5: Cryptography	🔑
Understand Fundamental Concepts of Cryptography - Key Management	Domain 5: Cryptography	🔑
Understand and Support Secure Protocols	Domain 5: Cryptography	🔑

Chapter Objectives

After completing this chapter, the participant will be able to:

1. Describe encryption.
2. Understand encryption terminology.
3. Use the exclusive OR (XOR) function.
4. Describe confusion, diffusion and avalanche.
5. Discuss encryption benefits.
6. Define data classification.
7. Discuss the legal implications of encryption.
8. Discuss training requirements.
9. Describe the methods of cryptography.
10. Compare symmetric and asymmetric cryptography.
11. Describe the advantages and disadvantages of symmetric and asymmetric cryptography.
12. Differentiate between messages at rest and messages in transit.
13. Differentiate between stream-based and block-based encryption.
14. Describe out-of-band key distribution.
15. Explain proof of origin.
16. Describe a public-key infrastructure (PKI) and its components.
17. Explain the field within a certificate.
18. Describe a web of trust.
19. Apply integrity checking.
20. Distinguish between hashing algorithms.
21. Describe the birthday paradox.
22. Explain the benefit of salting.
23. Differentiate between the mandatory access control (MAC) and the hash message authentication code (HMAC).
24. Understand the concepts of key management.
25. Explain Kerckhoff's Principle.
26. Discuss the Extensible Markup Language (XML) and American National Standards Institute (ANSI) specifications.
27. Understand key management requirements.
28. Understand the challenge of key storage and destruction.
29. Describe an escrow service.
30. Compare key distribution options.
31. Discuss key protection techniques.
32. Understand and support secure protocols.
33. Describe Secure Sockets Layer (SSL) and Transport Layer Security (TLS).
34. Discuss Internet Protocol security (IPsec) and its implementation.
35. Explain steganography.
36. Understand cryptanalysis and steganalysis.
37. Differentiate between different types of attacks.

Module 1: Understand Fundamental Concepts of Cryptography

Module Objectives

After completing this module, the participant will be able to:

1. Describe encryption.
2. Understand encryption terminology.
3. Use the Exclusive OR (XOR) function.
4. Describe confusion, diffusion and avalanche.
5. Discuss encryption benefits.
6. Define data classification.
7. Discuss the legal implications of encryption.
8. Discuss training requirements.

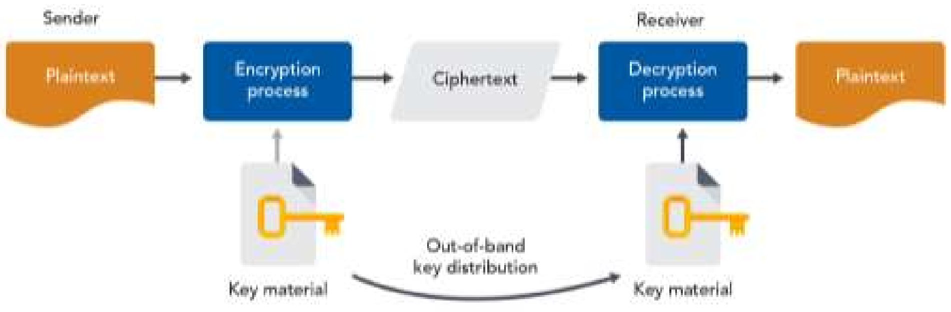
Overview

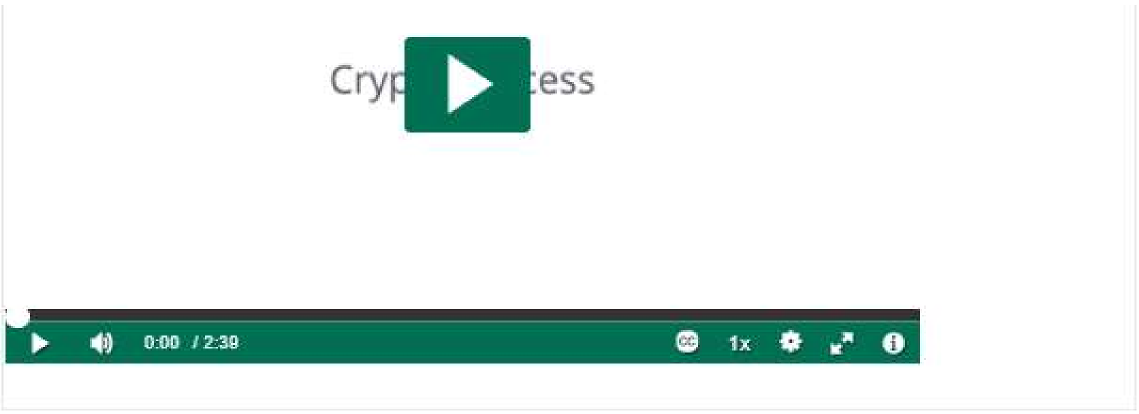
Cryptography is a fascinating field that encompasses aspects of history, mathematics, creativity, and science. Encryption has been used a tool to preserve secrets (confidentiality) for thousands of years, from merchants protecting their tangible and intellectual property to military generals communicating with their armies.

Once explained, encryption is quite simple and straightforward. This module will explain some of the basic concepts of cryptography.

Historically, cryptography was a manual process performed by the writer, but through the years it became a mechanical and electro-mechanical process using crypto devices such as the Enigma machine and the M209. Nowadays, cryptography is just a computer program built onto a chip or run from an application.

In this diagram we can examine the terminology and process of cryptography.





Cryptographic operations are based on two simple activities: substitution and transposition. Substitution is the process of replacing one value for another. An example of that process was the cipher developed by Julius Caesar. When he wrote a message, he substituted the letter he wanted with the letter three places further down the alphabet.

For example:

To encrypt the message FACE

If the natural alphabet is: A B C D E F G H . . . Z

Use an alphabet shifted 3 places: D E F G H I J K . . . C

The plaintext of the message would be: FACE

The encrypted message would be: IDFH

The secret to unlocking and reading the message would be to know the key – the value used to define the substitution alphabet. In this case, the key would be 3 – the three places used to shift the alphabet.

The other type of cryptographic operation is based on transposition, which is changing the position (order) of the letters. For example, writing the plaintext word FACE as CFEA. With transposition we still use the same letters, just in a different order.

Early encryption systems were usually based on either substitution or transposition, but today's encryption systems do multiple rounds of transposition and substitution. The Advanced Encryption Standard (AES) does ten (or more) rounds of substituting and transposing the input message to generate ciphertext.

Terminology

The greatest challenge in understanding encryption is learning the terminology. Here are the terms we will use in this course:

Plaintext/Cleartext: The message in its natural format.

Ciphertext/Cryptogram: The encrypted message that could not be understood without having the ability to decrypt the message.

Encrypt/Encode/Encipher: These terms are not identical, but for our purposes we will use them without defining the differences among them. These terms describe the process of changing plaintext to ciphertext.

Decrypt/Decode/Decipher: The process of changing ciphertext into plaintext.

Cryptosystem: The device or process that is used to encrypt or decrypt a message.

Algorithm: The mathematical process used to convert a message from plaintext to ciphertext and back again.

Key: Also known as the cryptovariable; the variable used by the algorithm during the encryption/decryption process. Typically the key is a secret password, a passphrase, or a PIN chosen either by the person or by the tool encrypting the message.

Key space: Represents the total number of possible values of keys in a cryptographic algorithm or other security measure, such as a password. For example, a 20-bit key would have a key space of 1,048,576 possible keys, or 2 to the 20th power.

Collision: Occurs when a hash function (to be explained later) generates the same output for different inputs.

Key clustering: Occurs when different encryption keys generate the same ciphertext from the same plaintext message.

Work factor: The time and effort required to break a protective measure.

Initialization vector (IV): A non-secret binary value used as the starting input algorithm for the encryption of a plaintext block sequence. An IV serves two functions: to synchronize cryptographic equipment, and to increase security by introducing additional cryptographic variance.

Confusion: Is accomplished by mixing (changing) the key values used during the repeated rounds of encryption. When the key is modified for each round, it provides added complexity that the attacker would encounter.

Diffusion: Is accomplished by mixing up the location of the plaintext throughout the ciphertext. Through transposition, the location of the first character of the plaintext may change several times during the encryption process, and this makes the cryptanalysis process much more difficult.

Avalanche effect: Cryptography feature used to design algorithms such that a minor change in either the key or the plaintext will have a significant change in the resulting ciphertext. This is also a feature of a strong-hashing algorithm.

Man-in-the-Middle (MITM): A person or entity that inserts itself into the communications channel to intercept the communications. MITM may want to "sniff" the traffic to listen in (a passive attack); or to insert new data, modify the communications, delete traffic, or replay previous messages (an active attack).

Symmetric: Encryption and decryption are carried out using the same (pre-shared) key.

Asymmetric: Two different but mathematically related keys are used, where one key is used to encrypt and another is used to decrypt. This term is most commonly used in reference to public-key infrastructure (PKI).

Synchronous: Each encryption or decryption request is performed immediately.

Asynchronous: Encryption and decryption requests are processed in queues. A key benefit of asynchronous cryptography is utilization of hardware devices and multiprocessor systems for cryptographic acceleration.

Hashing: A one-way mathematical operation that reduces a message or data file into a smaller fixed length output, or hash value. By comparing the hash value computed by the sender with the hash value computed by the receiver over the original file, unauthorized changes to the file can be detected, assuming they both used the same hash function. Ideally, there should never be more than one unique hash for a given input and one hash exclusively for a given input.

Digital Signing: Provides authentication of a sender and integrity of a sender's message. A message is input into a hash function and then is encrypted using the private key of the sender, thus yielding a digital signature. The receiver can verify the digital signature by decrypting the hash value using the signer's public key, then perform the same hash computation over the message, and then compare the hash values. If this yields an exact match, then the signature is valid.

The Properties of Exclusive-OR (XOR)

A simple XOR process is a form of additive cipher and is used in conjunction with more complex cipher systems to add increased confusion to the encryption process.
Exclusive-OR is a mathematical operation a computer processer can perform very quickly, and it lends itself to the data encryption process.

XOR operates on the principle that:

- When the input values are the same, then the output is 0, and
- When the input values are different then the output is 1.

Input Character	ASCII Equivalent	Binary Value
C	67	01000011
A	65	01000001
XOR OutPut		00000010

The XOR result would be "00000010.

If this process was being used to encrypt the message of 'C', the value of 'A' would be substituted with a double quote (binary value of 00000010) in the ciphertext.

Symmetric algorithms use the XOR operation extensively to encrypt and decrypt messages because it is a fast method of substitution. In fact, it is one of the fastest processes a computer can perform.

Benefits of Cryptography

For thousands of years, the primary benefit of cryptography has been to ensure the confidentiality of messages between two people being sent over an untrusted network . However, in the past few decades the development of new cryptographic algorithms has brought many new benefits from the use of cryptography, including:

- Confidentiality
- Access control
- Integrity
- Authentication
- Non-repudiation

Cryptography is used in protecting sensitive data in storage or when being communicated. The use of disk encryption – encryption of data stored in databases and displayed in applications – protects stored data. In addition to encryption, displayed data can also be protected by obfuscation and masking the data or using screen filters to avoid "shoulder surfing" attacks. Shoulder surfing was originally the act of obtaining sensitive data by physically peering over a person's shoulder, but today it can also be done using cameras directed at PIN pads or the screen of a user.

Cryptography is also used to protect data traveling across an untrusted network. There are many encryption products and protocols in use to protect data in transit, such as IPSec, SSL, TLS, and S/MIME . These will be examined later in the course.

Data Sensitivity and Regulatory Requirements

Some information requires special care and handling, especially when inappropriate handling of the information could result in penalties, identity theft, financial loss, invasion of privacy, or unauthorized access by an individual or many individuals. Some information is also subject to regulation by state or federal laws and requires notification in the event of a disclosure.

Data is assigned a level of sensitivity or criticality based on who should have access to it and how much harm would be done if it were disclosed or not available. This assignment of sensitivity or criticality is called data classification. An organization's data classification process must be context sensitive in many cases, where data on its own may not be sensitive, but when it is combined with other data it may become sensitive. Incidents involving data in the organization's custody should be judged on a case-by-case basis.

Encryption is a common method of protecting sensitive data. Everything from data backups to databases and application data may be encrypted to protect it.

Activity: Cryptography Requirements

INSTRUCTIONS

Answer the following questions:

1. Which type of cryptographic algorithm uses the same key to both encrypt and decrypt a message?
2. Describe the risk of key clustering.

Activity Answers: Cryptography Requirements

Answers

1. Which type of cryptographic algorithm uses the same key to both encrypt and decrypt a message?

   Symmetric.

2. Describe the risk of key clustering.

   The weakness in an algorithm where two different keys would produce the same encryption results.

Legal Issues Related to Cryptography

Cryptography has long been considered a weapon of war as well as a commercial tool. This makes it a "Dual Use Good" and in many countries the use and distribution of cryptographic algorithms has been controlled by laws that restrict the distribution of military weapons. Laws related to encryption were intended to keep encryption algorithms out of the hands of criminals or enemies and are often based on controls over the export of encryption or use of encryption under certain circumstances.

On the other hand, regulations today often require the use of encryption to protect sensitive data. Industry standards such as Payment Card Industry–Data Security Standards (PCI DSS) also require that sensitive data such as payment (credit/debit) card numbers are "rendered unreadable" when stored and that access to sensitive data be restricted according to business need-to-know.

Legislative and Regulatory Compliance

Organizations operate in environments where laws, regulations, and compliance requirements must be met. You should understand the laws and regulations of the country and industry in which you are working. An organization's governance and risk management processes must consider these requirements from an implementation and a risk perspective. These laws and regulations often offer specific actions that must be met for compliance or, in some cases, must be followed to protect an organization from penalties of laws or regulations.

Compliance stemming from legal or regulatory requirements is best addressed by ensuring that an organization's policies, procedures, standards, and guidance are consistent with any laws or regulations that may govern the organization. Furthermore, it is advisable that specific laws and their requirements are cited in an organization's governance program and information security training programs.

Privacy Requirements Compliance

Privacy laws and regulations pose confidentiality challenges. Personally identifiable information (PII) or personal Information (PI) is becoming an extremely valuable commodity for marketers, as demonstrated by the tremendous growth of social networking sites based on demography, and the targeted marketing activities that come with them. Although valuable, this information can also become a liability for an organization that runs afoul of information privacy regulations and laws.

Stay up to date on the latest developments such as those being pursued by your country's legislature with regard to processing and handling of personal information. Doing so will help ensure that the compliance activities you engage in are directed toward supporting the required laws and regulations that are in force within the geographies where your enterprise operates and that the enterprise is responsible for.

User Training

Most encryption systems in use today require very little user interaction. Most users are engaging in online banking, emails, and e-commerce transactions without even realizing that encryption is being utilized. The ease of use of cryptography has made it possible for everyone to benefit from secure communications, but it has also had the effect of leading to serious breaches of sensitive data when users fail to follow best practices in the deployment and use of encryption. The most important responsibility of users is to protect the keys they use, create strong keys and not share them with others. This is where training of users is critical to ensuring that the benefits of cryptography are realized.

Security Awareness Training Topics

Topics covered in security awareness training related to cryptography include:

- The nature of sensitive material and physical assets they may come in contact with, such as trade secrets, privacy concerns, and government classified information
- Employee and contractor responsibilities in handling sensitive information, including review of employee nondisclosure agreements
- Requirements for proper handling of sensitive material in physical form, including marking (identifying data by type, sensitivity etc.), transmission, storage, and destruction
- Proper methods for protecting sensitive information on computer systems, including password policy and use of two-factor authentication
- Other computer security concerns: malware, phishing, social engineering, and so forth
- Consequences of failure to properly protect information, including potential loss of employment, economic damage to the firm, harm to individuals whose private records are divulged, and possible civil and criminal penalties

Summary

In order to understand following modules in this chapter on cryptography, it is essential to be familiar with the terminology. The security practitioner should appreciate the uses and benefits of cryptography so that it can be implemented effectively.

Module 2: Understand Fundamental Concepts of Cryptography - Encryption Methods

Module Objectives

After Completing this module, the participant will be able to:

1. Describe the methods of cryptography.
2. Compare symmetric and asymmetric cryptography.
3. Describe the advantages and disadvantages of symmetric and asymmetric cryptography.
4. Differentiate between messages at rest and messages in transit.
5. Differentiate between stream-based and block-based encryption.
6. Describe out-of-band key distribution.
7. Explain proof of origin.

Overview

The two main types of algorithms used in encryption are symmetric and asymmetric. These types of algorithms provide substantially different benefits and have different uses as part of a cryptographic implementation. This module will examine each of these types of algorithms and describe their uses and benefits.

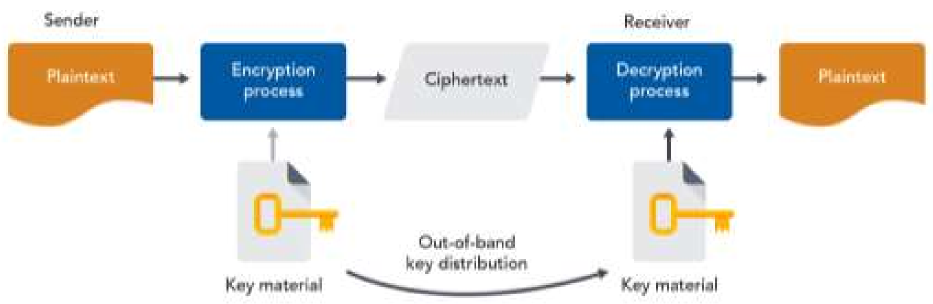
Evaluation of Cryptographic Algorithms

There are two primary types of cryptographic algorithms: symmetric and asymmetric. The security practitioner should be familiar with the strengths and weaknesses of each type of algorithm and how the algorithms can be used effectively.

Symmetric Algorithms

Symmetric algorithms are the traditional type of algorithm that has been in use for many years. The central characteristic of a symmetric algorithm is that it uses the same key in both the encryption and the decryption processes. It could be said that the decryption process is just a mirror image of the encryption process.

Symmetric algorithms work as shown in the figure below.



The same key is used for both the encryption and decryption processes. This means that the two parties communicating need to share knowledge of the same key. This type of algorithm protects data since a person who does not have the correct key would not be able to read the encrypted message.

Since the key is shared, however, this can lead to several other challenges:

- Each individual or group of people wishing to communicate would need to use a different key for each individual or group they want to talk with than the one they use to talk with other groups. This raises the challenge of scalability – the number of keys needed grows quickly as the number of different users or groups increases. The formula for the number of key is $n(n-1)/2$, where n is the number of users. Under this type of symmetric arrangement, an organization of 1000 employees would need to manage 499,500 keys.

Primary uses of symmetric algorithms:

- Encrypting bulk data (backups, hard drives, portable media)
- Encrypting messages traversing communications channels (IPSec, TLS)

Examples of symmetric algorithms:

- DES, 3DES
- AES
- Blowfish/Twofish
- MARS
- Serpent
- RC 4, 5, 6

Other names for symmetric algorithms:

- Same key
- Single key
- Shared key
- Private key
- Secret key
- Session key

Advantages and Disadvantages of Symmetric Algorithms

There are many benefits to using symmetric algorithms for encryption. The main benefit is that symmetric algorithms can provide an excellent level of confidentiality and protection of sensitive data from unauthorized disclosure. The United States Government mandates the use of the symmetric algorithm-based Advanced Encryption Standard (AES) for use in all government agencies to protect sensitive data.

Symmetric algorithms:

- Are excellent for confidentiality
- Can provide some integrity (HMAC, Keyed Hashing)
- Are often freely available – many algorithms (including AES) are not patent protected
- Are fast – they encrypt and decrypt messages relatively quickly

Symmetric algorithms have some disadvantages:

- Key management including scalability, key distribution
- Weak for authentication or non-repudiation

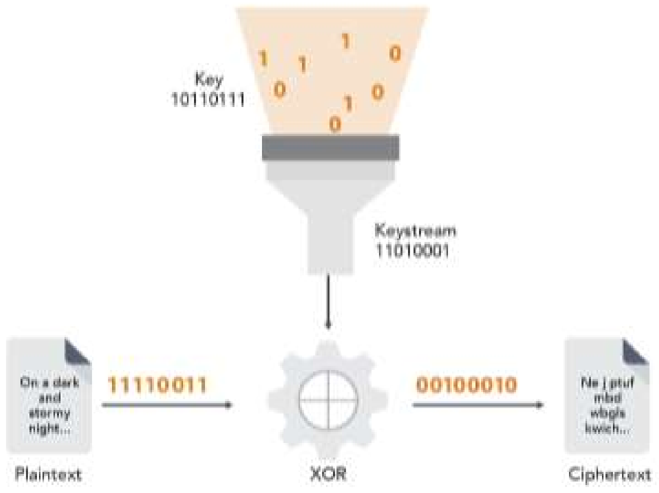
However, the situation becomes a little more complicated when we look at the types of messages that we want to protect. Some will be at-rest (on a HDD (high density drive) etc.) while others will be in transmission (emails etc.).

Does this difference really have any impact on what we are trying to accomplish? In one key area, it does. Messages in transmission are real-time and, as such, any latency in the encryption–decryption process will potentially have serious implications.

To help overcome these potential problems we have two different approaches: block-based and stream-based algorithms.

Stream-Based Algorithms

When we talk on the phone, our voice is a stream of data being passed between the two parties. If we want to encrypt that data, then we want an encryption process that operates quickly and does not break up the conversation or cause delays in processing. For this we use a stream-based algorithm that will encrypt very quickly. An example of a stream-based algorithm is the Rivest Cipher 4 (RC4), which was used as the basis for the wired equivalent privacy (WEP) and used to secure wireless devices as), WiFi Protected Access (WPA), and many other encryption implementations. Stream-based algorithms are often implemented on hardware, and they provide fast, secure encryption often unnoticed by the end users. Stream-based algorithms usually will encrypt each bit or each byte as it is transmitted using a keystream that is generated by the original key the users submitted (see the figure below).



The key (in binary format shown here as 10110111) is input into a keystream generator. The keystream generator creates a pseudo-random keystream based on the key. (It is pseudo-random since it appears to be a totally random value but it really isn't – it is based on the key, and the same keystream must be generated at the destination in order to decrypt the message.) The keystream of ones and zeros is XORed with the input plaintext one bit at a time to generate the ciphertext.

Block-Based Algorithms

The other type of symmetric algorithms are block based, meaning they will encrypt an entire block of data at a time. AES, for example, accepts input data in blocks of 128 bits in size. Then it performs multiple operations of transposition and substitution on that block of data as it converts it into ciphertext.

Often it would be fair to say that block-based encryption is more robust that stream-based, but we still have the latency (the delay in processing) issue to deal with. So, with block encryption we further divide its function into two operational groupings according to the type of block mode cipher used.

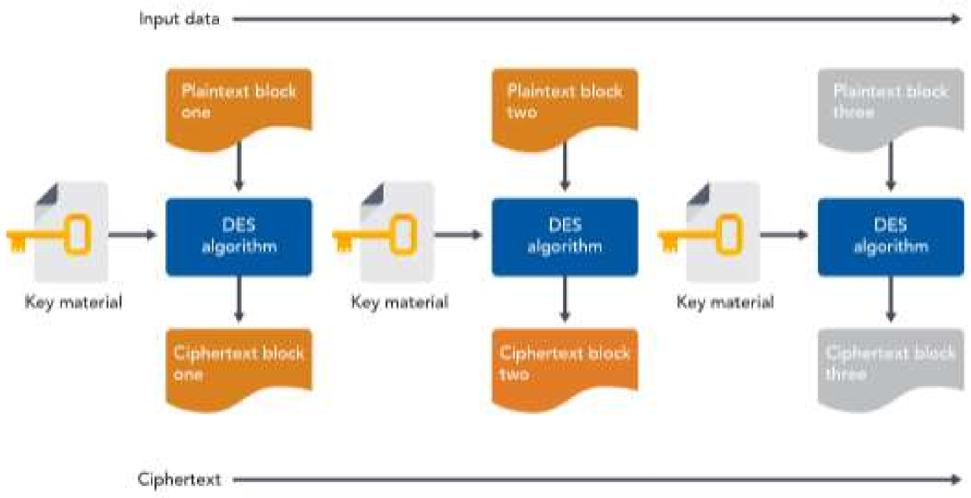
Messages where latency isn't an issue are encrypted using a block mode cipher in one of the two traditional block modes: ECB and CBC. And messages where latency is an issue are encrypted using a block mode cipher in operations that work the same way as a stream-based cipher by encrypting one byte at a time. The stream-mode implementations of block mode ciphers are: CFB, OFB and CTR.

So now we see that a block mode algorithm can operate in multiple modes either as a block operation, or it can simulate a stream-based operation.

Electronic Code Book (ECB) and Cipher Block Chaining (CBC)

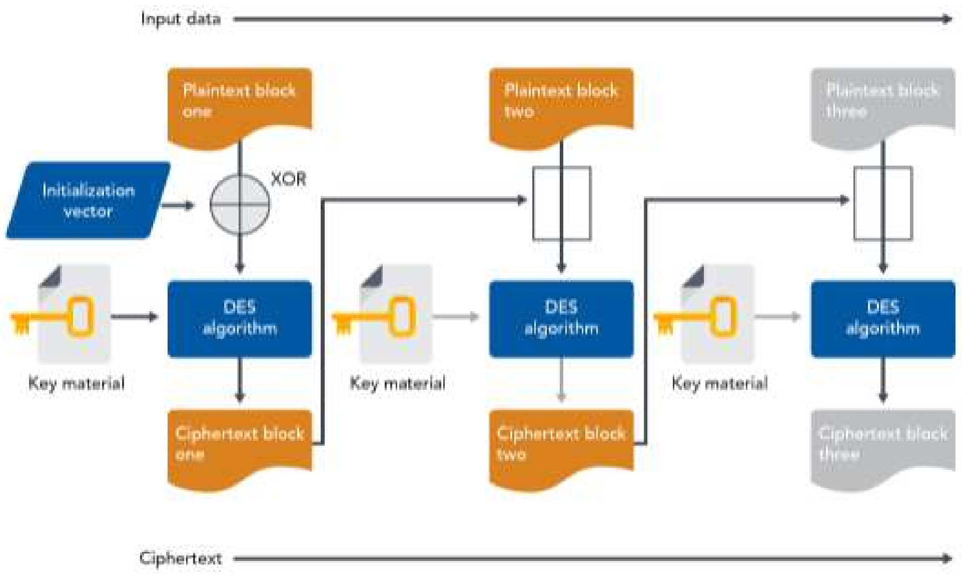
The two primary block modes are Electronic Code Book (ECB) and Cipher Block Chaining (CBC).

- **Electronic Code Book (ECB) Mode:** The ECB is the most basic block cipher mode. It is called codebook because it is similar to having a large codebook containing every piece of 64-bit plaintext input and all possible 64-bit ciphertext outputs. In a manual sense, it would be the same as looking up the input in a book and finding what the output would be depending on which key was used. When a plaintext input is received by ECB, it operates on that block independently and produces the ciphertext output. If the input was more than 64 bits long and each 64-bit block was the same, then the output blocks would also be the same. Such regularity would make cryptanalysis simple. For that reason, ECB is only used for very short messages (less than 64 bits in length), such as transmission of a key. As with all Feistel ciphers, the decryption process is the reverse of the encryption process.



CBC is the more common mode of implementation for a symmetric block mode algorithm.

- **Cipher Block Chaining (CBC) Mode:** The CBC mode is stronger than ECB in that each input block will produce a different output – even if the input blocks are identical. This is accomplished by introducing two new factors into the encryption process – an IV and a chaining function that XORs each input with the previous ciphertext. (Note: Without the IV, the chaining process applied to the same messages would create the same ciphertext.) The IV is a randomly chosen value that is mixed with the first block of plaintext. This acts just like a seed in a stream-based cipher. The sender and the receiver must know the IV so that the message can be decrypted later.



The initial input block is XORed with the IV, and the result of that process is encrypted to produce the first block of ciphertext. This first ciphertext block is then XORed with the next input plaintext block. This is the chaining process, which ensures that even if the input blocks are the same, the resulting outputs will be different.

The three modes of operation where symmetric block algorithms act as if they were a stream-based algorithm include CFB (Cipher Feedback, OFB, (Output Feedback and CTR (Counter mode):

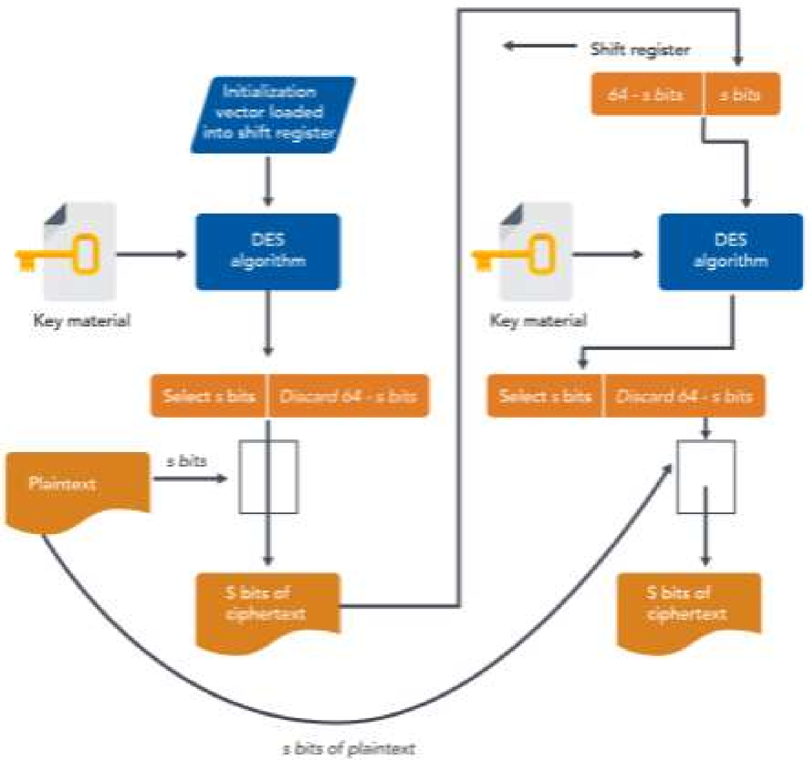
The Stream Modes of DES, AES and other block mode symmetric ciphers

Data Encryption Standard (DES) can operate as a stream cipher; even though it is actually a block mode cipher, by using one of these three modes. A block-based cipher is subject to the problems of latency or delay in processing. This makes them unsuitable for many applications where simultaneous transmission of the data is desired. In a stream mode, DES tries to simulate a stream to be more versatile and provide support for stream-based applications.

Cipher Feedback (CFB)

In Cipher Feedback (CFB) mode, the input is separated into individual segments, the size of which can be 1-bit, 8-bit, 64-bit, or 128-bit (the four sub-modes of CFB)* – usually of 8 bits, because that is the size of one character. When the encryption process starts, the IV is chosen and loaded into a shift register. It is then run through the encryption algorithm. The first 8 bits that come from the algorithm are then XORed with the first 8 bits of the plaintext (the first segment). Each 8-bit segment is then transmitted to the receiver and also fed back into the shift register. The shift register contents are then encrypted again to generate the keystream to be XORed with the next plaintext segment.

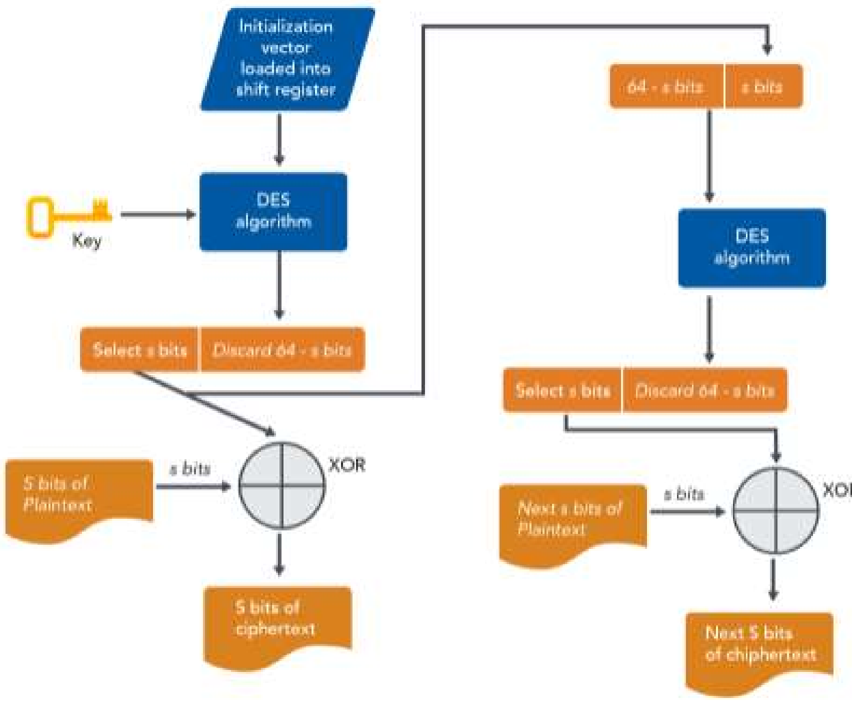
This process continues until the end of the input. One of the drawbacks of this, however, is that if a bit is corrupted or altered, all of the data from that point onward will be damaged. It is interesting to note that because of the nature of the operation in CFB, the decryption process uses the encryption operation rather than operating in reverse like CBC.



CFB mode operates by encrypting an IV (random value) with the chosen symmetric key. The output of this encryption is fed into a shift register. The first 's' bits of the plaintext are XORed with the first 's' bits of the value in the shift register. This generates 's' bits of ciphertext. The ciphertext is sent to the destination but also used to reload the shift register by encrypting the ciphertext with the key. This creates the value in the shift register to encrypt the next 's' bits of the plaintext. This process continues until all the plaintext has been encrypted – 's' bits at a time.

Output Feedback (OFB)

Output Feedback (OFB) mode is very similar in operation to the CFB, except that instead of using the ciphertext result of the XOR operation to feed back into the shift register for the ongoing keystream, it feeds the encrypted keystream itself back into the shift register to create the next portion of the keystream.

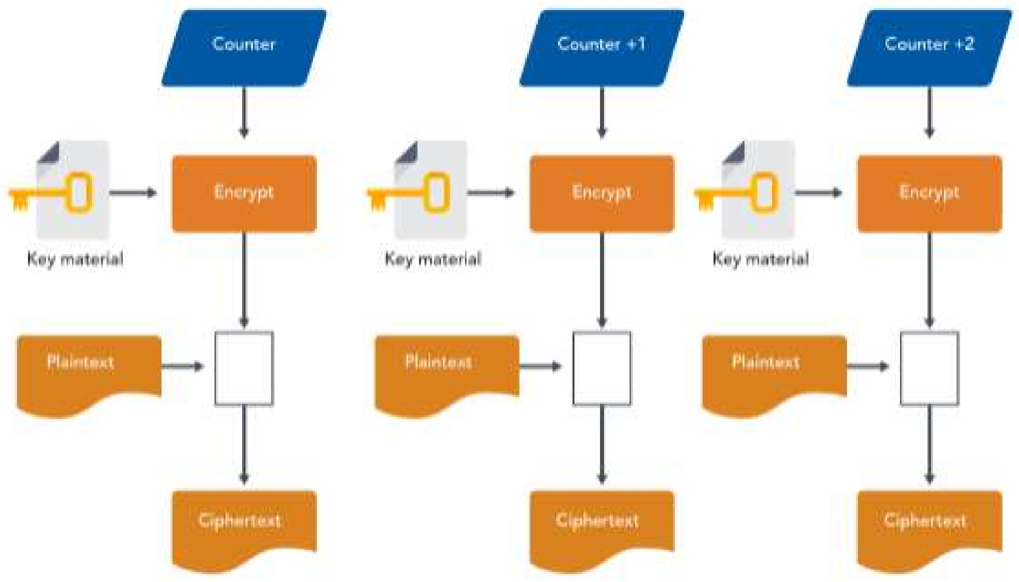


Because the keystream and message data are completely independent (the keystream itself is chained, but there is no chaining of the ciphertext), it is now possible to generate the entire keystream in advance and store it for later use. However, this does pose some storage complications, especially if it were to be used in a high-speed link.

Counter (CTR)

The Counter (CTR) mode is used in high-speed applications such as IPSec and ATM. In this mode, a counter – a 64-bit random data block – is used as the first IV. A requirement of CTR is that the counter must be different for every block of plaintext, so for each subsequent block, the counter is incremented by 1. The counter is then encrypted just as in OFB, and the result is used as a keystream and XORed with the plaintext.

Because the keystream is independent from the message, it is possible to process several blocks of data at the same time, thus speeding up the throughput of the algorithm. Again, because of the characteristics of the algorithm, the encryption process is used at both ends of the process – there is no need to install the decryption process.



The Data Encryption Standard (DES)

A Data Encryption Standard (DES) key is 64 bits in length; however, every eighth bit (used for parity) is ignored. Therefore, the effective length of the DES key is 56 bits. Because every bit has a possible value of either 1 or 0, it can be stated that the effective key space for the DES key is 256. This yields a total number of keys for DES as 7.2 × 1016. However, the modes of operation discussed next are used by a variety of other block ciphers, not just in DES. Originally, there were four modes of DES accepted for use by the U.S. federal government's National Institute of Standards and Technology (NIST); in later years, the CTR (Counter) mode was also accepted.

Advantages and Disadvantages of DES

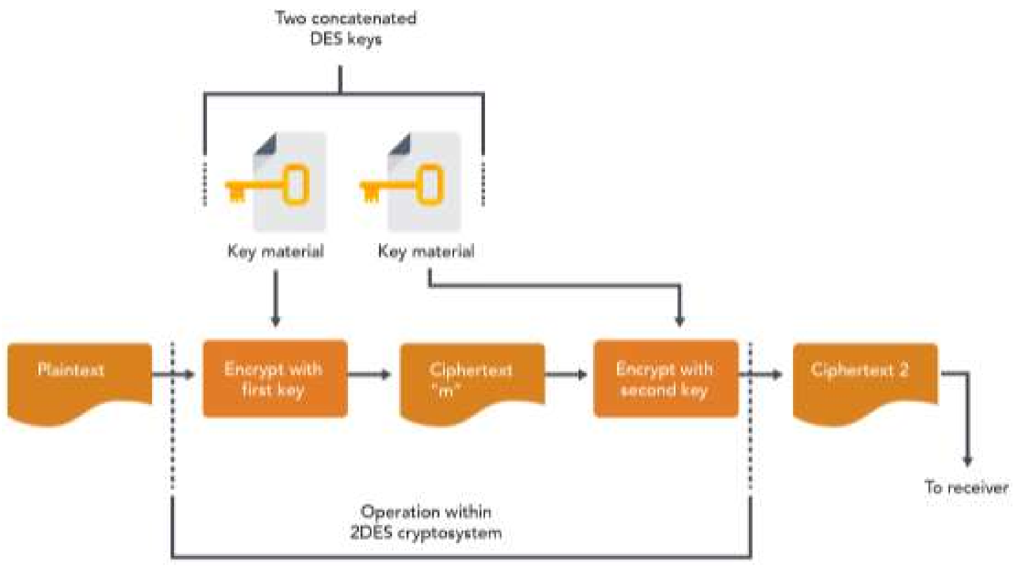
DES is a strong, fast algorithm that has endured the test of time; however, it is not suitable for use for very confidential data due to the increase in computing power over the years.

Initially, DES was considered unbreakable, and early attempts to break a DES message were unrealistic. (A computer running at one attempt per millisecond would still take more than 1,000 years to try all possible keys.) However, DES is susceptible to a brute-force attack. Because the key is only 56 bits long, the key may be determined by trying all possible keys against the ciphertext until the true plaintext is recovered. The Electronic Frontier Foundation demonstrated this several years ago. However, it should be noted that they did the simplest form of attack – a known plaintext attack; they tried all possible keys against a ciphertext knowing what they were looking for (they knew the plaintext). If they had not known the plaintext (had not known what they were looking for), the attack would have been significantly more difficult.

Regardless, DES can be deciphered using today's computing power and enough stubborn persistence. There have also been criticisms of the structure of the DES algorithm. The design of the S-boxes used in the encryption and decryption operations was secret, and this can lead to claims that they may contain hidden code or untried operations.

Double DES

The primary complaint about DES was that the key was too short. This made a known plaintext brute-force attack possible. One of the first alternatives considered to create a stronger version of DES was to double the encryption process, hence double DES (2DES). The first DES operation created an intermediate ciphertext, which will be referred to as "m" for discussion purposes.



This intermediate ciphertext, m, was then re-encrypted using a second 56-bit DES key for greater cryptographic strength. Initially, there was a lot of discussion as to whether the ciphertext created by the second DES operation would be the same as the ciphertext that would have been created by a third DES key.

In other words, double DES looks like this:

C = EK2(EK1(P))

Ciphertext (C) = the Plaintext (P) encrypted

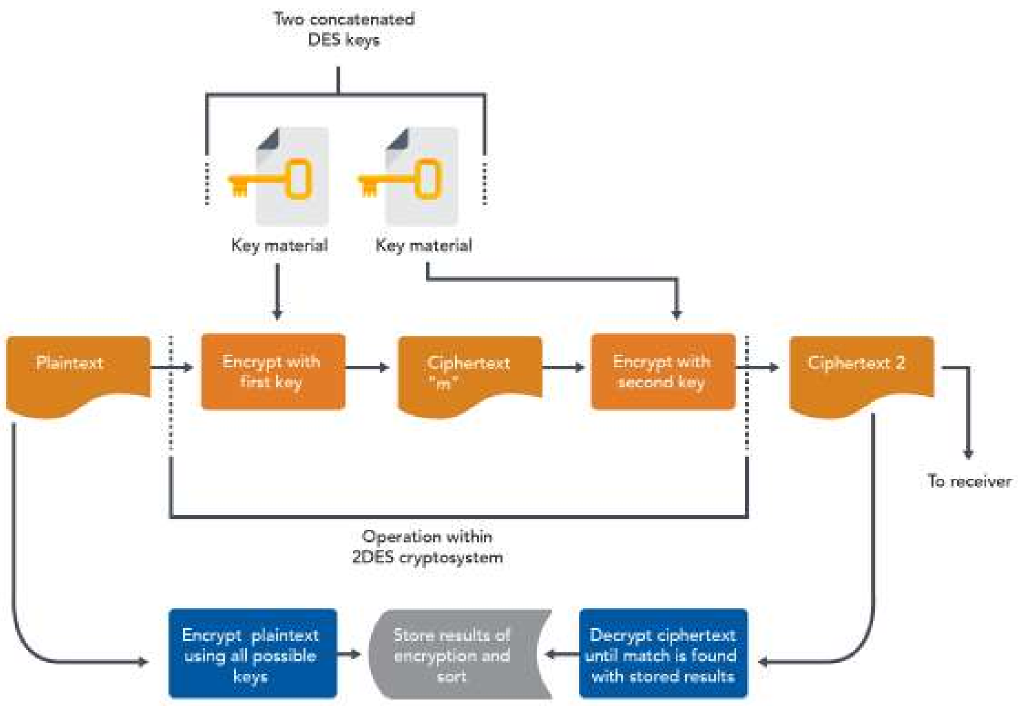
with Key 1 (EK1) and then re-encrypted

with Key 2 (EK2)

Ciphertext created by 2DES is the result of the plaintext encrypted with the first 56-bit DES key and then re-encrypted with the second 56-bit DES key. Double DES therefore is the result of single encryption operation done twice using different keys – it is not the same as encrypting with a 112-bit key. Unfortunately, this makes double DES vulnerable to a "meet-in-the-middle" attack, where an attacker could attack the operation from both sides. It also explains why the lifespan of 2DES was very short.

Meet-in-the-Middle

The most effective attack against 2DES was just like the successful attacks on single DES, based on doing a brute-force attack against known plaintext. The attacker would encrypt the plaintext using all possible keys and create a table containing all possible results. This intermediate cipher is referred to as "m" for this discussion. This would mean encrypting using all 256 possible keys. The table would then be sorted according to the values of m. The attacker would then decrypt the ciphertext using all possible keys until he found a match with the value of m. This would result in a true strength of double DES of approximately 256 (twice the strength of DES, but not strong enough to be considered effective), instead of the 2112 originally hoped for.


Two concatenated DES keys
Key material
Key material
Plaintext
Encrypt with first key
Ciphertext "m"
Encrypt with second key
Ciphertext 2
To receiver
Operation within 2DES cryptosystem
Encrypt plaintext using all possible keys
Store results of encryption and sort
Decrypt ciphertext until match is found with stored results

Triple DES (3DES)

The defeat of 2DES resulted in the adoption of triple DES (3DES) as the next solution to overcome the weaknesses of single DES. Triple DES was designed to operate at a relative strength of 2112 using two different keys to perform the encryption.

The 3DES operation using two keys is shown below:

C = EK1(EK2(EK1(P)))

The ciphertext is created by encrypting the plaintext with key 1, re-encrypting with key 2, and then encrypting again with key 1.

This would have a relative strength of 2112 and be unfeasible for attack using either the known plaintext or differential cryptanalysis attacks. This mode of 3DES would be referred to as EEE2 (encrypt, encrypt, encrypt using two different keys).

The preferred method of using triple DES was to use a decrypt step for the intermediate operation, as shown below:

C = EK1(DK2(EK1(P)))

The plaintext was encrypted using key 1, then decrypted using key 2, and then encrypted using key 1.

Doing the decrypt operation for the intermediate step does not make a difference in the strength of the cryptographic operation, but it does allow backward compatibility through permitting a user of triple DES to also access files encrypted with single DES. This mode of triple DES is referred to as EDE2. Originally, the use of triple DES was primarily done using two keys as shown above, and this was compliant with ISO 8732 and ANS X9.17; however, some users, such as PGP and Secure/Multipurpose Internet Mail Extension (S/ MIME), are moving toward the adoption of triple DES using three separate keys.

This would be shown as follows:

C = EK3(EK2(EK1(P))) for the EEE3 mode

or

C = EK3(DK2(EK1(P))) for the EDE3 mode

There are seven different modes of 3DES, but the four explained above are the most common and of the most concern to the security professional.

Advanced Encryption Standard (AES)

In 1997, the National Institute of Standards and Technology (NIST) in the United States issued a call for a product to replace DES and 3DES. The requirements were that the new algorithm must be at least as strong as DES, have a larger block size (because a larger block size would be more efficient and more secure), and overcome the problems of performance with DES. DES was developed for hardware implementations and is too slow in software. 3DES is even slower, and thus creates a serious latency in encryption as well as significant processing overhead.

After considerable research, the product chosen to be the new Advanced Encryption Standard (AES) was the Rijndael algorithm, created by Dr. Joan Daemon and Dr. Vincent Rijmen of Belgium. (The name Rijndael was merely a contraction of their surnames.) Rijndael beat out the other finalists: Serpent, of which Ross Anderson was an author; MARS, an IBM product; RC6, from Ron Rivest and RSA; and Twofish, developed by Bruce Schneier. The AES algorithm was obliged to meet many criteria, including the need to be flexible, implementable on many types of platforms, and free of royalties. AES is only one implementation of Rijndael as Rijndael can be implemented in other modes besides the ones specified for AES.

Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP)

When people use WPA2 (WiFi Protected Access 2) for secure wireless communications, they are using Advanced Encryption Standard – Counter Cipher Block Chaining Message Authentication Code Protocol (AES-CCMP), which is a combination of the block mode operation of CBC (to prove message integrity) and the stream mode operation of CTR (to provide message confidentiality). CCMP is defined in the IETF RFC 3610 and is included as a component of the 802.11i IEEE standard.

How CCMP Works

AES processing in CCMP must use AES 128-bit key and 128-bit block size. According to the United States' Federal Information Processing Standard (FIPS) 197, the AES algorithm (a block cipher) uses blocks of 128 bits, cipher keys with lengths of 128, 192, and 256 bits, as well as a number of rounds, 10, 12, and 14, respectively. CCMP use of 128-bit keys and a 48-bit IV minimizes vulnerability to replay attacks. The CTR component provides data privacy.

The CCMP (Cipher Block Chaining (CBC) message authentication code (MAC)) component produces a message integrity code (MIC) that provides data origin authentication and data integrity for the packet payload data.

The 802.11i standard includes CCMP. AES is often referred to as the encryption protocol used by 802.11i; however, AES itself is simply a block cipher. The actual encryption protocol is CCMP. It is important to note here that, although the 802.11i standard allows for TKIP encryption, Robust Security Network (RSN) is part of the 802.11i IEEE standard and negotiates authentication and encryption algorithms between access points and wireless clients.

This flexibility allows new algorithms to be added at any time and supported alongside previous algorithms. The use of AES-CCMP is mandated for RSNs. AES-CCMP introduces a higher level of security from past protocols by providing protection for the MAC protocol data unit (MPDU) and parts of the 802.11 MAC headers. This protects even more of the data packet from eavesdropping and tampering.

Rijndael

The Rijndael algorithm can be used with block sizes of 128, 192, or 256 bits. The key can also be 128, 192, or 256 bits, with a variable number of rounds of operation depending on the key size. Using AES with a 128-bit key would do 10 rounds, whereas a 192-bit key would do 12 and a 256-bit key would do 14. Although Rijndael supports multiple block sizes, AES only supports one block size (subset of Rijndael).

International Data Encryption Algorithm (IDEA)

IDEA was developed as a replacement for DES by Xuejai Lai and James Massey in 1991. IDEA uses a 128-bit key and operates on 64-bit blocks. IDEA does eight rounds of transposition and substitution using modular addition and multiplication, and bitwise exclusive-or (XOR).

CAST

CAST was developed in 1996 by Carlisle Adams and Stafford Tavares. CAST-128 can use keys between 40 and 128 bits in length and will do between 12 and 16 rounds of operation, depending on key length. CAST-128 is a Feistal-type block cipher with 64-bit blocks.

CAST-256 was submitted as an unsuccessful candidate for the new AES. CAST-256 operates on 128-bit blocks and with keys of 128, 192, 160, 224, and 256 bits. It performs 48 rounds and is described in RFC 2612.

Secure and Fast Encryption Routine (SAFER)

All of the algorithms in SAFER are patent-free. The algorithms were developed by James Massey and work on either 64-bit input blocks (SAFER-SK64) or 128- bit blocks (SAFER-SK128). A variation of SAFER is used as a block cipher in Bluetooth.

Blowfish

Blowfish is a symmetrical algorithm developed by Bruce Schneier. It is an extremely fast cipher and can be implemented in as little as 5K of memory. It is a Feistal-type cipher in that it divides the input blocks into two halves and then uses them in XORs against each other. However, it varies from the traditional Feistal cipher in that Blowfish does work against both halves, not just one.

The Blowfish algorithm operates with variable key sizes, from 32 up to 448 bits on 64-bit input and output blocks. One of the characteristics of Blowfish is that the S-boxes are created from the key and are stored for later use. Because of the processing time taken to change keys and recompute the S-boxes, Blowfish is unsuitable for applications where the key is changed frequently or in applications on smart cards or with limited processing power. Blowfish is currently considered unbreakable (using today's technology), and in fact, because the key is used to generate the S-boxes, it takes over 500 rounds of the Blowfish algorithm to test any single key.

Twofish

Twofish was one of the finalists for the AES. It is an adapted version of Blowfish developed by a team of cryptographers led by Bruce Schneier. It can operate with keys of 128, 192, or 256 bits on blocks of 128 bits. It performs 16 rounds during the encryption/decryption process.

RC5

RC5 was developed by Ron Rivest of RSA and is deployed in many of RSA's products. It is a very adaptable product useful for many applications, ranging from software to hardware implementations. The key for RC5 can vary from 0 to 2040 bits, the number of rounds it executes can be adjusted from 0 to 255, and the length of the input words can also be chosen from 16-, 32-, and 64-bit lengths. The algorithm operates on two words at a time in a fast and secure manner.

RC5 is defined in RFC 2040 for four different modes of operation:

1. RC5 block cipher is similar to DES ECB, producing a ciphertext block of the same length as the input.
2. RC5-CBC is a cipher block chaining form of RC5 using chaining to ensure that repeated input blocks would not generate the same output.
3. RC5-CBC-Pad combines chaining with the ability to handle input plaintext of any length. The ciphertext will be longer than the plaintext by at most one block.
4. RC5-CTS is called ciphertext stealing and will generate a ciphertext equal in length to a plaintext of any length.

RC4

RC4, a stream-based cipher, was developed in 1987 by Ron Rivest for RSA Data Security and has become the most widely used stream cipher, being deployed, for example, in WEP. RC4 uses a variable-length key ranging from 8 to 2048 bits (1 to 256 bytes) and a period of greater than 10100. In other words, the keystream should not repeat for at least that length.

Asymmetric Algorithms

Symmetric encryption has been in use for many thousands of years, but asymmetric encryption is relatively new, becoming publicly available only in the 1970s.

Asymmetric algorithms became commonly known when Drs. Whitfield Diffie and Martin Hellman released a paper in 1976 called "New Directions in Cryptography." The Diffie–Hellman paper described the concept of using two different keys (a key pair) to perform the cryptographic operations – the essence of asymmetric cryptography.

The pair of keys used in asymmetric cryptography are mathematically related and must always be used as a pair. One key will not work without the other key also being used. The key pair consists of a private key, which the owner of the key pair MUST keep private; and a public key, which is computed from the private key and can be shared with anyone the owner wishes to share it with. The mathematics used in creating the key pair makes it simple to calculate the value of the public key if a person knows the value of the private key, but the reverse (i.e., to determine the value of the private key based on the value of the public key) is computationally infeasible.

Public key cryptography uses what is known as a trapdoor function, meaning that while it may be easy to compute a value in one direction, reversing the process is extremely difficult to impossible. Starting with A to compute B is easy, but given value B trying to calculate the start value of A is not possible.

A user wishing to communicate using an asymmetric algorithm would first generate a key pair. To ensure the strength of the key generation process, this is usually done by the cryptographic application or the PKI implementation without user involvement. One half of the key pair is kept secret; only the key holder knows that key. This is why it is called the private key. The other half of the key pair can be given freely to anyone who wants a copy. In many companies, it may be available through the corporate website or access to a key server. This is why this second half of the key pair is referred to as the public key.

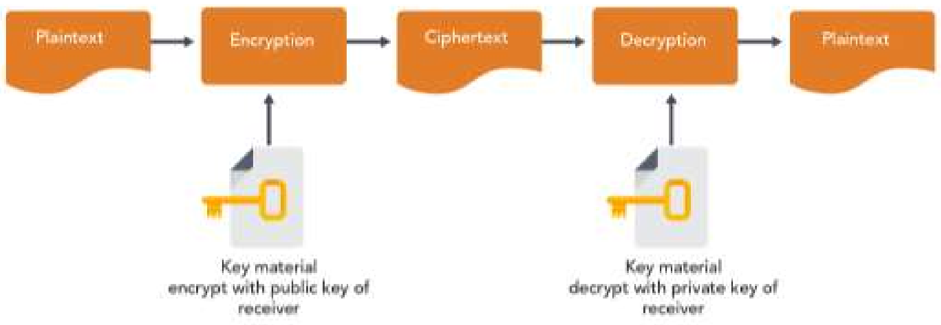
> Asymmetric cryptography is commonly called public-key cryptography (PKC) because it uses public keys as well as private keys, and it is often implemented as a public-key infrastructure (PKI) as will be explained in Module 3 of this chapter.

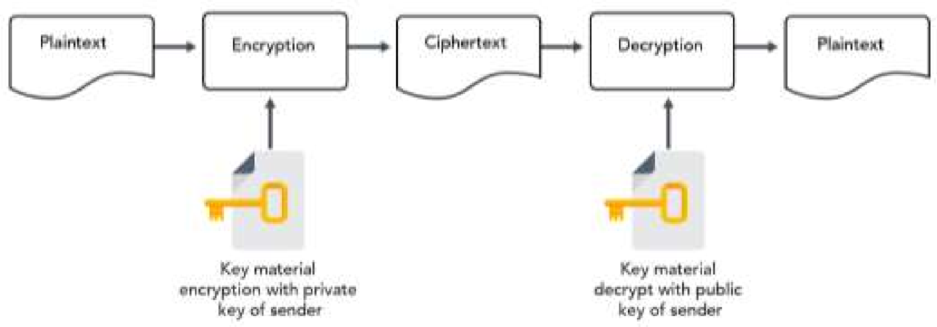
Use of Public-Key Algorithms
Confidentiality

The two keys (private and public) are a key pair; they must be used together. This means that any message that is encrypted with a public key can only be decrypted with the corresponding other half of the key pair, the private key. Therefore, as long as the key holder keeps the private key secure, there exists a method of transmitting a message confidentially. The sender would encrypt the message with the public key of the receiver. Only the receiver with the private key would be able to open or read the message, providing confidentiality.

In the below figure we can see the theory of how public key encryption can be used to send a confidential message across an untrusted channel.

Using Public Key Cryptography to Send a Confidential Message.




Plaintext
Encryption
Ciphertext
Decryption
Plaintext
Key material encryption with private key of sender
Key material decrypt with public key of sender

Open Message or Proof of Origin

Conversely, when a message is encrypted with the private key of a sender, it can be opened or read by anyone who possesses the corresponding public key. Anyone who needs to send a message and provide proof of origin (non-repudiation) and authentication of the sender can do so by encrypting the message with the sender's own private key. When recipients receive the message, they must decrypt the message with the sender's public key since that is the only key that will decrypt a message that was encrypted using the sender's private key. Recipients then have some guarantee that, because they opened it with the public key from the sender, the message did in fact originate with the sender.

Examples of public-key algorithms include:

- Diffie-Hellman
- RSA
- ECC

Diffie-Helman Algorithm

The Diffie-Hellman algorithm is used differently than RSA or ECC. Whereas RSA and ECC are often used to encrypt and transmit a symmetric key to the recipient, D-H is used instead to negotiate the value of the symmetric key between the sender and receiver. Instead of the sender of a message choosing the symmetric key to be used, as is done with the other asymmetric algorithms, with D-H each participant exchanges their public key that is then used to negotiate the value of the symmetric key. This is a common practice with an implementation of IPSec.

Diffie-Hellman is a mathematical function based first on finding the primitive root of a prime number.

El Gamal

The El Gamal cryptographic algorithm is based on the work of Diffie and Hellman. It is based on the same mathematical functions of discrete logs. However, it includes the ability to provide message confidentiality and digital signature services, not just session key exchange.

RSA

Despite NISTs recommendation (2010), RSA is still the most widely used public key algorithm and operates on blocks of text. It was developed in 1978 by Ron Rivest, Adi Shamir, and Len Adleman when they were at MIT. RSA is based on the mathematical challenge of factoring the product of two large prime numbers.

A prime number, by definition, can be divided only by 1 and itself. Some prime numbers include 2, 3, 5, 7, 11, 13, and so on. Factoring is defined as taking a number and finding the numbers that can be multiplied together to calculate that number. For example, if the product of a*b = c, then c can be factored into a and b. As 3*4 = 12, then 12 can be factored into 3, 4 and 6, 2 and 12, 1.

The RSA algorithm uses large prime numbers that when multiplied together would be incredibly difficult to factor. Successful factoring attacks have been executed against 512-bit numbers (at a cost of approximately 8000 MIPS years), and successful attacks against 1024-bit numbers appear increasingly possible in the near term. The U.S. government organization NIST recommended moving away from 1024-bit RSA key size by the end of 2010. The recommendation stated in part:

"If information is initially signed in 2009 and needs to remain secure for a maximum of 10 years (i.e., from 2009 to 2019), a 1024-bit RSA key would not provide sufficient protection between 2011 and 2019 and, therefore, it is not recommended that 1024-bit RSA be used in this case."

Elliptic Curve Cryptography (ECC)

One branch of discrete logarithmic algorithms is based on the algebraic structure of elliptic curves over finite fields. Developed in the mid-1980s, this type of cryptography was largely ignored until recently.

The elliptic curve algorithms have the highest strength per bit of key length of any of the asymmetric algorithms. The ability to use much shorter keys for ECC implementations provides savings on computational power and bandwidth. This makes ECC especially beneficial for implementation in smart cards, wireless, and other similar application areas.

The below table compares ECC and RSA key sizes. You will notice that to achieve high levels of protection ECC keys are substantially smaller.

ECC	RSA
256-Bit key	3072-Bit key
384-Bit key (NSA acceptable for top secret)	7680-Bit key

Advantages and Disadvantages of Asymmetric-Key Algorithms

The development of asymmetric cryptography revolutionized the cryptographic community. Now it was possible to send a message across an untrusted medium in a secure manner without the overhead of prior key exchange or key material distribution. It allowed several other features not readily available in symmetric cryptography, such as the non-repudiation of origin, access control, data integrity, and the non-repudiation of delivery. But it is extremely slow because it is many times more computationally intensive than symmetric algorithms.

Benefits of Asymmetric Cryptography

Asymmetric cryptography provides at least five primary benefits:

1. Confidentiality
2. Access Control
3. Integrity
4. Authentication
5. Non-repudiation

Problem with Asymmetric Cryptography

The use of asymmetric cryptography is based on mathematical calculations of large numbers. This makes asymmetric encryption very slow – so slow that it would not work for most communications.

Given that the message being transmitted could be large, the time involved to encrypt and decrypt might make the use of asymmetric encryption impractical. These is where the use of hybrid cryptography becomes useful.

Hybrid Cryptography

The solutions to many of the problems with symmetric encryption lie in developing a hybrid technique of cryptography that combined the strengths of both symmetric (i.e. great speed and secure algorithms) and asymmetric cryptography (i.e. secure exchange of session keys, message authentication, and non-repudiation).

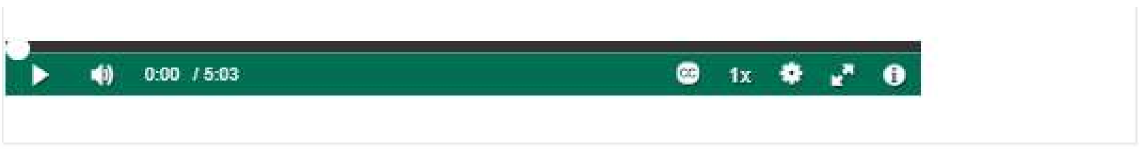
Symmetric cryptography is best for encrypting large files. It can handle the encryption and decryption processes with little impact on delivery times or computational performance. Asymmetric cryptography can handle the initial setup of the communications session through the exchange or negotiation of the symmetric keys to be used for this session. In many cases, the symmetric key is only needed for the length of this communication and can be discarded following the completion of the transaction, so the symmetric key in this case will be referred to as a session key.

Hybrid Encryption Implementations

As described above, we have two types of encryption algorithms: symmetric algorithms that provide fast and confidential transmission of data, but have a disadvantage of difficult key distribution and scalability; and asymmetric algorithms that can transmit a message confidentially, but are really slow. The solution is to use the strengths of each type of algorithm to support the other type. This requires the use of symmetric algorithms to encrypt a message quickly and the use of asymmetric algorithms to send the symmetric key to the recipient that is needed to decrypt the secure message at the far end of the communications channel. We can see how a hybrid system works in the below figure.

Hybr ems
0:00 / 3:38
1x

(ISC)²
SSCP
Systems Security
Certified Practitioner
0:00 / 5:03
1x



Session Keys

Symmetric algorithms are excellent for sending confidential communications over untrusted networks, but they are weak in other areas such as key management, non-repudiation and authentication. Therefore: symmetric algorithms are often used in combination with asymmetric algorithms as a hybrid implementation of both types of algorithms. In a hybrid implementation, the symmetric key used in encrypting the confidential message is generated and used for a single communications session and then it is discarded. For this reason, the symmetric keys used for that single use are called session keys.

Summary

The security practitioner should understand the concepts, terminology, strengths, and weaknesses of cryptographic algorithms as just described. There is much more fascinating knowledge that can be examined in cryptography, but the level of knowledge described here is important to ensure that cryptography can be implemented effectively within the organization.

Day 3

Module Objectives

After completing this module, the participant will be able to:

1. Describe a public-key infrastructure (PKI) and its components.
2. Explain the fields within a certificate.
3. Describe a web of trust.
4. Apply integrity checking.
5. Distinguish between hashing algorithms.
6. Describe the birthday paradox.
7. Explain the benefit of salting.
8. Differentiate between the mandatory access control (MAC) and the hash message authentication code (HMAC).

Overview

A PKI is a set of system, software, and communication protocols required to use, manage, and control public key cryptography. It has three primary purposes:

- Publish public keys/certificates
- Certify that a key is tied to an individual or entity
- Provide verification of the validity of a public key

Integrity

An important part of secure communications is integrity. People sending data across a network need to be sure that the data has not been altered in transit. The alteration of data may be accidental as the result of noise or a loss of network connectivity, or it may be intentional where a hacker is attempting to modify the data and disrupt communications. Numerous ways to prove the integrity of communications have been developed over the years, including:

- Parity bits
- Checksums
- Cyclic Redundancy Checks (CRC)
- Message authentication codes (MAC and HMAC)
- Hashing