CCNA Cyber Ops (Version 1.1) – Chapter 9: Cryptography and the Public Key Infrastructure

itexamanswers.net/ccna-cyber-ops-version-1-1-chapter-9-cryptography-and-the-public-key-infrastructure.html

June 16, 2019

Contents

Objectives

- Upon completion of this chapter, you will be able to answer the following questions:
- How is cryptography used to secure communications?
- What is the role of cryptography in ensuring the integrity and authenticity of data?
- How do cryptographic approaches enhance data confidentiality?
- What is public key cryptography?
- How does the Public Key Infrastructure function?
- How does the use of cryptography affect cybersecurity operations?

Key Terms

This chapter uses the following key terms. You can find the definitions in the Glossary.

cryptology
cryptography
cryptanalysis
cipher
hashes
Message Digest 5 (MD5)
Secure Hash Algorithm 1 (SHA-1)
Secure Hash Algorithm 2 (SHA-2)
hash message authentication code (HMAC)
symmetric encryption
asymmetric encryption
block ciphers
stream ciphers
Data Encryption Standard (DES)
3DES (Triple DES)
Advanced Encryption Standard (AES)
Software-Optimized Encryption Algorithm (SEAL)
Rivest ciphers (RC)
Diffie-Helman (DH)
Digital Signature Standard (DSS)

Digital Signature Algorithm (DSA)
RSA
EIGamal
Elliptical curve
Public Key Infrastructure (PKI)

Introduction (9.0)

When Internet standards were first drafted, no one was thinking that data would need to be protected from threat actors. As you have seen in previous chapters,the protocols of the TCP/IP protocol suite are vulnerable to a variety of attacks.

To address these vulnerabilities, we use a variety of cryptographic technologies to keep our data private and secure. However, cryptography is a double-edged sword in that threat actors can also use it to hide their actions. This chapter covers the impact of cryptography on network security monitoring.

Class Activity 9.0.1.2: Creating Codes
Secret codes have been used for thousands of years. Ancient Greeks and Spartans used a scytale (rhymes with Italy) to encode messages. Romans used a Caesar cipher to encrypt messages. A few hundred years ago, the French used the Vigenère cipher to encode messages. Today, there are many ways that messages can be encoded.

In this lab, you will create and encrypt messages using online tools.

Cryptography (9.1)

In this section, you will learn how tools are used to encrypt and decrypt data.

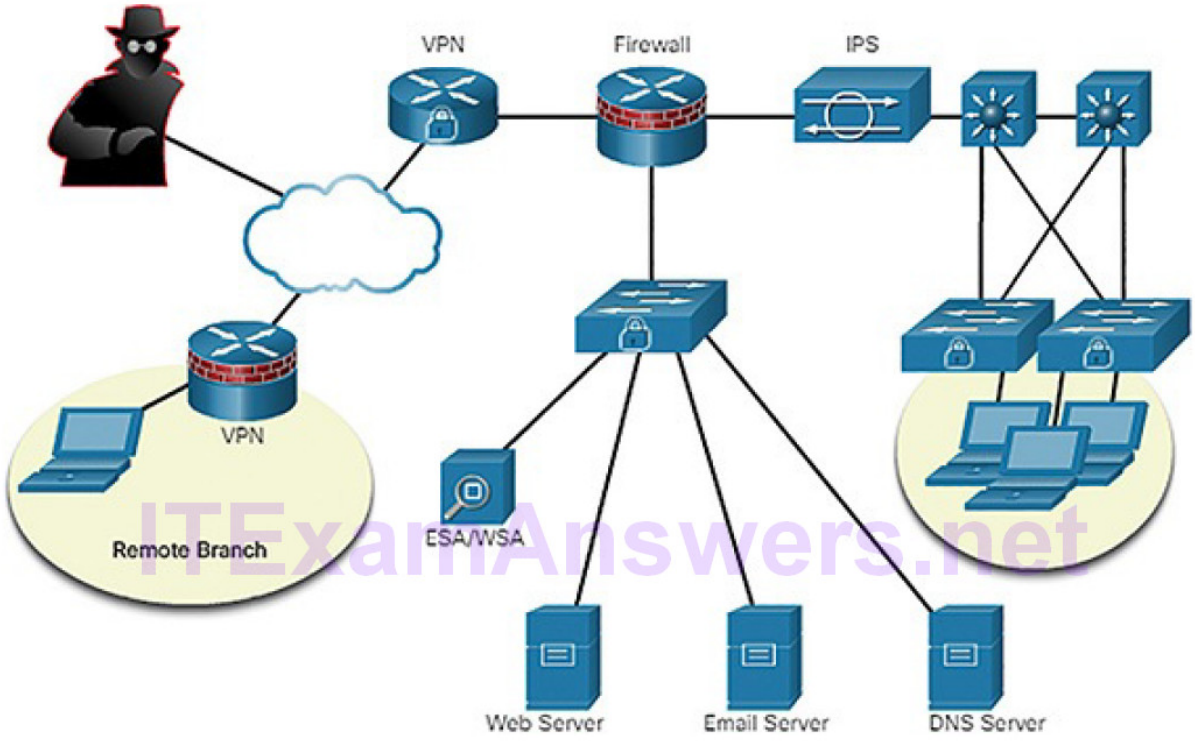
What Is Cryptography? (9.1.1)

In this topic, you will learn how cryptography is used to secure communications.

Securing Communications (9.1.1.1)

To ensure secure communications across both public and private networks, the first goal is to secure devices, including routers, switches, servers, and hosts.

For example, the topology in Figure 9-1 displays a number of secure devices indicated by the padlock or red brick firewall icon.



ITExamAnswers.net

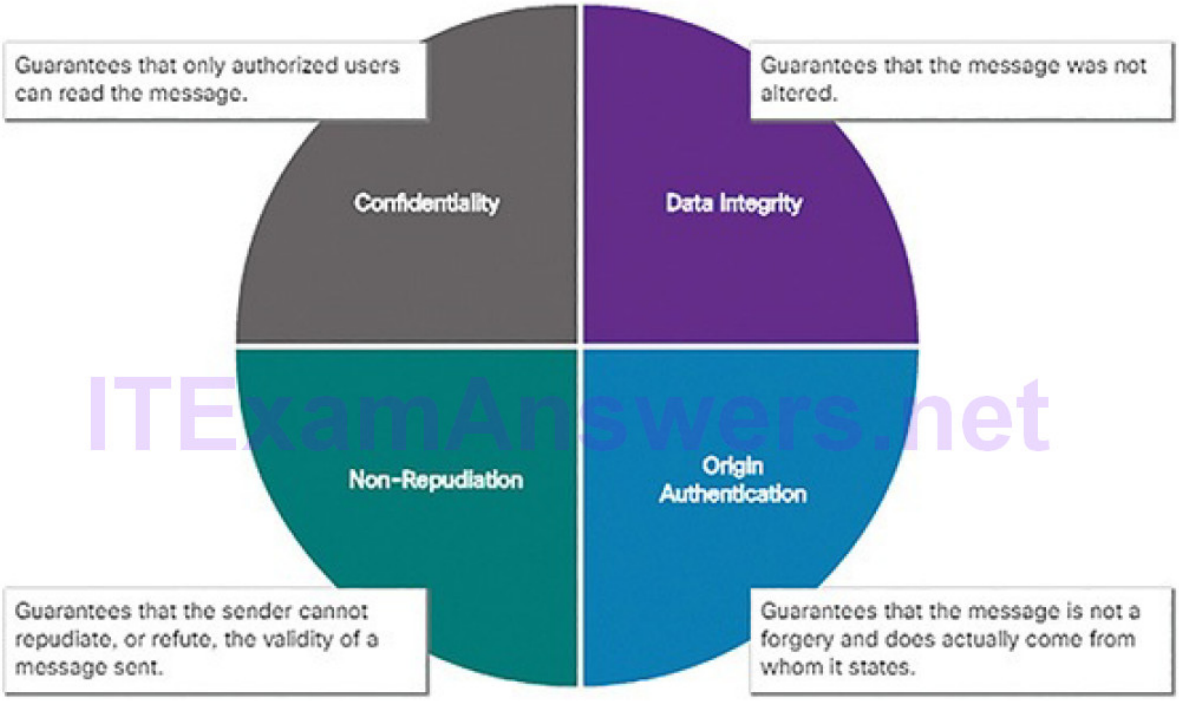
Figure 9-1 Secure Network Topology

Network infrastructure devices and hosts are secured using a variety of techniques:

- Device hardening AAA
- (Authentication, Authorization, and Accounting) access control
- Access control lists (ACLs)
- Firewalls
- Monitoring threats using an intrusion prevention system (IPS)
- Securing endpoints using Cisco Advanced Malware Protection (AMP)
- Enforcing email and web security using the Cisco Email Security Appliance (ESA) and Cisco Web Security Appliance (WSA)

The next goal is to secure the data as it travels across various links. This may include internal traffic, but of greater concern is protecting the data that travels outside of the organization to branch sites, telecommuter sites, and partner sites.

Secure communications consists of four elements, as summarized in Figure 9-2:



ITExamAnswers.net

Figure 9-2 Elements of Secure Communications

Data confidentiality: Guarantees that only authorized users can read the message. If the message is intercepted, it cannot be deciphered within a reasonable amount of time. Data confidentiality is implemented using symmetric and asymmetric encryption algorithms.

Data integrity: Guarantees that the message was not altered. Any changes to data in transit will be detected. Integrity is ensured by implementing either Message Digest version 5 (MD5) or Secure Hash Algorithm (SHA) hash-generating algorithms.

Origin authentication: Guarantees that the message is not a forgery and does actually come from whom it states. Many modern networks ensure authentication with protocols, such as hash message authentication code (HMAC).

Data non-repudiation: Guarantees that the sender cannot repudiate, or refute, the validity of a message sent. Non-repudiation relies on the fact that only the sender has the unique characteristics or signature for how that message is treated.

Note
MD5, SHA, and HMAC are discussed in more detail later in the chapter.

Cryptology (9.1.1.2)

Cryptology is used to secure communications. Cryptology is the science of making and breaking secret codes.

As shown in Figure 9-3, cryptology combines two separate disciplines:

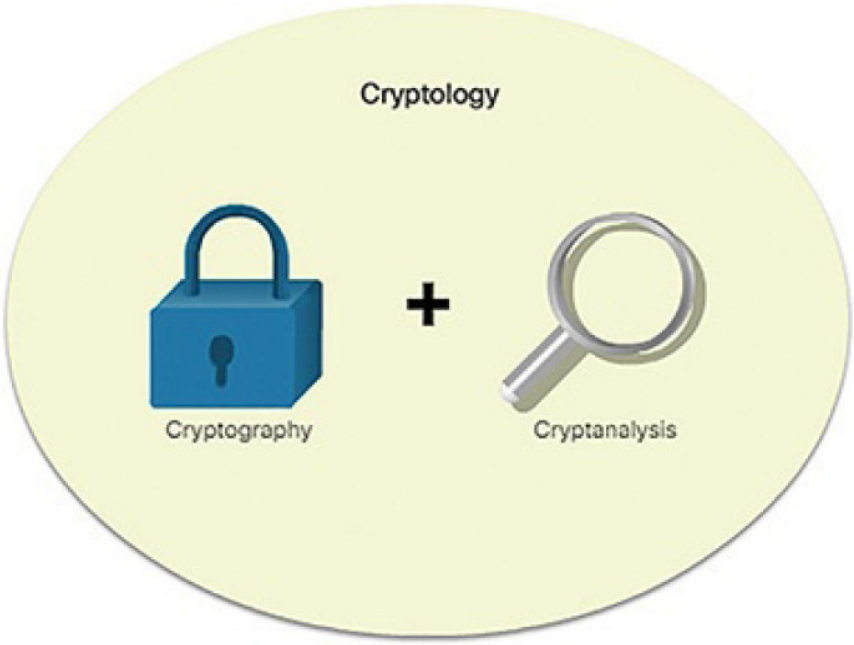


Figure 9-3 Cryptology = Cryptography + Cryptanalysis

Cryptography: This is the development and use of codes that are used for communicating privately. Specifically, it is the practice and study of techniques to secure communications. Historically, cryptography was synonymous with encryption.

Cryptanalysis: This is the breaking of those codes. Specifically, it is the practice and study of determining and exploiting weaknesses in cryptographic techniques.

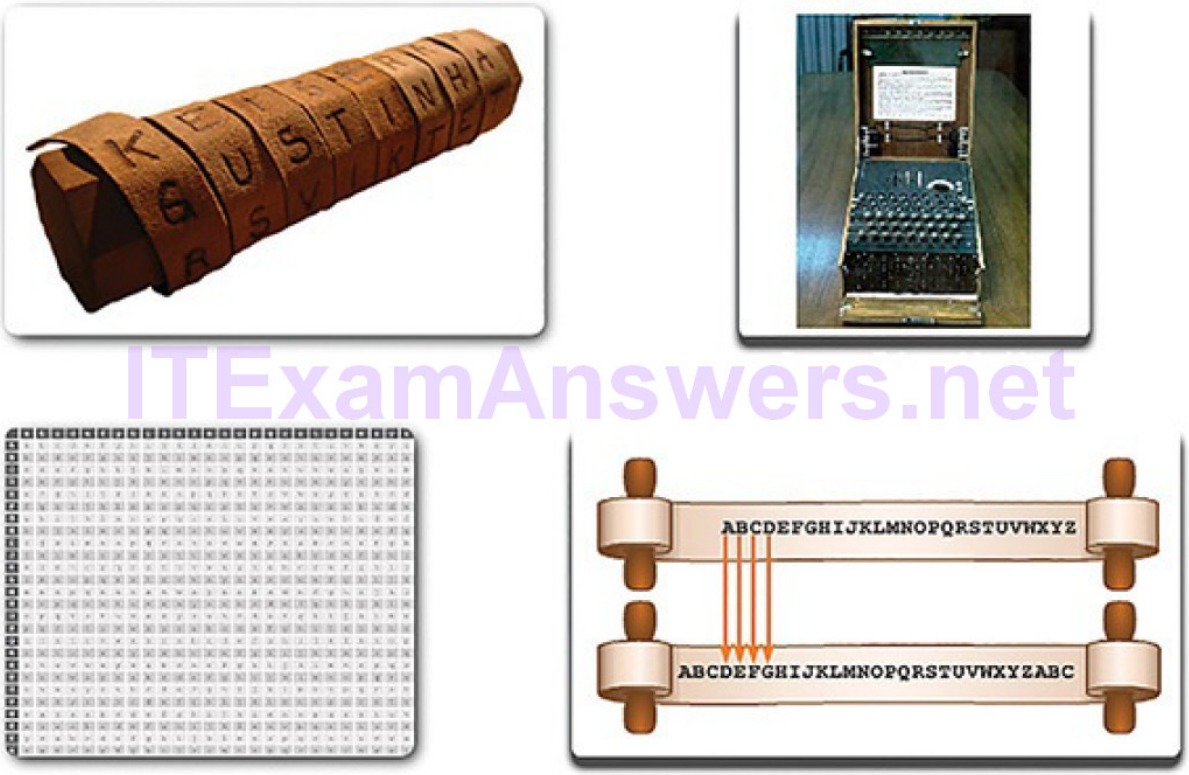
There is a symbiotic relationship between the two disciplines because each makes the other one stronger. National security organizations employ practitioners of both disciplines and put them to work against each other.

There have been times when one of the disciplines has been ahead of the other. For example, during the Hundred Years War between France and England, the cryptanalysts were leading the cryptographers. France mistakenly believed that the Vigenère cipher was unbreakable, and then the British cracked it. Somehistorians believe that the successful cracking of encrypted codes and messages had a major impact on the outcome of World War II.

Currently, it is believed that cryptographers have the advantage.

Cryptography: Ciphers (9.1.1.3)

Over the centuries, various cryptography methods, physical devices, and aids have been used to encrypt and decrypt text. The following historical ciphering examples are in displayed in Figure 9-4:



ITExamAnswers.net

Figure 9-4 Example of Ciphers

- Scytale (top left)
- Enigma machine (top right)
- Vigenère cipher (bottom left)
- Caesar cipher (bottom right)

Each of these encryption methods uses a specific algorithm, called a cipher. A cipher is an algorithm that consists of a series of well-defined steps that can befollowed as a procedure when encrypting and decrypting messages.

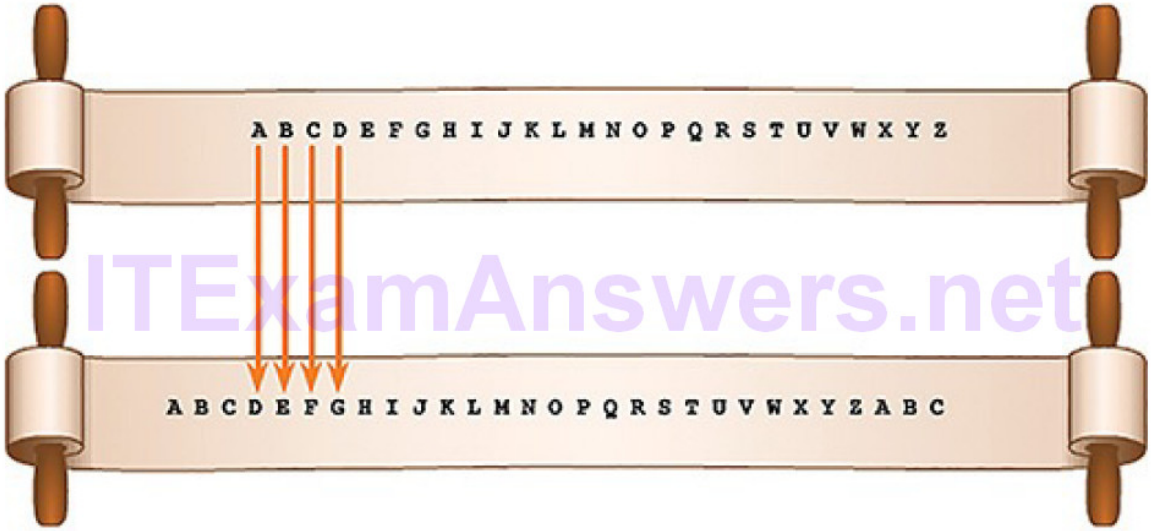
The following are types of ciphers that have been used over the years:

Substitution cipher: Substitution ciphers retain the letter frequency of the original message. The Caesar cipher was a simple substitution cipher. For example, refer to the plaintext message in Figure 9-5. If the key used was 3, the letter A was moved three letters to the right to become D, as shown in Figure 9-6. The resulting ciphertext is displayed in Figure 9-7.

ITExamAnswers.net

The plaintext message would be encoded using a key of 3.

Figure 9-5 Plaintext Message



ITExamAnswers.net

Shift the top scroll over by three characters (key of 3), an A becomes D, B becomes E, and so on.

Figure 9-6 Encrypting Using the Caesar Substitution Cipher

ITExamAnswers.net

The appearance of the text after encryption.

Figure 9-7 Resulting Ciphertext

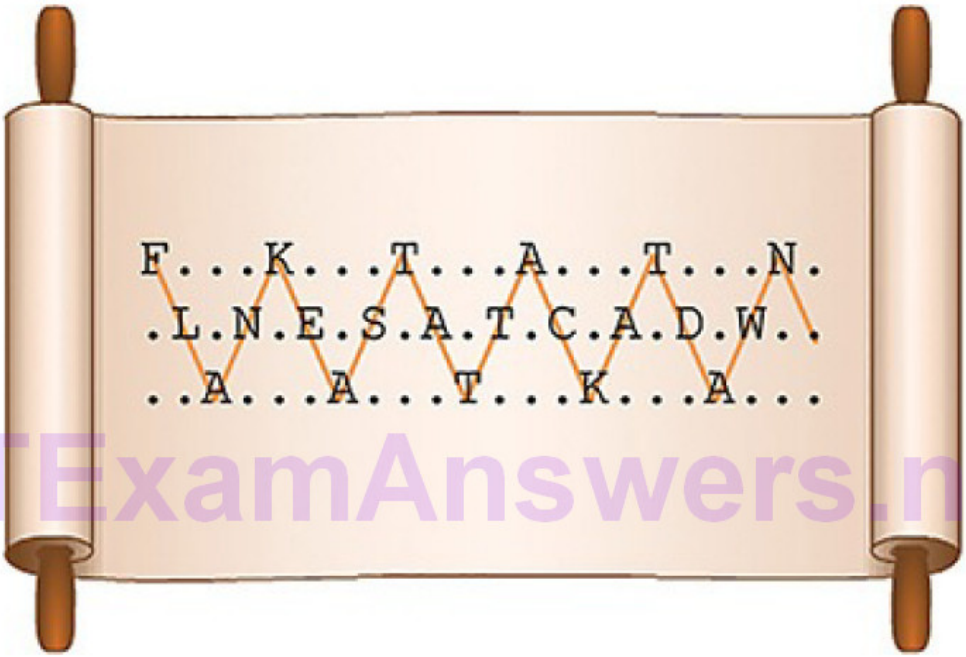
Transposition cipher: In transposition ciphers, no letters are replaced; they are simply rearranged. An example is taking the FLANK EAST ATTACK AT DAWN message and reversing it to read NWAD TAKCATTA TSAEKNALF. Another example of a transposition cipher is known as the rail fence cipher. For example, refer to the plaintext message in Figure 9-8.

Figure 9-9 displays how to transpose the message using a rail fence cipher with a key of 3. The key specifies that three lines are required when creating the encrypted code. The resulting ciphertext is displayed in Figure 9-10.

ITExamAnswers.net

The plaintext message would be encoded using a key of 3.

Figure 9-8 Plaintext Message



ITExamAnswers.net

Use a rail fence cipher and a key of 3.

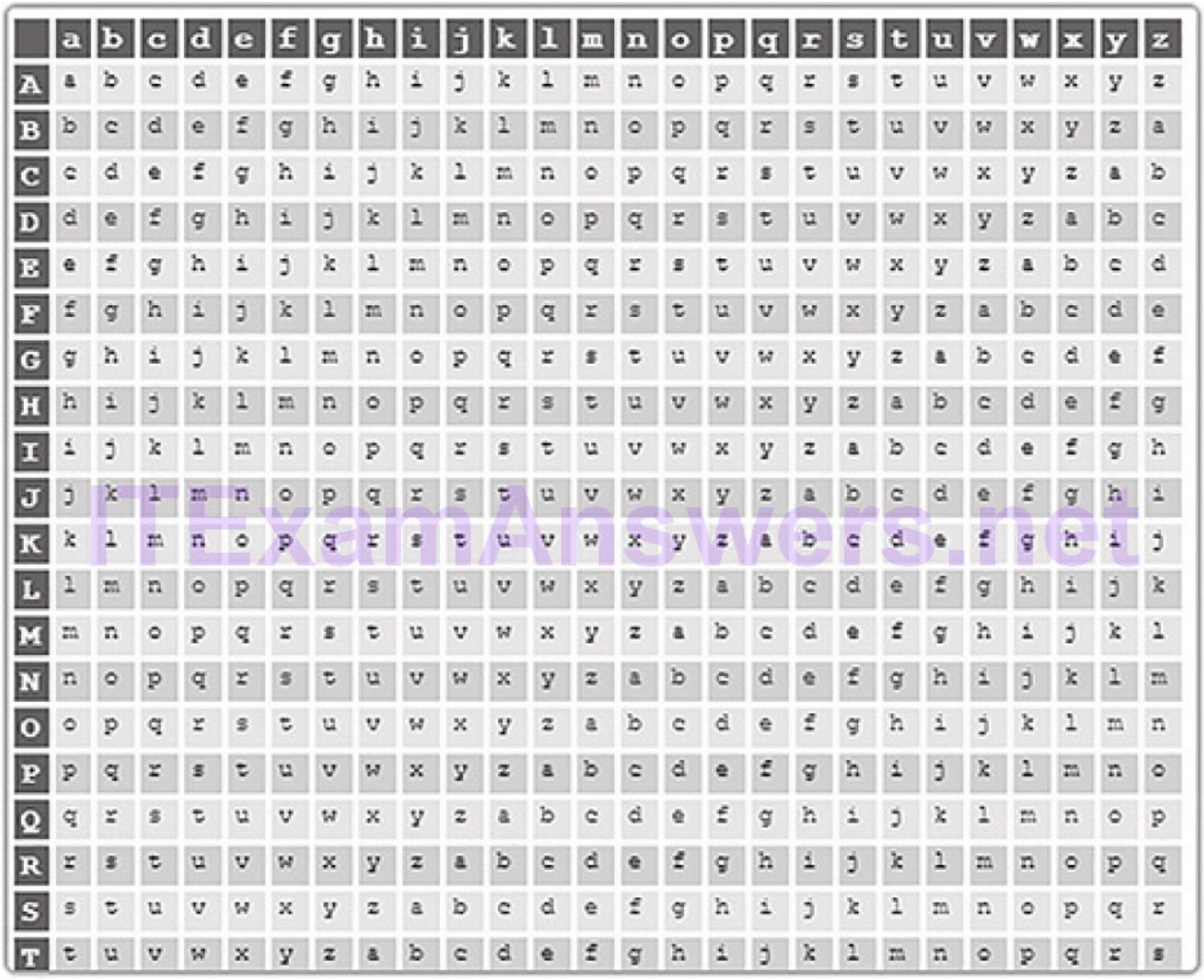
Figure 9-9 Encrypting Using the Rail Fence Transposition Cipher

ITExamAnswers.net

The appearance of the text after encryption.

Figure 9-10 Resulting Ciphertext

Polyalphabetic ciphers: Polyalphabetic ciphers are based on substitution, using multiple substitution alphabets. The famous Vigenère cipher is an example, shown in Figure 9-11. That cipher uses a series of different Caesarciphers that are based on the letters of a keyword. It is a simple form of polyalphabetic substitution and is therefore invulnerable to frequency analysis.



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
B	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
C	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
D	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
E	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
F	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
G	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
H	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
I	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
J	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
K	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
L	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
M	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
N	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
O	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
P	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
R	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
S	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
T	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s

ITExamAnswers.net

Figure 9-11 Vigenère Cipher

Cryptanalysis: Code Breaking (9.1.1.4)

Cryptanalysis is often used by cybercriminals to decipher encrypted messages. While cryptanalysis is often linked to mischievous purposes, it is actually a necessity.

Cryptanalysis is also used by governments in military and diplomatic surveillance, and by enterprises in testing the strength of security procedures. Enterprises and governments employ the services of mathematicians, scholars, security forensic experts, and cryptanalysts for these purposes.

Cryptanalysts are individuals who perform cryptanalysis to crack secret codes. Several methods are used in cryptanalysis:

Brute-force method: The cryptanalyst tries every possible key, knowing that eventually one of them will work. All algorithms are vulnerable to brute force. If every possible key is tried, one of the keys has to work.

Ciphertext method: The cryptanalyst has the ciphertext of several encrypted messages but no knowledge of the underlying plaintext.

Known-plaintext method: The cryptanalyst has access to the ciphertext of several messages and knows something about the plaintext underlying that ciphertext.

Chosen-plaintext method: The cryptanalyst chooses which data the encryption device encrypts and observes the ciphertext output.

Chosen-ciphertext method: The cryptanalyst can choose different ciphertext to be decrypted and has access to the decrypted plaintext.

Meet-in-the-middle method: The cryptanalyst knows a portion of the plaintext and the corresponding ciphertext.

Note
The actual processes of these cryptanalysis methods are beyond the scope of this course.

No algorithm is unbreakable. It is an ironic fact of cryptography that it is impossible to prove that any algorithm is secure. It can only be proven that it is not vulnerable to known cryptanalytic attacks.

Keys (9.1.1.5)

Authentication, integrity, and data confidentiality are implemented in many ways, using various protocols and algorithms. The choice of protocol and algorithm varies based on the level of security required to meet the goals of the network security policy.

Old encryption algorithms, such as the Caesar cipher or the Enigma machine, were based on the secrecy of the algorithm to achieve confidentiality.

With modern technology, security of encryption lies in the secrecy of the keys, not the algorithm.

Two terms that are used to describe keys are:

- **Key length:** Also called the key size, this is measured in bits. In this course, we will use the term key length.
- **Keyspace:** This is the number of possibilities that can be generated by a specific key length.

As key length increases, the keyspace increases exponentially. The keyspace of an algorithm is the set of all possible key values. A key that has n bits produces a keyspace that has 2n possible key values:

- A 2-bit (2^2) key length = a keyspace of 4 because there are four possible keys (00, 01, 10, and 11).
- A 3-bit (2^3) key length = a keyspace of 8, because there are eight possible keys (000, 001, 010, 011, 100, 101, 110, 111).
- A 4-bit (2^4) key length = a keyspace of 16 possible keys.
- A 40-bit (2^{40}) key length = a keyspace of 1,099,511,627,776 possible keys.

By adding one bit to the key, the keyspace is effectively doubled, as shown in Figure 9-12.

DES Key	Keyspace	# of Possible Keys
56-bit	2^{56} 1111111 1111111 1111111 1111111 1111111 1111111 1111111 1111111	72,000,000,000,000,000
57-bit	2^{57} 1111111 1111111 1111111 1111111 1111111 1111111 1111111 1111111 1	144,000,000,000,000,000
58-bit	2^{58} 1111111 1111111 1111111 1111111 1111111 1111111 1111111 1111111 11	288,000,000,000,000,000
59-bit	2^{59} 1111111 1111111 1111111 1111111 1111111 1111111 1111111 1111111 111	576,000,000,000,000,000
60-bit	2^{60} 1111111 1111111 1111111 1111111 1111111 1111111 1111111 1111111 1111	1,152,000,000,000,000,000

Figure 9-12 Double the Keyspace with Every Bit

Longer keys are more secure. However, they are also more resource intensive. Caution should be exercised when choosing longer keys because handling themcould add a significant load to the processor in lower-end products.

Lab 9.1.1.6: Encrypting and Decrypting Data Using OpenSSL

In this lab, you will complete the following objectives:

- Encrypting Messages with OpenSSL
- Decrypting Messages with OpenSSL

Lab 9.1.1.7: Encrypting and Decrypting Data Using a Hacker Tool

In this lab, you will complete the following objectives:

- Setup Scenario
- Create and Encrypt Files
- Recover Encrypted Zip File Passwords

Lab 9.1.1.8: Examining Telnet and SSH in Wireshark

In this lab, you will complete the following objectives:

- Examine a Telnet Session with Wireshark
- Examine an SSH Session with Wireshark

Integrity and Authenticity (9.1.2)

In this topic, you will learn the role of cryptography in ensuring the integrity and authenticity of data.

Cryptographic Hash Functions (9.1.2.1)

Hashes are used to verify and ensure data integrity. Hashing is based on a one-way mathematical function that is relatively easy to compute, but significantly harder to reverse. Grinding coffee is a good analogy of a one-way function. It iseasy to grind coffee beans, but it is almost impossible to put all of the tiny pieces back together to rebuild the original beans. The cryptographic hashing function can also be used to verify authentication.

As shown in Figure 9-13, a hash function takes a variable block of binary data, called the message, and produces a fixed-length, condensed representation, called the hash. The resulting hash is also sometimes called the message digest, digest, or digital fingerprint.

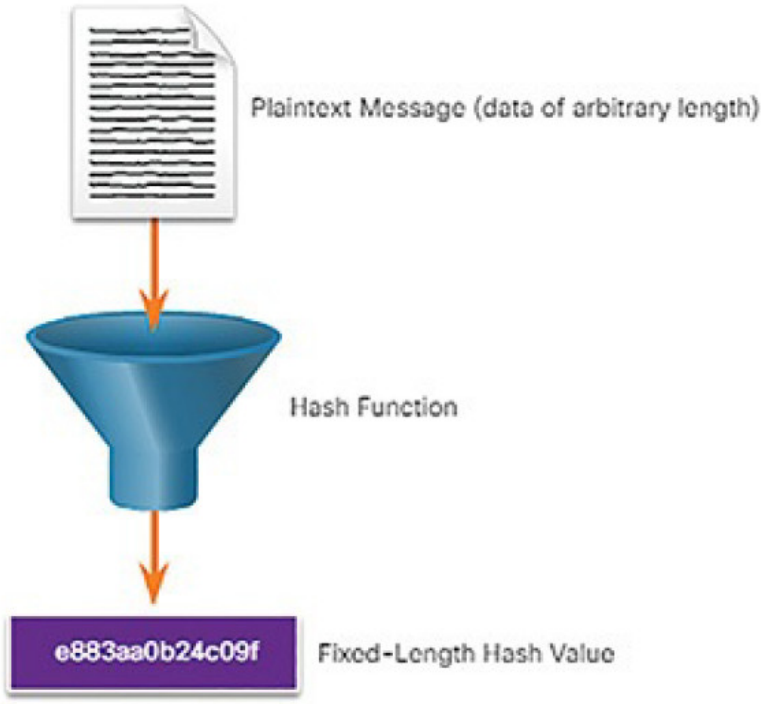


Figure 9-13 Creating a Hash

With hash functions, it is computationally infeasible for two different sets of data to come up with the same hash output. Every time the data is changed or altered, the hash value also changes. Because of this, cryptographic hash values are often called digital fingerprints. They can be used to detect duplicate data files, file version changes, and similar applications. These values are used to guard against an accidental or intentional change to the data, or accidental data corruption.

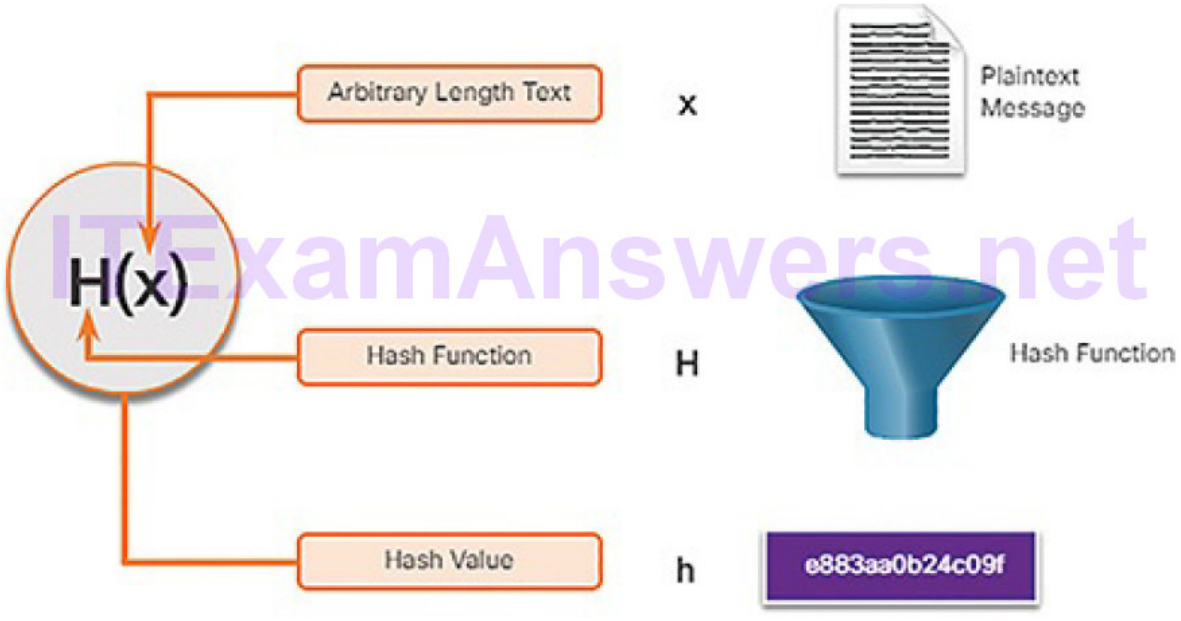
In Table 9-1, the config.txt and config-bk.txt files have the same hash value. This means that the content of these files is identical. The cryptographic hash function is applied in many different situations for entity authentication, data integrity, and data authenticity purposes.

Table 9-1 Creating a Hash

Date	Filename	Size	Hash
3/28/2017	topology.png	71 KB	d4a02c1520fb999b90fb5906a070130f
4/3/2017	config.txt	1 KB	aea9d4bb8c9457e37838db26bfe4a56e
4/10/2017	dev-gui.htm	53 KB	ee5b9602fd98414d3a2a051cb16a815a
4/12/2017	dev-image.zip	348,512 KB	f3b911b2dab14dfbc6f5a3a8c664033b
4/15/2017	config_bk.txt	1 KB	aea9d4bb8c9457e37838db26bfe4a56e

Cryptographic Hash Operation (9.1.2.2)

Mathematically, the equation h = H(x) is used to explain how a hash algorithm operates. As shown in Figure 9-14, a hash function H takes an input x and returns a fixed-size string hash value h.



ITExamAnswers.net

Figure 9-14 Hashing Formula

Figure 9-14 summarizes the mathematical process.

A cryptographic hash function should have the following properties:

- The input can be any length.
- The output has a fixed length.
- H(x) is relatively easy to compute for any given x.
- H(x) is one way and not reversible.
- H(x) is collision free, meaning that two different input values will result in different hash values.

If a hash function is hard to invert, it is considered a one-way hash. Hard to invert means that given a hash value of h, it is computationally infeasible to find an input for x such that h = H(x).

MD5 and SHA (9.1.2.3)

Hash functions are used to ensure the integrity of a message. They ensure data has not changed accidentally or intentionally.

In Figure 9-15, the sender is sending a $100 money transfer to Alex.

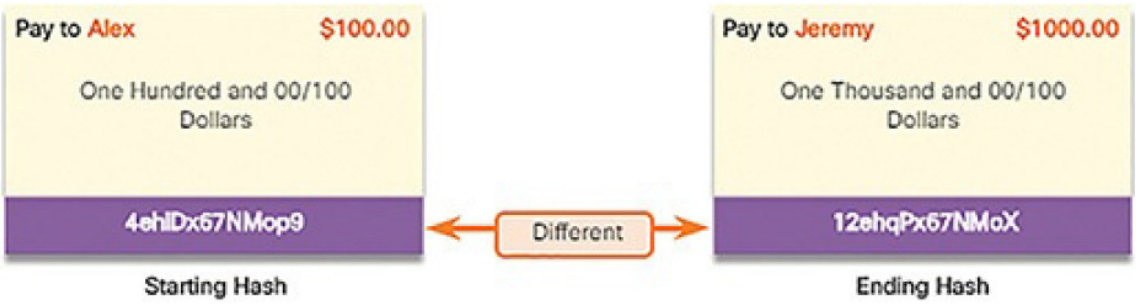


Figure 9-15 Hash Algorithms

The sender wants to ensure that the message is not altered on its way to the receiver.

1. The sending device inputs the message into a hashing algorithm and computes its fixed-length hash of 4ehiDx67NMop9.

2. This hash is then attached to the message and sent to the receiver. Both the message and the hash are in plaintext.

3. The receiving device removes the hash from the message and inputs the message into the same hashing algorithm. If the computed hash is equal to the one that is attached to the message, the message has not been altered during transit. If the hashes are not equal, as shown in Figure 9-15, then the integrity of the message can no longer be trusted.

There are three well-known hash functions:

Message Digest 5 (MD5) with 128-bit digest: Developed by Ron Rivest and used in a variety of Internet applications, MD5 is a one-way function that produces a 128-bit hashed message, as shown in Figure 9-16. MD5 is considered to be a legacy algorithm and should be avoided, and used only when no better alternatives are available. It is recommended that SHA-2 be used instead.

Secure Hash Algorithm 1 (SHA-1): Developed by the U.S. National Institute of Standards and Technology (NIST) in 1994 and is very similar to the MD5 hash functions, as shown in Figure 9-17. Several versions exist.
SHA-1 creates a 160-bit hashed message and is slightly slower than MD5.
SHA-1 has known flaws and is a legacy algorithm.

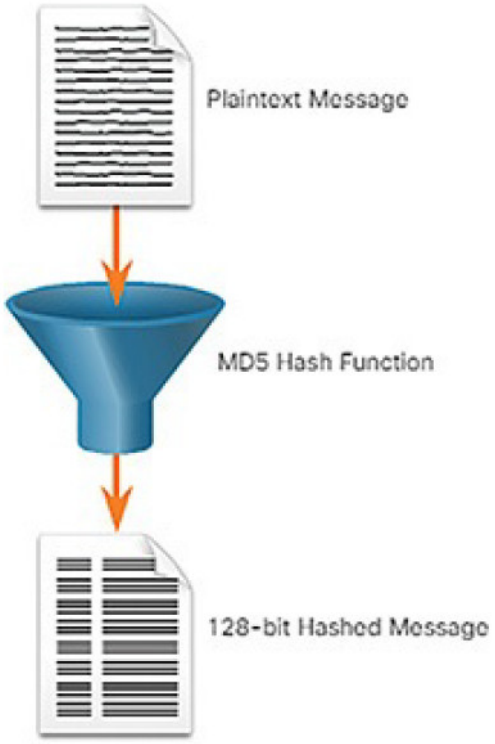


Figure 9-16 MD5 Hashing Algorithm

Secure Hash Algorithm 2 (SHA-2):
Developed by NIST and includes SHA-224 (224 bit), SHA-256 (256 bit), SHA-384 (384 bit), and SHA-512 (512 bit). SHA-256, SHA-384, and SHA-512 are next-generation algorithms and should be used whenever possible.

While hashing can be used to detect accidental changes, it cannot be used to guard against deliberate changes. There is no unique identifying information from the sender in the hashing procedure. This means that anyone can compute a hash for any data, as long as they have the correct hash function.

For example, when the message traverses the network, a potential attacker could intercept the message, change it, recalculate the hash, and append it to the message. The receiving device will only validate against whatever hash is appended.

Therefore, hashing is vulnerable to man-in-the-middle attacks and does not provide security to transmitted data. To provide integrity and origin authentication, something more is required.

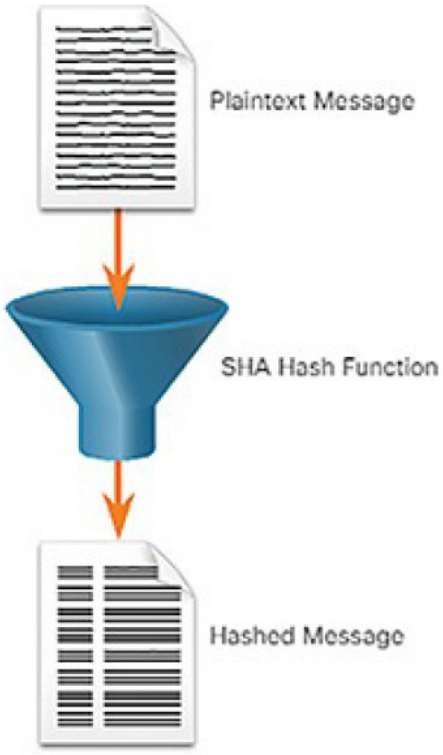
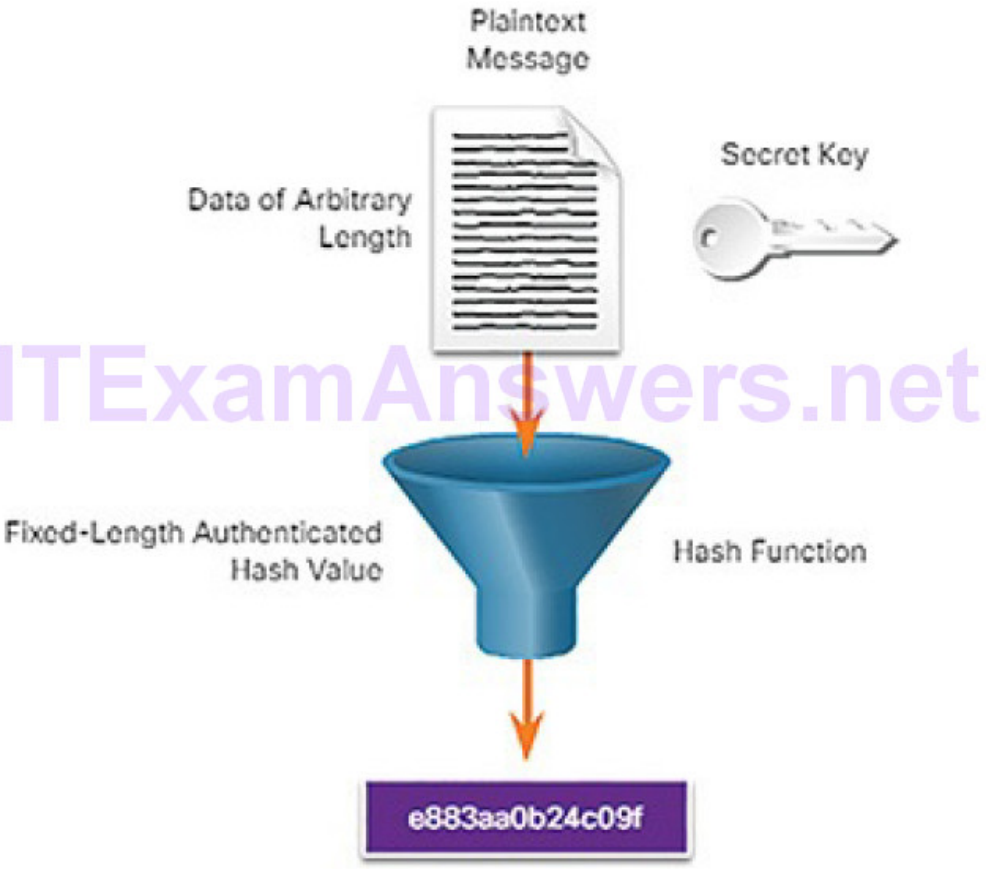


Figure 9-17 SHA Hashing Algorithm

Hash Message Authentication Code (9.1.2.4)

To add authentication to integrity assurance, a keyed-hash message authentication code (HMAC; also sometimes abbreviated as KHMAC) is used.To add authentication, HMAC uses an additional secret key as input to the hash function.

As shown in Figure 9-18, an HMAC is calculated using a specific algorithm (e.g., MD5 or SHA-1) that combines a cryptographic hash function with a secret key. Hash functions are the basis of the protection mechanism of HMACs.



ITExamAnswers.net

Figure 9-18 HMAC Hashing Algorithm

Only the sender and the receiver know the secret key, and the output of the hash function now depends on the input data and the secret key. Only parties who have access to that secret key can compute the digest of an HMAC function. This characteristic defeats man-in-the-middle attacks and provides authentication of the data origin.

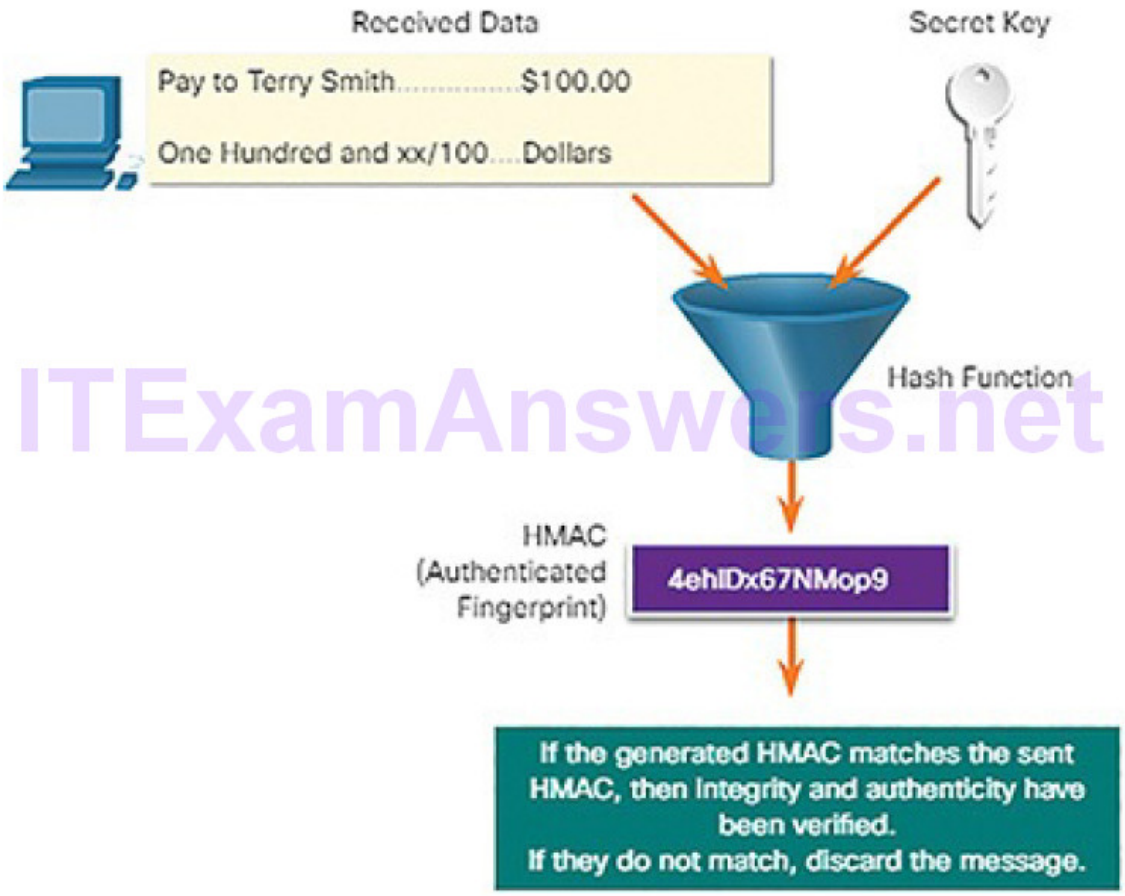
If two parties share a secret key and use HMAC functions for authentication, a properly constructed HMAC digest of a message that a party has received indicates that the other party was the originator of the message. This is because the other party possesses the secret key.

As shown in Figure 9-19, the sending device inputs data (such as Terry Smith's pay of $100 and the secret key) into the hashing algorithm and calculates the fixed-length HMAC digest. This authenticated digest is then attached to the message and sent to the receiver.

ITExamAnswers.net

Figure 9-19 Creating the HMAC Value

In Figure 9-20, the receiving device removes the digest from the message and uses the plaintext message with its secret key as input to the same hashing function.



ITExamAnswers.net

Figure 9-20 Verifying the HMAC Value

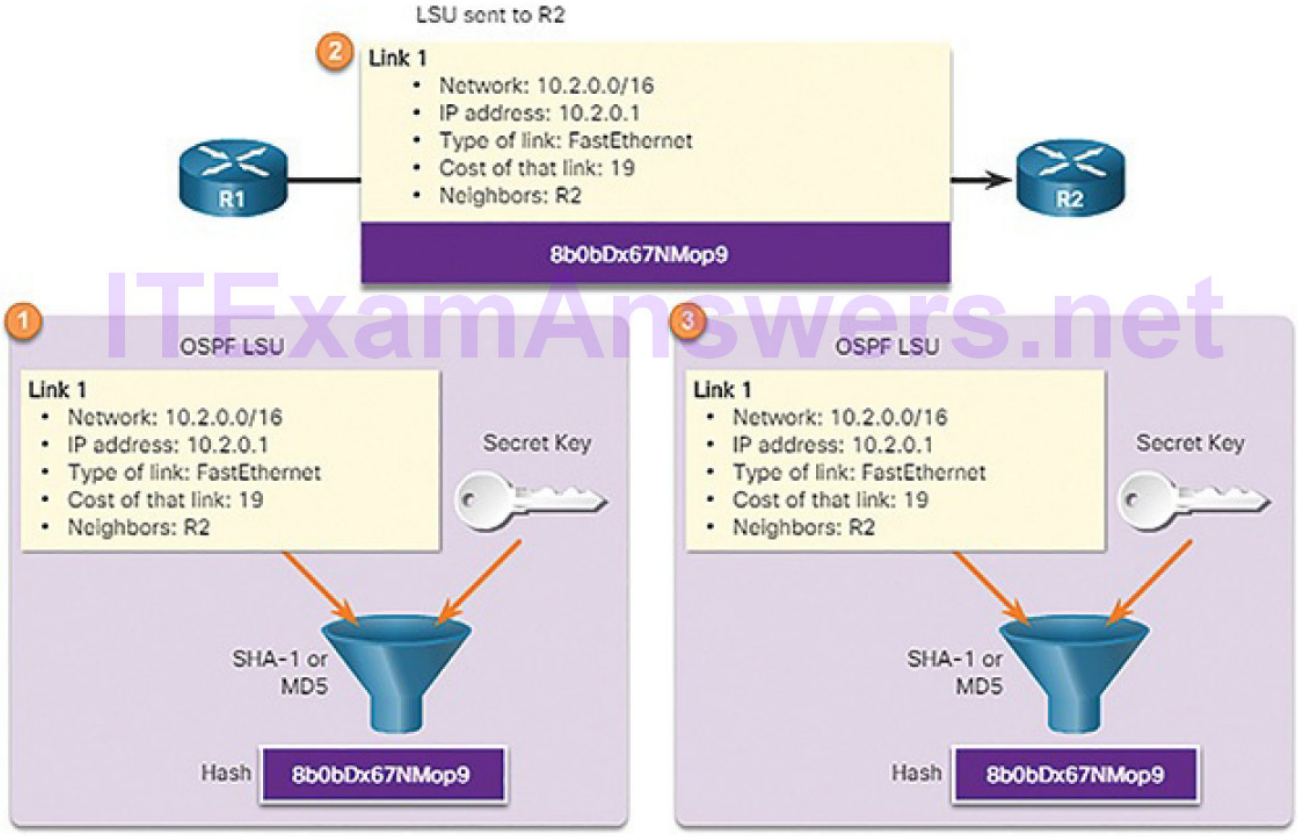
If the digest that is calculated by the receiving device is equal to the digest that was sent, the message has not been altered. Additionally, the origin of the message is authenticated because only the sender possesses a copy of the shared secret key. The HMAC function has ensured the authenticity of the message.

Figure 9-21 illustrates how HMACs are used by Cisco routers configured to use Open Shortest Path First (OSPF) routing authentication. R1 is sending a link state update (LSU) regarding a route to network 10.2.0.0/16:

1. R1 calculates the hash value using the LSU message and the secret key.

2. The resulting hash value is sent with the LSU to R2.

3. R2 calculates the hash value using the LSU and its secret key. R2 accepts the update if the hash values match. Otherwise, R2 discards the update.



ITExamAnswers.net

Figure 9-21 Verifying the HMAC Value: Details

Lab 9.1.2.5: Hashing Things Out

In this lab, you will complete the following objectives:

- Creating Hashes with OpenSSL
- Verifying Hashes

Confidentiality (9.1.3)

In this topic, you will learn how cryptographic approaches enhance data confidentiality.

Encryption (9.1.3.1)

There are two classes of encryption used to provide data confidentiality. These two classes differ in how they use keys:

Symmetric encryption algorithms: Encryption algorithms that use the same key to encrypt and decrypt data. They are based on the premise that each communicating party knows the pre-shared key.

Asymmetric encryption algorithms: Encryption algorithms that use different keys to encrypt and decrypt data. They are based on the assumption that the two communicating parties have not previously shared a secret and must establish a secure method to do so.

Asymmetric algorithms are resource intensive and slower to execute.

The following summarizes some differences between each encryption algorithm method.

Symmetrical Encryption

- Uses the same key to encrypt and decrypt data.
- Key lengths are short (40 bits to 256 bits).
- Faster than asymmetrical encryption.
- Commonly used for encrypting bulk data such as in VPN traffic.

Asymmetrical Encryption

- Uses different keys to encrypt and decrypt data.
- Key lengths are long (512 bits to 4096 bits)
- Computationally tasking and therefore slower than symmetrical encryption.
- Commonly used for quick data transactions such as HTTPS when accessing your bank data.

Symmetric Encryption (9.1.3.2)

Symmetric algorithms use the same pre-shared key to encrypt and decrypt data. A pre-shared key, also called a secret key, is known by the sender and receiver before any encrypted communications can take place.

To help illustrate how symmetric encryption works, consider an example where Alice and Bob live in different locations and want to exchange secret messages with one another through the mail system. In this example, Alice wants to send a secret message to Bob.

In Figure 9-22, Alice and Bob have identical keys to a single padlock.

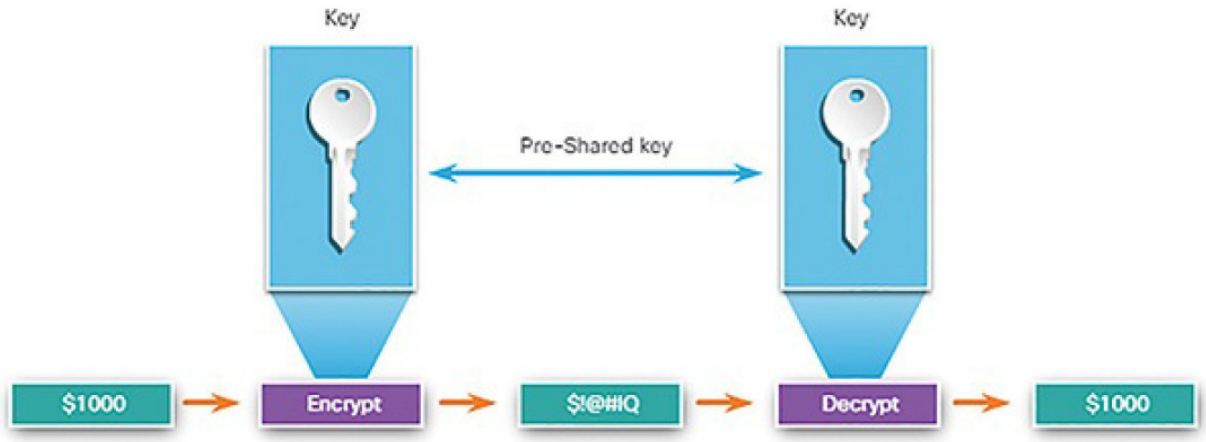


Figure 9-22 Symmetric Encryption Example

These keys were exchanged prior to sending any secret messages. Alice writes a secret message and puts it in a small box that she locks using the padlock with her key. She mails the box to Bob. The message is safely locked inside the box as the box makes its way through

the post office system. When Bob receives the box, he uses his key to unlock the padlock and retrieve the message. Bob can use the same box and padlock to send a secret reply back to Alice.

Today, symmetric encryption algorithms are commonly used with VPN traffic. This is because symmetric algorithms use less CPU than asymmetric encryption algorithms, which are discussed later. This allows the encryption and decryptionof data to be fast when using a VPN. When using symmetric encryption algorithms, like any other type of encryption, the longer the key, the longer it will take for someone to discover the key. Most encryption keys are between 112 and 256 bits. To ensure that the encryption is safe, a minimum key length of 128 bits should be used. Use a longer key for more secure communications.

Symmetric Encryption Algorithms (9.1.3.3)

Encryption algorithms are often classified as a block cipher or stream cipher:

Block ciphers: Block ciphers transform a fixed-length block of plaintext into a common block of ciphertext of 64 or 128 bits, as shown in Figure 9-23. Common block ciphers include DES with a 64-bit block size and AES with a 128-bit block size.

Figure 9-23 Block Cipher

Stream ciphers: Stream ciphers encrypt plaintext one byte or one bit at a time, as shown in Figure 9-24. Stream ciphers are basically a block cipher with a block size of one byte or bit. Stream ciphers are typically faster than block ciphers because data is continuously encrypted. Examples of stream ciphers include RC4 and A5, the latter of which is used to encrypt GSM cell phone communications. Data Encryption Standard (DES) can also be used in stream cipher mode.

Figure 9-24 Stream Cipher

Well-known symmetric encryption algorithms include

Data Encryption Standard (DES): This is an older legacy symmetric encryption algorithm. It can be used in stream cipher mode but usually operates in block mode by encrypting data in 64-bit block size. Table 9-2 summarizes DES.

Table 9-2 DES Scorecard

DES Characteristics	
Description	Data Encryption Standard
Timeline	Standardized 1976
Type of algorithm	Symmetric
Key size	56 bits
Speed	Medium
Time to crack (assuming a computer could try 255 keys per second)	Days (6.4 days by the COPACABANA machine, a specialized cracking device)
Resource consumption	Medium

3DES (Triple DES): This is a newer version of DES, but it repeats the DES algorithm process three times. It is more computationally taxing than DES.
The basic algorithm has been well tested in the field for more than 35 years and is therefore considered very trustworthy when implemented using very short key lifetimes. Table 9-3 summarizes 3DES.

Table 9-3 3DES Scorecard

3DES Characteristics	
Description	Triple Data EncryptionStandard
Timeline	Standardized 1977
Type of algorithm	Symmetric
Key size	112 and 168 bits
Speed	Low
Time to crack (assuming a computer could try 255 keys per second)	4.6 billion years with current technology
Resource consumption	Medium

Advanced Encryption Standard (AES): Based on the Rijndael cipher, it is a popular and recommended symmetric encryption algorithm. It offers nine combinations of key and block length by using a variable key length of 128-, 192-, or 256-bit key to encrypt data blocks that are 128, 192, or 256 bits long. AES is a secure and more efficient algorithm than 3DES. Table 9-4 summarizes AES. AES counter mode (AES-CTR) is the preferred encryption algorithm for SSHv2. It can use any of the AES key lengths, such as AES256-CTR.

Table 9-4 AES Scorecard

AES Characteristics	
Description	Advanced Encryption Standard
Timeline	Official standard since 2001
Type of algorithm	Symmetric
Key size	128, 192, and 256 bits
Speed	High
Time to crack (assuming a computer could try 255 keys per second)	149 trillion years
Resource consumption	Low

Software-Optimized Encryption Algorithm (SEAL): SEAL is a fast, alternative symmetric encryption algorithm to DES, 3DES, and AES. It is a stream cipher that uses a 160-bit encryption key. SEAL has a lower impact on the CPU compared to other software-based algorithms but is still considered unproven. Table 9-5 summarizes SEAL.

Table 9-5 SEAL Scorecard

SEAL Characteristics	
Description	Software-Optimized Encryption Algorithm
Timeline	First published in 1994; current version is 3.0 (1997)
Type of algorithm	Symmetric
Key size	160 bits
Speed	High

Time to crack (assuming a computer could try 255 keys per second)	Unknown but considered very safe
Resource consumption	Low

Rivest ciphers (RC) series algorithms (includes RC2, RC4, RC5, and RC6): This was developed by Ron Rivest. Several variations have been developed, but RC4 is the most prevalent in use. RC4 is a stream cipher and is used to secure web traffic in SSL and TLS. Table 9-6 summarizes the RC algorithms.

Table 9-6 RC Algorithms Summary

RC Algorithms	Timeline	Type of Algorithm	Key Size in Bits
RC2	1987	Block cipher	40 and 64
RC4	1987	Stream cipher	1 to 256
RC5	1994	Block cipher	0 to 2048
RC6	1998	Block cipher	128, 192, or 256

Note

There are many other symmetric encryption algorithms, such as Blowfish, Twofish, Threefish, and Serpent. However, these algorithms are beyond the scope of this course.

Asymmetric Encryption Algorithms (9.1.3.4)

Asymmetric algorithms, also called public key algorithms, are designed so that the key that is used for encryption is different from the key that is used for decryption, as shown in Figure 9-25. The decryption key cannot, in any reasonable amount of time, be calculated from the encryption key and vice versa.

Figure 9-25 Asymmetric Encryption Example

Asymmetric algorithms use a public key and a private key. Both keys are capable of the encryption process, but the complementary paired key is required for decryption. The process is also reversible in that data encrypted with the public key requires the private key to decrypt.

This process enables asymmetric algorithms to achieve confidentiality, authentication, and integrity.

Because neither party has a shared secret, very long key lengths must be used. Asymmetric encryption can use key lengths between 512 to 4,096 bits. Key lengths greater than or equal to 1,024 bits can be trusted, while shorter keylengths are considered unreliable.

Examples of protocols that use asymmetric key algorithms include

Internet Key Exchange (IKE): This is a fundamental component of IPsec VPNs.
Secure Sockets Layer (SSL): This is now implemented as IETF standard Transport Layer Security (TLS).
Secure Shell (SSH): This is a protocol that provides a secure remote access connection to network devices.
Pretty Good Privacy (PGP): This is a computer program that provides cryptographic privacy and authentication. It is often used to increase the security of email communications.

Asymmetric algorithms are substantially slower than symmetric algorithms. Their design is based on computational problems, such as factoring extremely large numbers or computing discrete logarithms of extremely large numbers.

Because they lack speed, asymmetric algorithms are typically used in low-volume cryptographic mechanisms, such as digital signatures and key exchange. However, the key management of asymmetric algorithms tends to be simpler than symmetric algorithms, because usually one of the two encryption or decryption keys can be made public.

Common examples of asymmetric encryption algorithms are described in Table 9-7.

Table 9-7 Asymmetric Encryption Algorithms

Asymmetric Encryption Algorithm	Key Length (in Bits)	Description

Diffie-Hellman (DH)	512, 1024, 2048, 3072, or 4096	The Diffie-Hellman algorithm is a public key algorithm invented in 1976 by Whitfield Diffie and Martin Hellman. It allows two parties to agree on a key that they can use to encrypt messages they want to send to each other. The security of this algorithm depends on the assumption that it is easy to raise a number to a certain power, but difficult tocompute which power was used given the number and the outcome.
Digital Signature Standard (DSS) and Digital Signature Algorithm (DSA)	512 to 1024	DSS was created by NIST and specifies DSA as the algorithm for digital signatures. DSA is a public key algorithm based on the ElGamal signature scheme. Signature creation speed is similar with RSA, but is 10 to 40 times as slow for verification.
RSA encryption algorithms	512 to 2048	Developed by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT in 1977, RSA is an algorithm for public key cryptography that is based on the current difficulty of factoring very large numbers. It is the first algorithm known to be suitable for signing as well as encryption, and one of the first great advances in public key cryptography. It is widely used in electronic commerce protocols, and is believed to be secure given sufficiently long keys and the use of up-to-date implementations.
ElGamal	512 to 1024	An asymmetric key encryption algorithm for public key cryptography which is based on the Diffie-Hellman key agreement. It is described by Taher ElGamal in 1984 and is used in GNU Privacy Guard software, PGP, and other cryptosystems. A disadvantage of the ElGamal system is that the encrypted message becomes very big, about twice the size of the original message, and for this reason ElGamal is only used for small messages such as secret keys.
Elliptical curve techniques	160	Elliptic curve cryptography was invented by Neil Koblitz and Victor Miller in the mid 1980s. Can be used to adapt many cryptographic algorithms, such as Diffie-Hellman or ElGamal. The main advantage of elliptic curve cryptography is that the keys can be much smaller.

Asymmetric Encryption: Confidentiality (9.1.3.5)

Asymmetric algorithms are used to provide confidentiality without pre-sharing a password. The confidentiality objective of asymmetric algorithms is initiated when the encryption process is started with the public key.

The process can be summarized using the formula:

Public Key (Encrypt) + Private Key (Decrypt) = Confidentiality

When the public key is used to encrypt the data, the private key must be used to decrypt the data. Only one host has the private key; therefore, confidentiality is achieved.

If the private key is compromised, another key pair must be generated to replace the compromised key.

For example, in Figure 9-26, Alice requests and obtains Bob's public key.

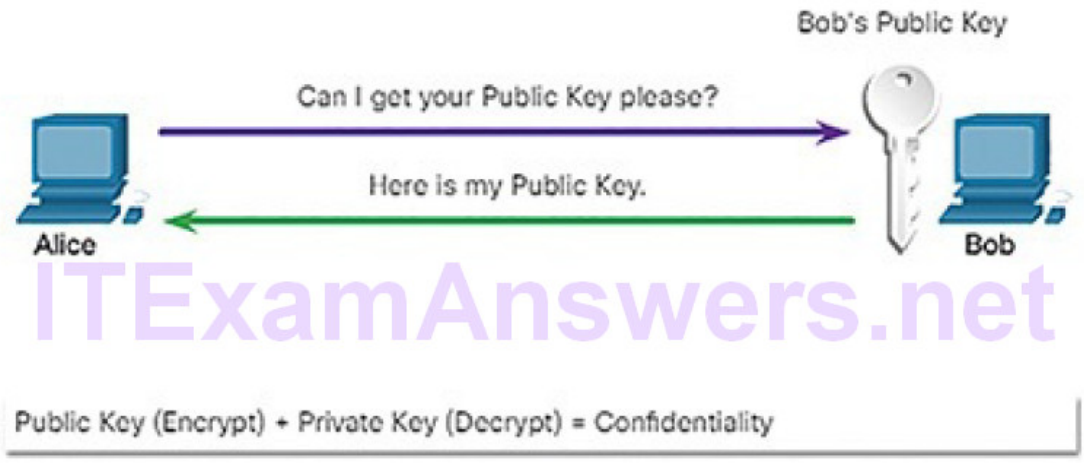


Figure 9-26 Alice Acquires Public Key

In Figure 9-27, Alice uses Bob's public key to encrypt a message using an agreed-upon algorithm.

Alice sends the encrypted message to Bob. Bob then uses his private key to decrypt the message as shown in Figure 9-28.

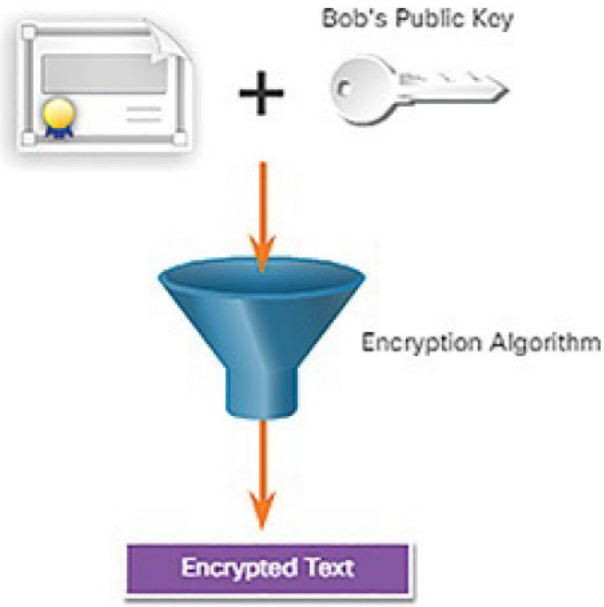


Figure 9-27 Alice Encrypts Message Using Bob's Public Key

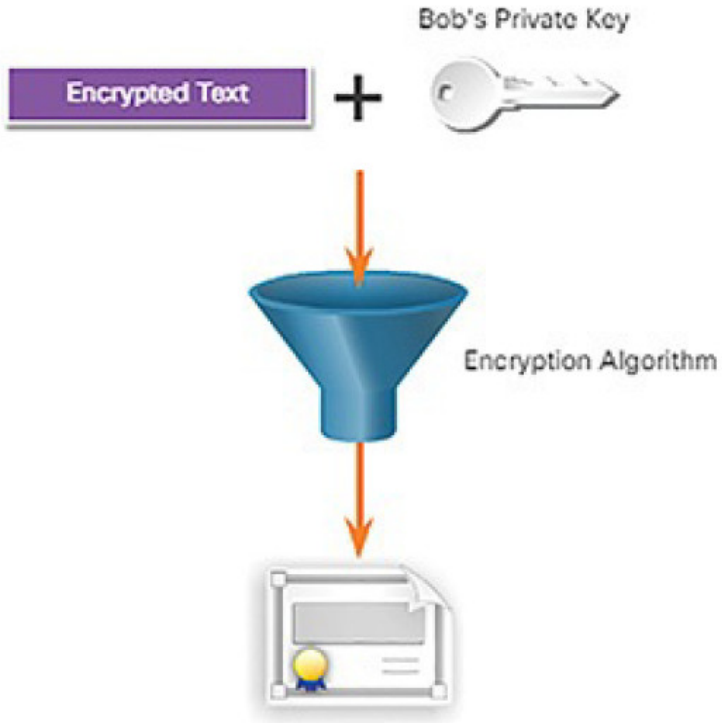


Figure 9-28 Bob Decrypts the Message Using His Private Key

Asymmetric Encryption: Authentication (9.1.3.6)

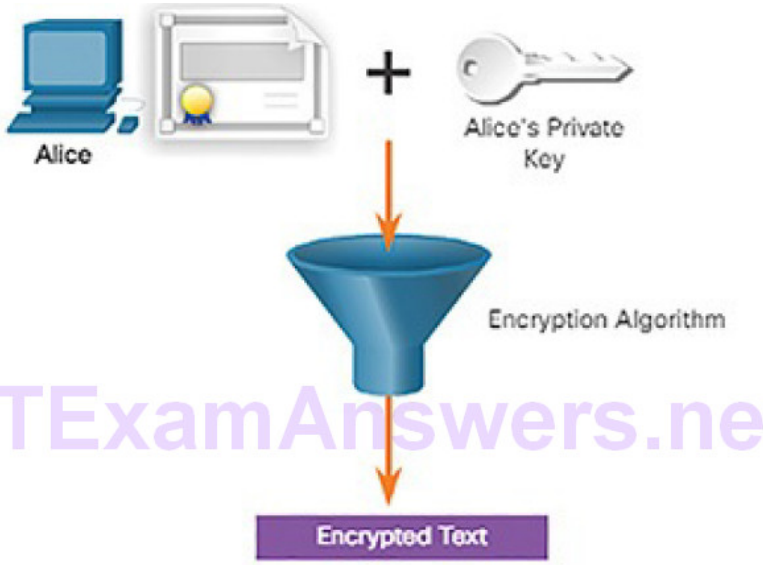
The authentication objective of asymmetric algorithms is initiated when the encryption process is started with the private key.

The process can be summarized using the formula:

Private Key (Encrypt) + Public Key (Decrypt) = Authentication

When the private key is used to encrypt the data, the corresponding public key must be used to decrypt the data. Because only one host has the private key, only that host could have encrypted the message, providing authentication of the sender. Typically, no attempt is made to preserve the secrecy of the public key, so any number of hosts can decrypt the message. When a host successfully decrypts a message using a public key, it is trusted that the private key encrypted the message, which verifies who the sender is. This is a form of authentication.

For example, in Figure 9-29, Alice encrypts a message using her private key.



ITExamAnswers.net

Private Key (Encrypt) + Public Key (Decrypt) = Authentication

Figure 9-29 Alice Encrypts the Message Using Her Private Key

Alice sends the encrypted message to Bob. Bob needs to authenticate that the message did, indeed, come from Alice. Therefore, in Figure 9-30, Bob requests Alice's public key.

Bob needs to verify that the message actually came from Alice. He requests and acquires Alice's public key.

Figure 9-30 Bob Requests Alice's Public Key

In Figure 9-31, Bob uses Alice's public key to decrypt the message.

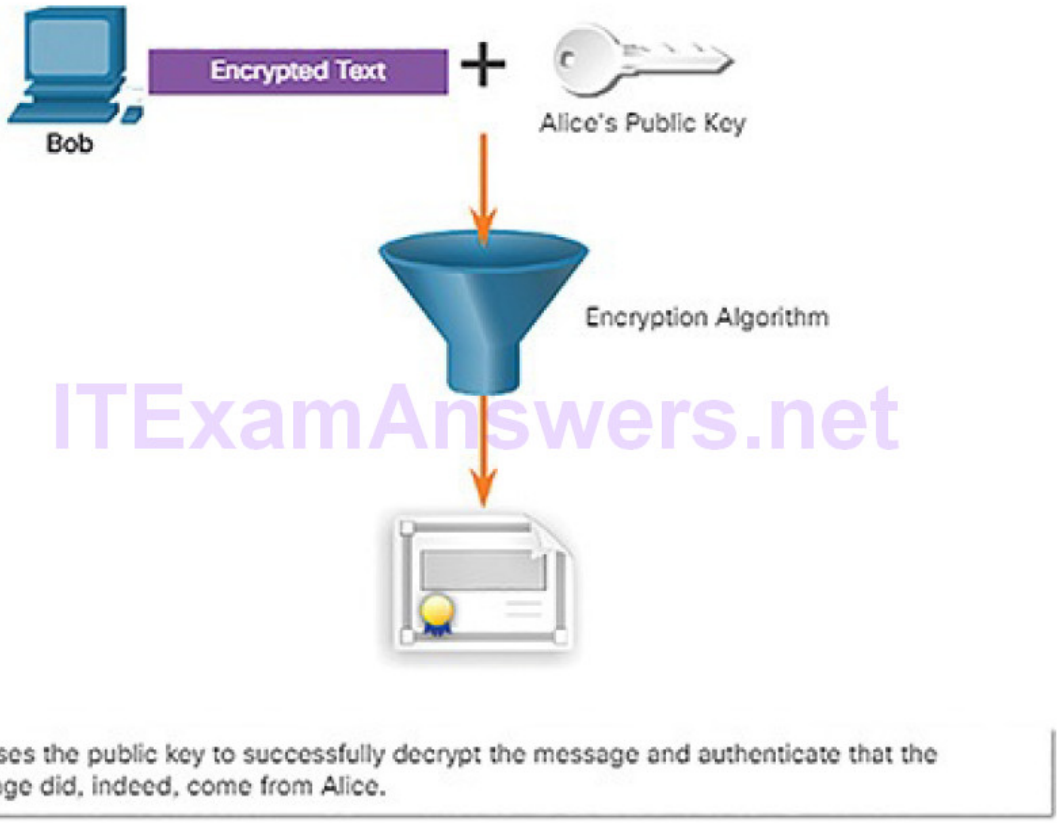


Figure 9-31 Bob Deciphers the Message Using the Public Key

Asymmetric Encryption: Integrity (9.1.3.7)

Combining the two asymmetric encryption processes provides message confidentiality, authentication, and integrity.

The following example will be used to illustrate this process. In this example, a message will be ciphered using Bob's public key and a ciphered hash will beencrypted using Alice's private key to provide confidentiality, authenticity, and integrity.

The process in Figure 9-32 provides confidentiality. Alice wants to send a message to Bob, ensuring that only Bob can read the document. In other words, Alice wants to ensure message confidentiality. Alice uses the public key of Bob to cipher the message. Only Bob will be able to decipher it using his private key.

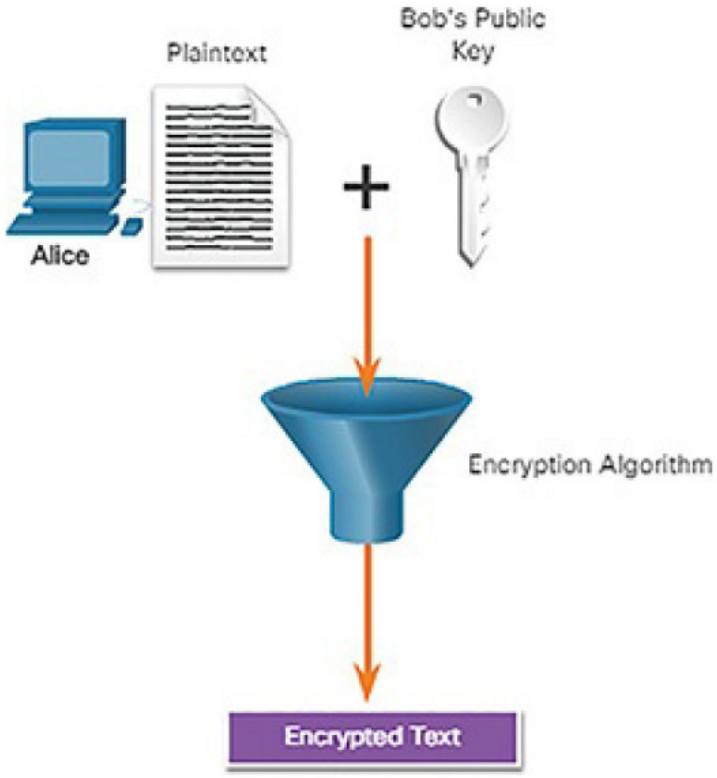


Figure 9-32 Alice Encrypts a Message Using Bob's Public Key

Alice also wants to ensure message authentication and integrity. Authentication ensures Bob that the document was sent by Alice, and integrity ensures that it was not modified. In Figure 9-33, Alice uses her private key to cipher a hash of the message. Alice sends the encrypted message with its encrypted hash to Bob.

In Figure 9-34, Bob uses Alice's public key to verify that the message was not modified. The received hash is equal to the locally determined hash based on Alice's public key. Additionally, this verifies that Alice is definitely the sender of the message because nobody else has Alice's private key.

Finally, in Figure 9-35, Bob uses his private key to decipher the message.

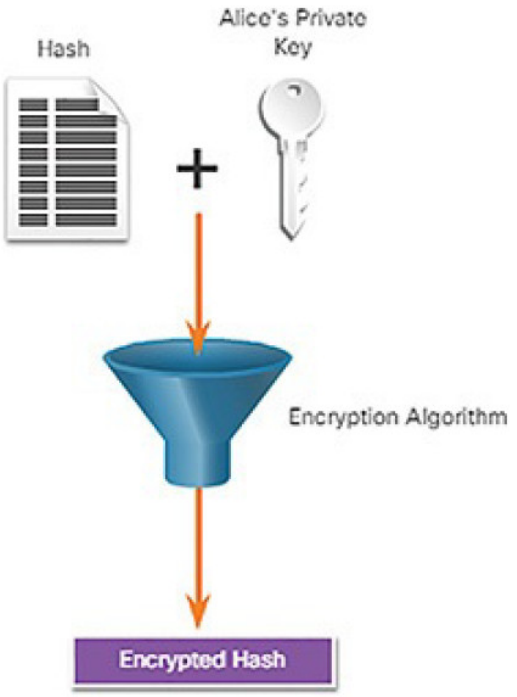


Figure 9-33 Alice Encrypts a Hash Using Alice's Private Key

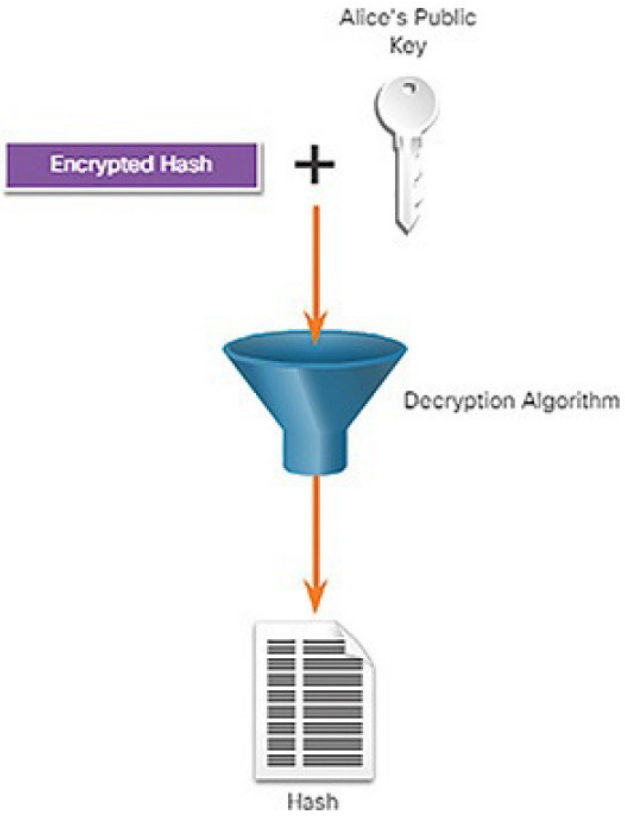


Figure 9-34 Bob Uses Alice's Public Key to Decrypt the Hash

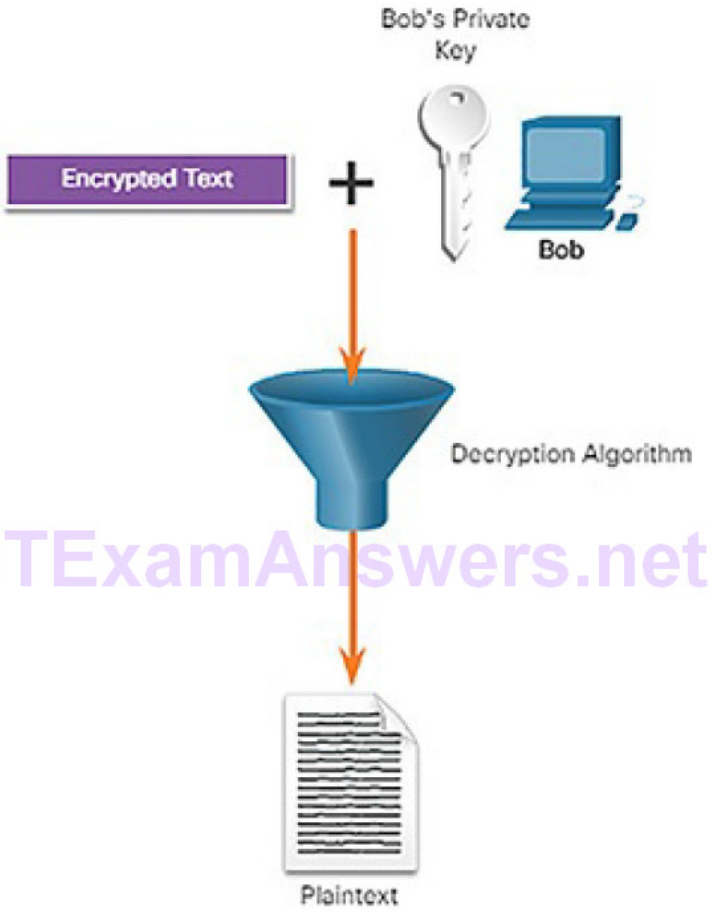


Figure 9-35 Bob Uses His Private Key to Decrypt Message

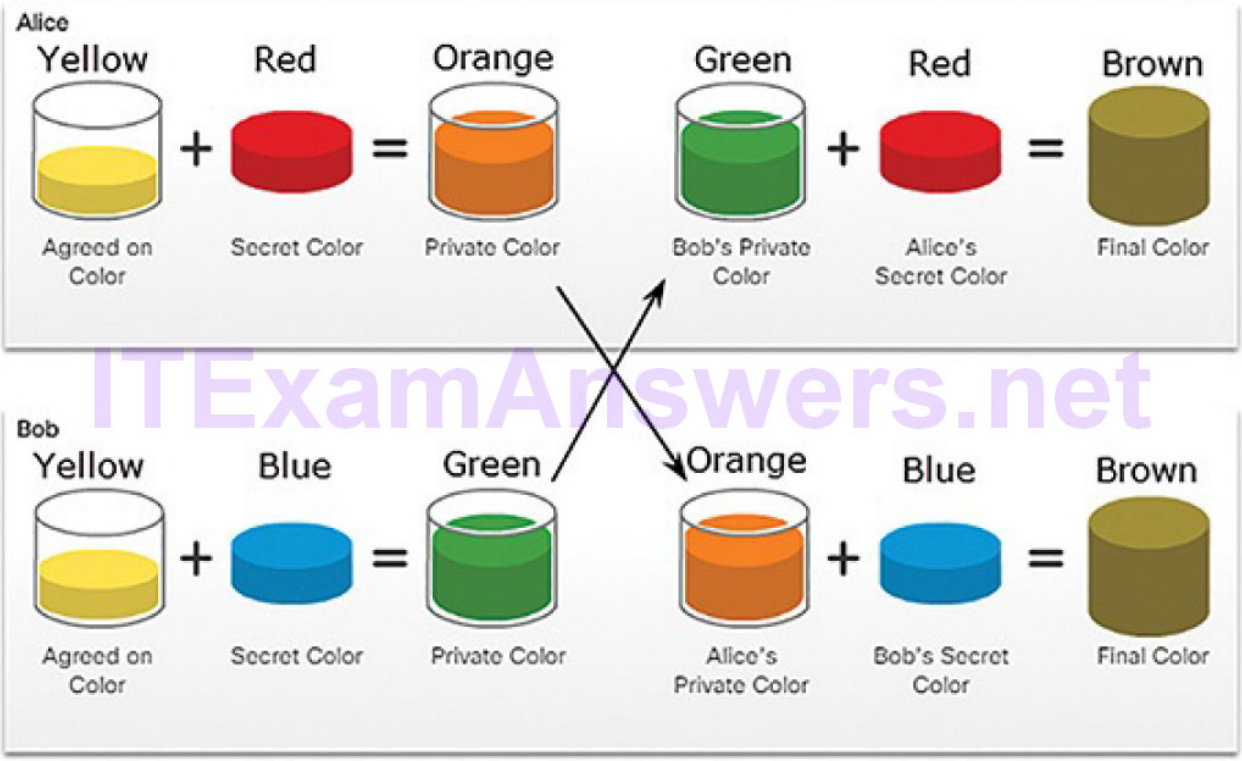
Diffie-Hellman (9.1.3.8)

Diffie-Hellman (DH) is an asymmetric mathematical algorithm that allows two computers to generate an identical shared secret without having communicated before. The new shared key is never actually exchanged between the sender and receiver. However, because both parties know it, the key can be used by an encryption algorithm to encrypt traffic between the two systems.

Here are three examples of instances when DH is commonly used:

- Data is exchanged using an IPsec VPN.
- Data is encrypted on the Internet using either SSL or TLS.
- SSH data is exchanged.

To help illustrate how DH operates, refer to Figure 9-36. Colors are used instead of complex long numbers to simplify the DH key agreement process. The DH key exchange begins with Alice and Bob agreeing on an arbitrary common color that does not need to be kept secret. The agreed on color in our example is yellow.



ITExamAnswers.net

Figure 9-36 DH Operation Simplified

Next, Alice and Bob will each select a secret color. Alice chose red while Bob chose blue. These secret colors will never be shared with anyone. The secret color represents the chosen secret private key of each party.

Alice and Bob now mix the shared common color (yellow) with their respective secret color to produce a private color. Therefore, Alice will mix the yellow with her red color to produce a private color of orange. Bob will mix the yellow and the blue to produce a private color of green.

Alice sends her private color (orange) to Bob and Bob sends his private color (green) to Alice.

Alice and Bob each mix the color they received with their own, original secret color (red for Alice and blue for Bob). The result is a final brown color mixture that is identical to the partner's final color mixture. The brown color represents the resulting shared secret key between Bob and Alice.

The security of DH is based on the fact that it uses unbelievably large numbers in its calculations. For example, a DH 1,024-bit number is roughly equal to a decimal number of 309 digits. Considering that 1 billion is 10 decimal digits (1,000,000,000), one can easily imagine the complexity of working with not one, but multiple 309-digit decimal numbers.

Unfortunately, asymmetric key systems are extremely slow for any sort of bulk encryption. This is why it is common to encrypt the bulk of the traffic using a symmetric algorithm, such as 3DES or AES, and use the DH algorithm to create keys that will be used by the encryption

algorithm.

Activity 9.1.3.9: Classify the Encryption Algorithms
Refer to the online course to complete this Activity.

Public Key Infrastructure (9.2)

In this section, you will learn how the public key infrastructure (PKI) supports network security.

Public Key Cryptography (9.2.1)

In this topic, you will learn about public key cryptography.

Using Digital Signatures (9.2.1.1)

Digital signatures are a mathematical technique used to provide three basic security services:

Authenticity: Provides authenticity of digitally signed data. Digital signatures authenticate a source, proving that a certain party has seen and signed the data in question.

Integrity: Provides integrity of digitally signed data. Digital signatures guarantee that the data has not changed from the time it was signed.

Non-repudiation: Provides non-repudiation of the transaction. The recipient can take the data to a third party, and the third party accepts the digital signature as a proof that this data exchange did take place. The signing party cannot repudiate that it has signed the data.

Digital signatures have the following properties that enable entity authentication and data integrity:

Signature is authentic: The signature is not forgeable and provides proof that the signer, and no one else, signed the document.

Signature is not reusable: The signature is part of the document and cannot be moved to a different document.

Signature is unaltered: After a document is signed, it cannot be altered.

Signature cannot be repudiated: For legal purposes, the signature and the document are considered physical things. Signers cannot claim later that they did not sign it.

Digital signatures are commonly used in the following two situations:

Code signing: This is used for data integrity and authentication purposes. Code signing is used to verify the integrity of executable files downloaded from a vendor website. It also uses signed digital certificates to authenticate and verify the identity of the site.

Digital certificates: These are similar to a virtual ID card and used to authenticate the identity of the system with a vendor website and establishan encrypted connection to exchange confidential data.

There are three Digital Signature Standard (DSS) algorithms that are used for generating and verifying digital signatures:

Digital Signature Algorithm (DSA): DSA is the original standard for generating public and private key pairs, and for generating and verifying digital signatures. Table 9-8 summarizes DSA.

Table 9-8 DSA Scorecard

DSA Characteristics	
Description	Digital Signature Algorithm (DSA)
Timeline	1994
Type of algorithm	Provides digital signatures
Advantages	Signature generation is fast
Disadvantages	Signature verification is slow

Rivest-Shamir-Adleman Algorithm (RSA): RSA is an asymmetric algorithm that is commonly used for generating and verifying digital signatures. Table 9-9 summarizes RSA.

RSA Characteristics	
Description	Ron Rivest, Adi Shamir, and Len Adleman
Timeline	1977
Type of algorithm	Asymmetric
Key size	512 to 2048 bits
Advantages	Signature verification is fast
Disadvantages	Signature generation is slow

Elliptic Curve Digital Signature Algorithm (ECDSA): ECDSA is a newer variant of DSA and provides digital signature authentication and non-repudiation with the added benefits of computational efficiency, small signature sizes, and minimal bandwidth.

Diffie-Hellman uses different DH groups to determine the strength of the key that is used in the key agreement process. The higher group numbers are more secure, but require additional time to compute the key. The following identifies the DH groups supported by Cisco IOS Software and their associated prime number value:

- DH Group 1: 768 bits
- DH Group 2: 1024 bits
- DH Group 5: 1536 bits
- DH Group 14: 2048 bits
- DH Group 15: 3072 bits
- DH Group 16: 4096 bits

Note
A DH key agreement can also be based on elliptic curve cryptography. DH groups 19, 20, and 24, based on elliptic curve cryptography, are also supported by Cisco IOS Software.

In the 1990s, RSA Security Inc. started to publish public key cryptography standards (PKCS). There were 15 PKCS, although 1 has been withdrawn as of the time of this writing. RSA published these standards because they had the patents to the standards and wished to promote them. PKCS are not industry standards, but are well recognized in the security industry and have recently begun to become relevant to standards organizations such as the IETF and PKIX working group.

Digital Signatures for Code Signing (9.2.1.2)

Digital signatures are commonly used to provide assurance of the authenticity and integrity of software code. Executable files are wrapped in a digitally signed envelope, which allows the end user to verify the signature before installing thesoftware.

Digitally signing code provides several assurances about the code:

- The code is authentic and is actually sourced by the publisher.
- The code has not been modified since it left the software publisher.
- The publisher undeniably published the code. This provides non-repudiation of the act of publishing.

The U.S. Government Federal Information Processing Standard (FIPS) Publication 140-3 specifies that software available for download on the Internet is to be digitally signed and verified. The purpose of digitally signed software is to ensure that the software has not been tampered with, and that it originated from the trusted source as claimed.

Refer to Figures 9-37 through 9-41 to see the properties of a file with a digitally signed certificate. Figure 9-37 displays the properties of a file that has been downloaded from the Internet. The intent of the file is to update the flash player on the host.

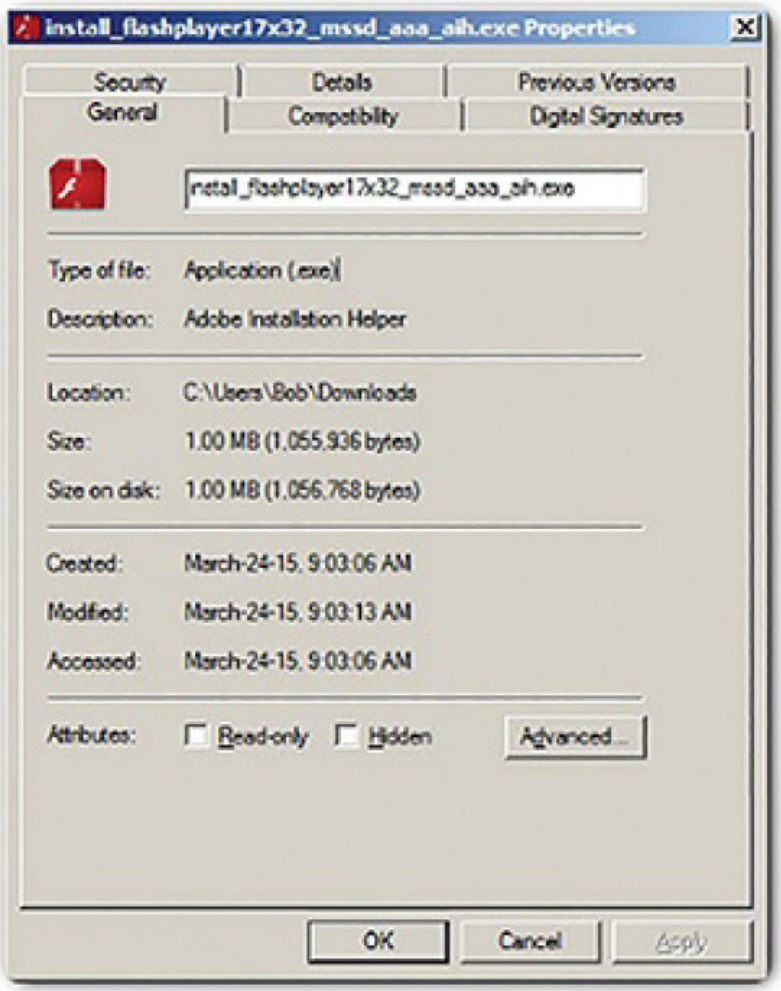


Figure 9-37 File Properties

Clicking the Digital Signature tab (Figure 9-38) reveals that the file is from a trusted organization, Adobe Systems.

Clicking the Details button opens the Digital Signature Details window (Figure 9-39), which reveals that the Symantec corporation is validating that the file has indeed originated from Adobe Systems Incorporated.

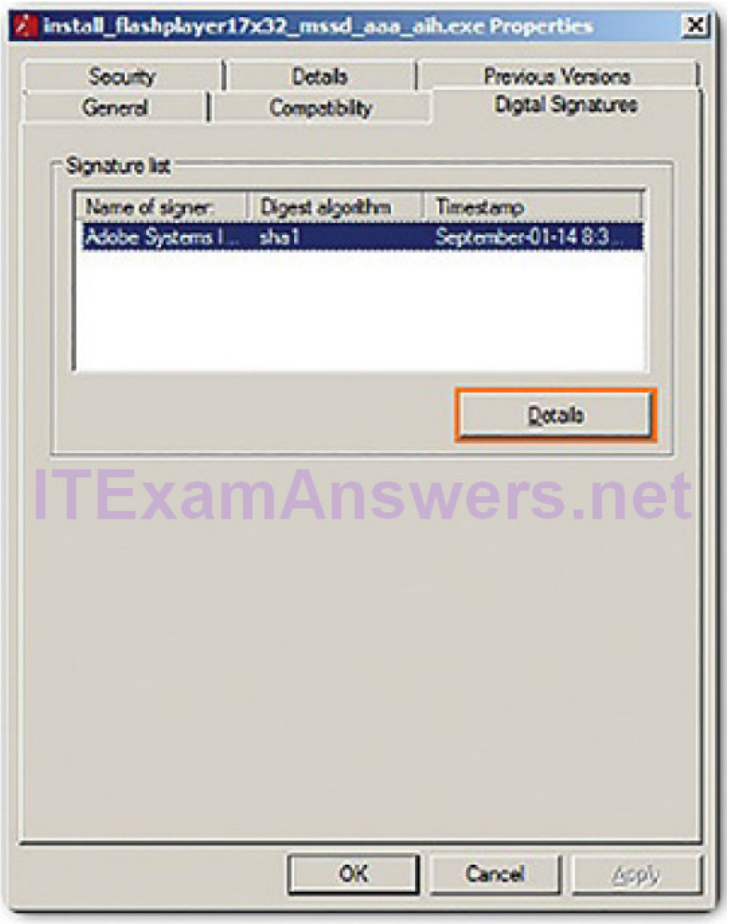


Figure 9-38 Digital Signatures Tab

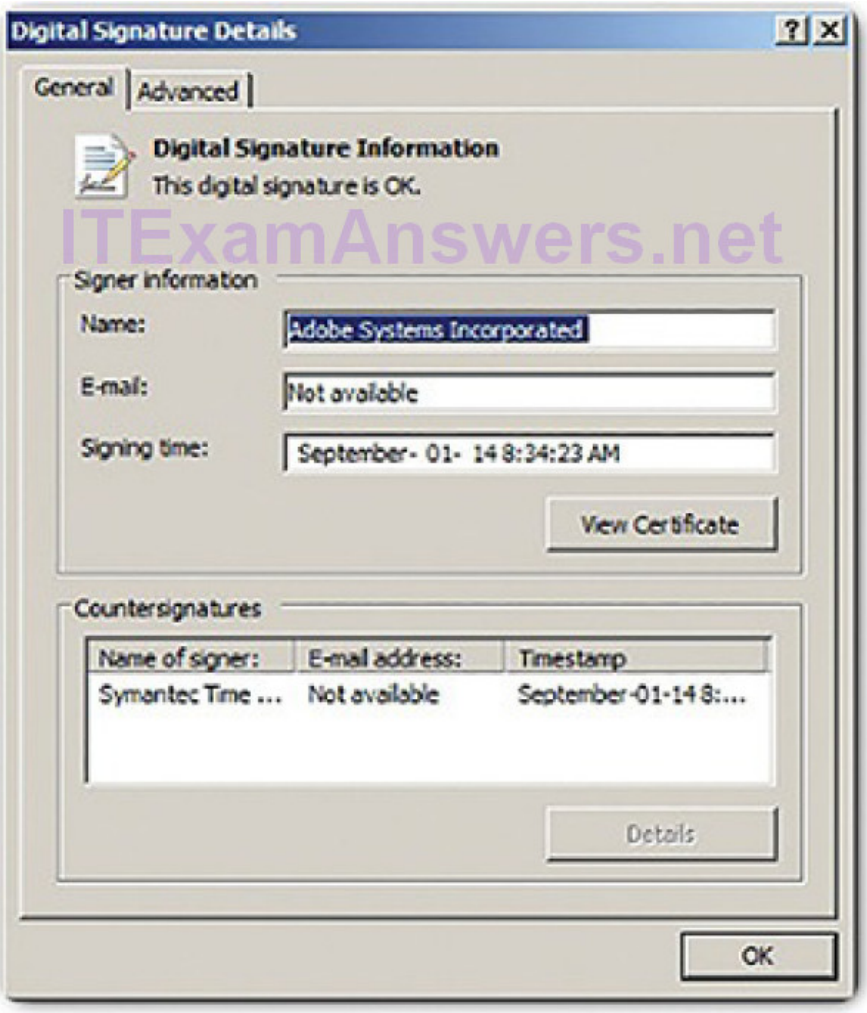


Figure 9-39 Digital Signature Details

Clicking the View Certificate button opens the details of the certificate (Figure 9-40). The certificate details display the purposes of the certificate, who it was issued to, and who is was issued by. It also displays the amount of time that this certificate is valid.

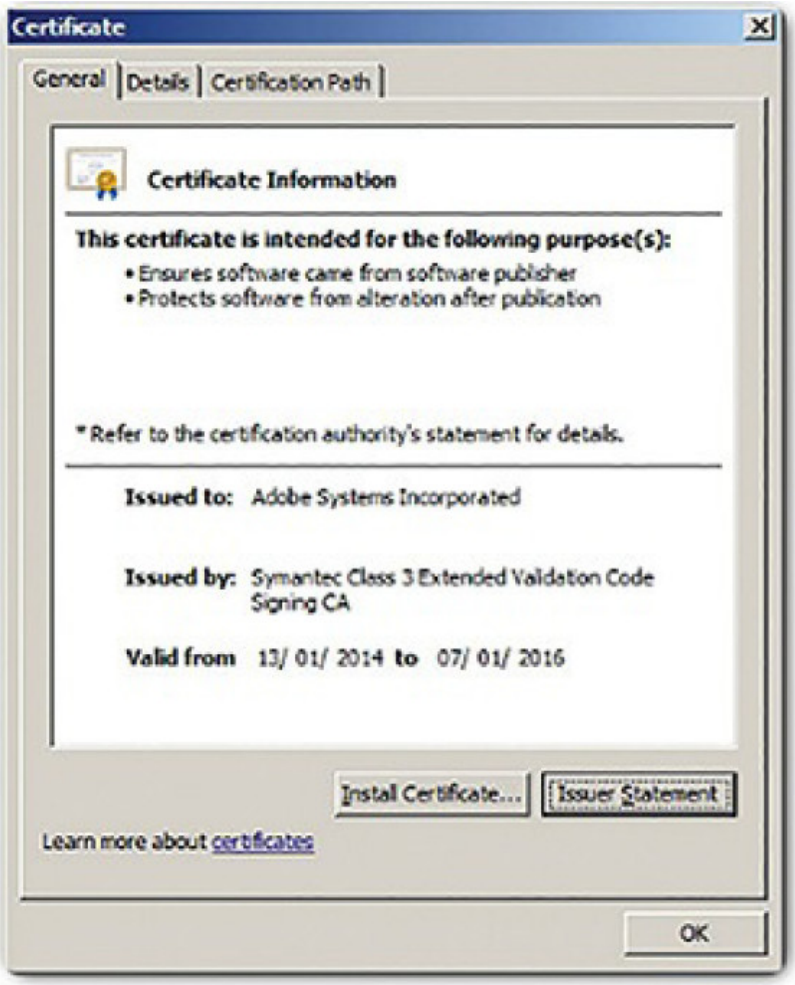


Figure 9-40 Digital Certificate Information

The Certificate Path tab (Figure 9-41) indicates that the file came from Adobe as authenticated by Symantec, which in turn is validated by VeriSign.

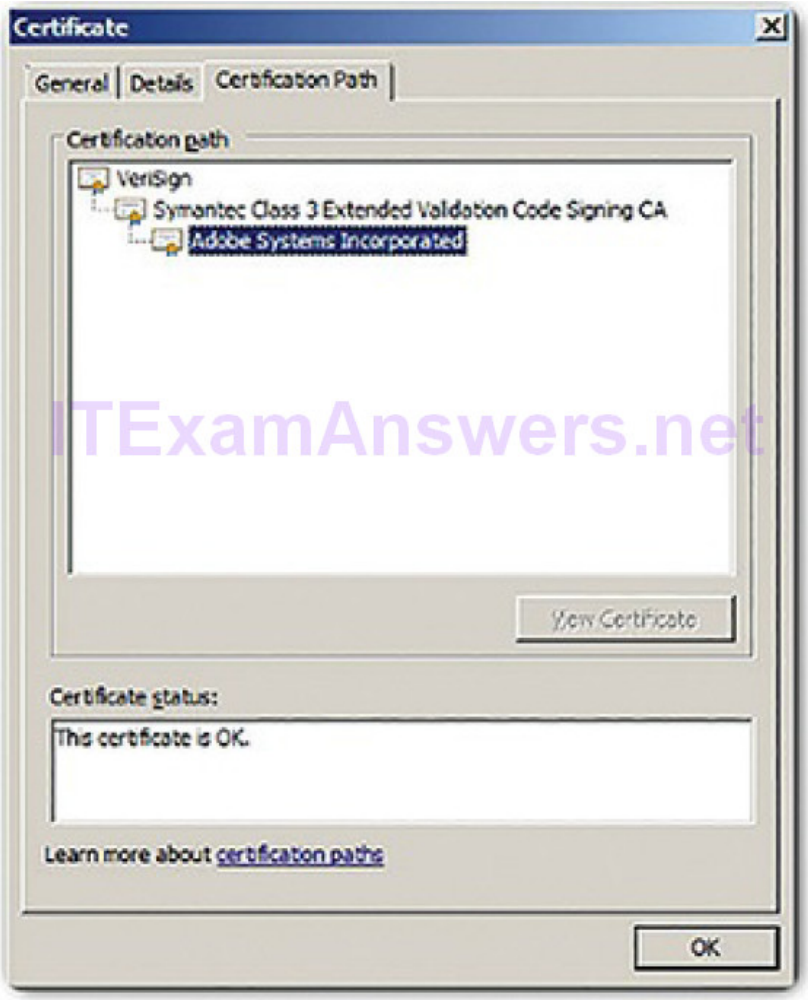


Figure 9-41 Digital Certificate Path

Digital Signatures for Digital Certificates (9.2.1.3)

A digital certificate is equivalent to an electronic passport. It enables users, hosts, and organizations to securely exchange information over the Internet. Specifically, a digital certificate is used to authenticate and verify that users sending a message are who they claim to be. Digital certificates can also be used to provide confidentiality for the receiver with the means to encrypt a reply.

Digital certificates are similar to physical certificates. For example, the paper-based Cisco Certified Network Associate Security (CCNA-S) certificate in Figure 9-42 identifies who the certificate is issued to, who authorized the certificate, and for how long the certificate is valid.

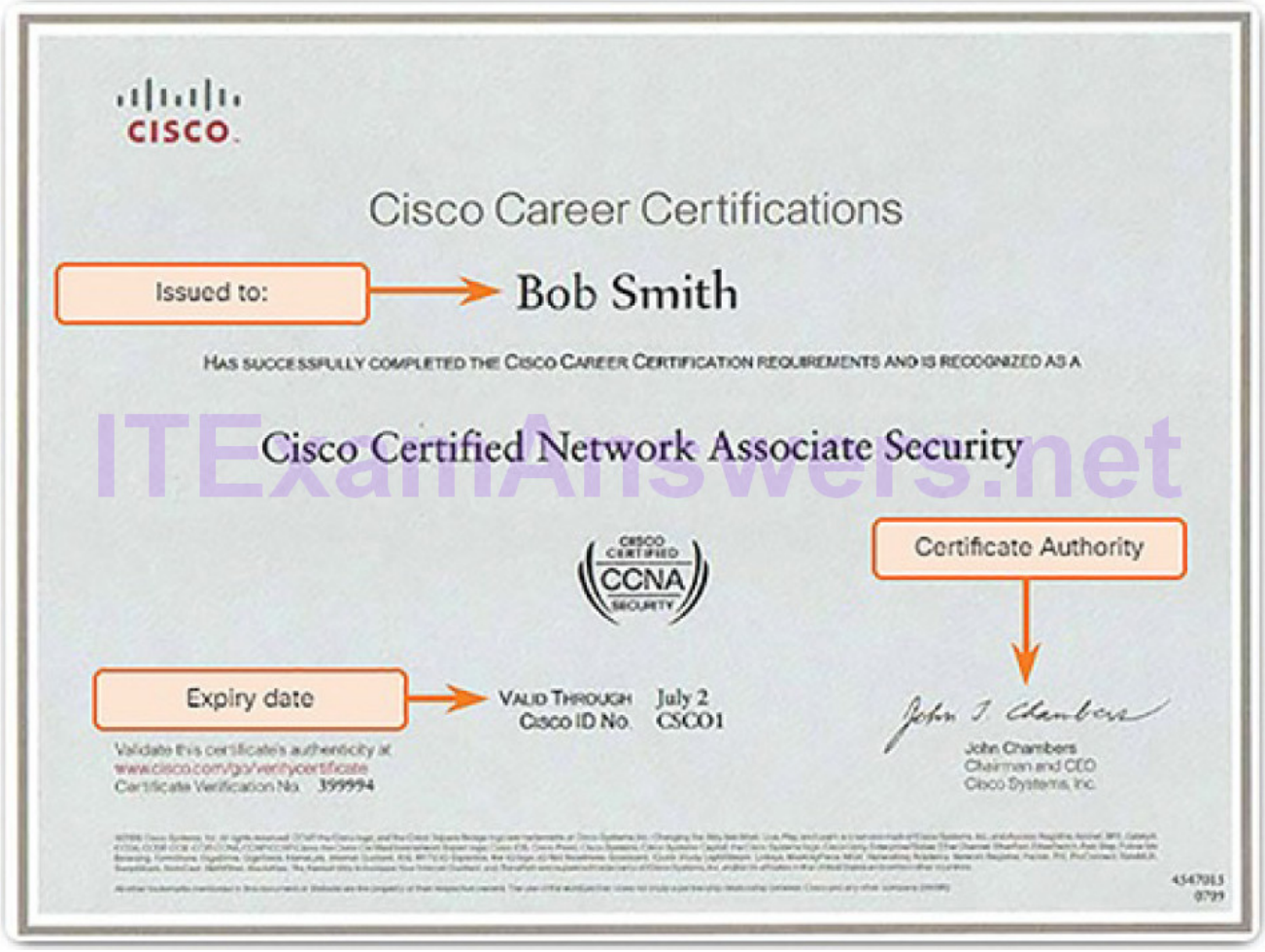
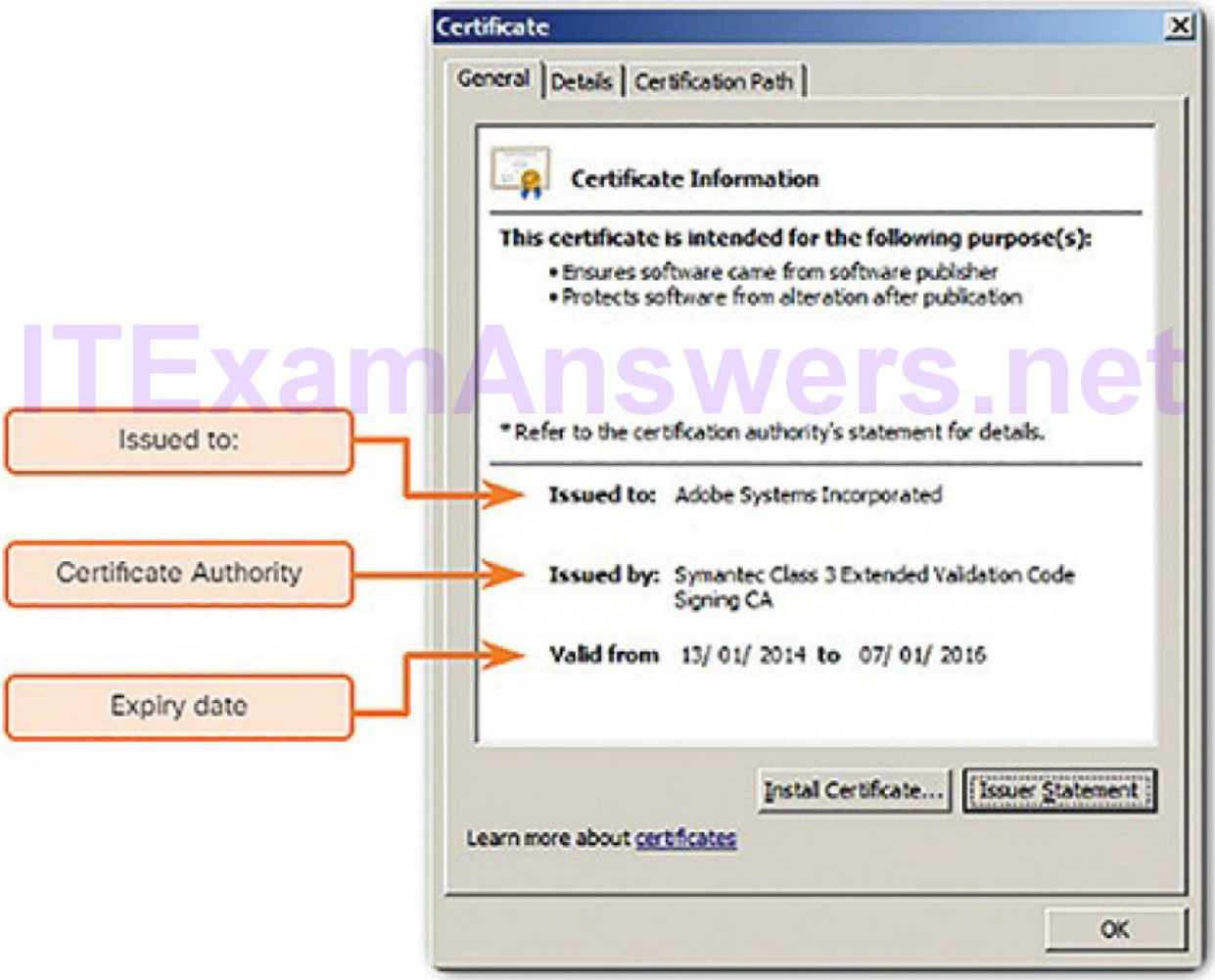


Figure 9-42 Physical CCNA Security Certificate

Notice how the digital certificate in Figure 9-43 also identifies similar elements.



ITExamAnswers.net

Figure 9-43 Digital Certificate Information

To help understand how a digital certificate is used, refer to Figure 9-44. In this scenario, Bob is confirming an order with Alice. The steps are as follows:

1. Bob confirms the order and his computer creates a hash of the confirmation.

2. The computer encrypts the hash with the private key of Bob.

3. The encrypted hash, known as the digital signature, is appended to the document. The order confirmation is sent to Alice over the Internet.

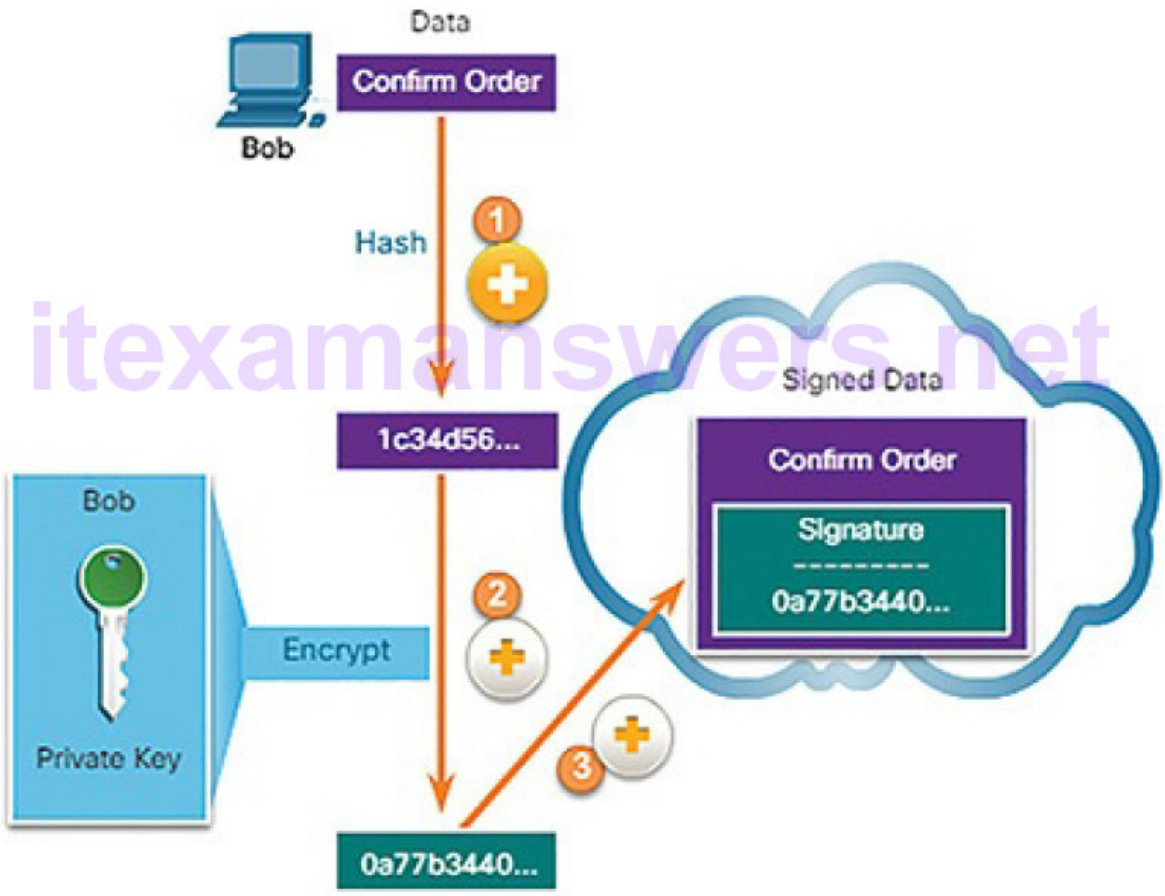


Figure 9-44 Sending a Digital Certificate

Figure 9-45 shows how Alice will use the digital certificate:

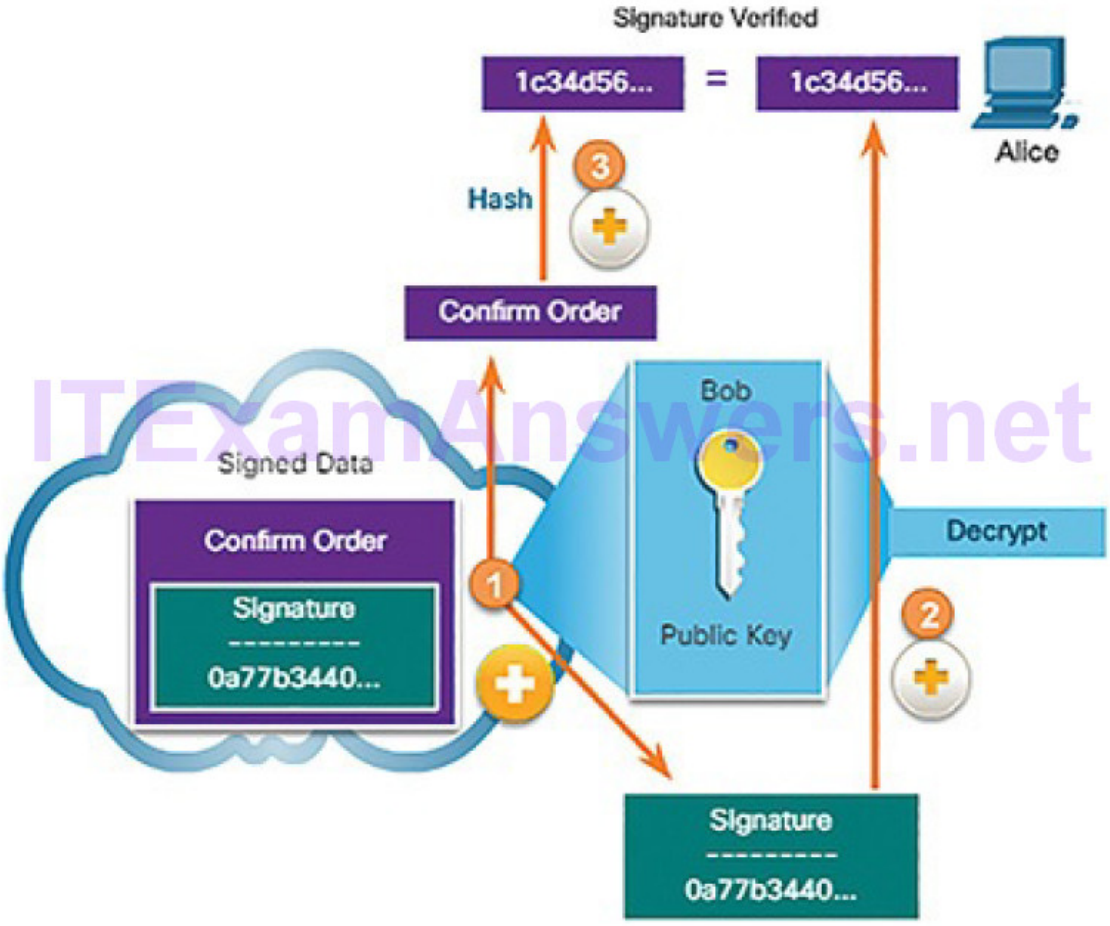


Figure 9-45 Receiving a Digital Certificate

1. Alice's receiving device accepts the order confirmation with the digital signature and obtains Bob's public key.

2. Her computer then decrypts the signature using Bob's public key. This step reveals the assumed hash value of the sending device.

3. Alice's computer creates a hash of the received document, without its signature, and compares this hash to the decrypted signature hash. If the hashes match, the document is authentic. This means it authenticates that it was sent by Bob and that it has not changed since it was signed.

Authorities and the PKI Trust System (9.2.2)

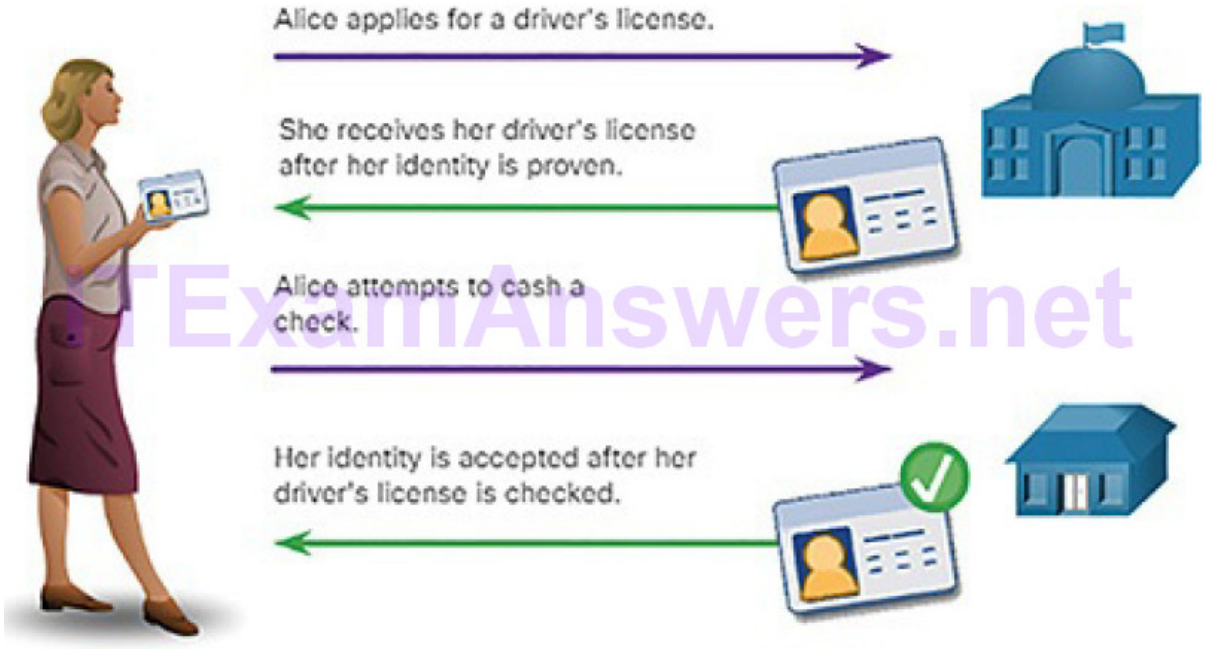
In this topic, you will learn how the public key infrastructure functions.

Public Key Management (9.2.2.1)

Internet traffic consists of traffic between two parties. When establishing an asymmetric connection between two hosts, the hosts will exchange their public key information. There are trusted third parties on the Internet that validate the authenticity of these public keys

using digital certificates. The trusted third party does an in-depth investigation prior to the issuance of credentials. After this in-depth investigation, the third party issues credentials (i.e., digital certificate) that are difficult to forge. From that point forward, all individuals who trust the third party simply accept the credentials that the third party issues.

These trusted third parties provide services similar to governmental licensing bureaus. Figure 9-46 illustrates how a driver's license is analogous to a digital certificate.



ITExamAnswers.net

Figure 9-46 Driver's License PKI Analogy

The Public Key Infrastructure (PKI) is an example of a trusted third-party system referred to as certificate authority (CA). The CA provides a service similar to governmental licensing bureaus. The PKI is the framework used to securely exchange information between parties. The CA issues digital certificates that authenticate the identity of organizations and users. These certificates are also used to sign messages to ensure that the messages have not been tampered with.

The Public Key Infrastructure (9.2.2.2)

PKI is needed to support large-scale distribution and identification of public encryption keys. The PKI framework facilitates a highly scalable trust relationship.

It consists of the hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates.

Figure 9-47 shows the main elements of the PKI:

Figure 9-47 Elements of the PKI Framework

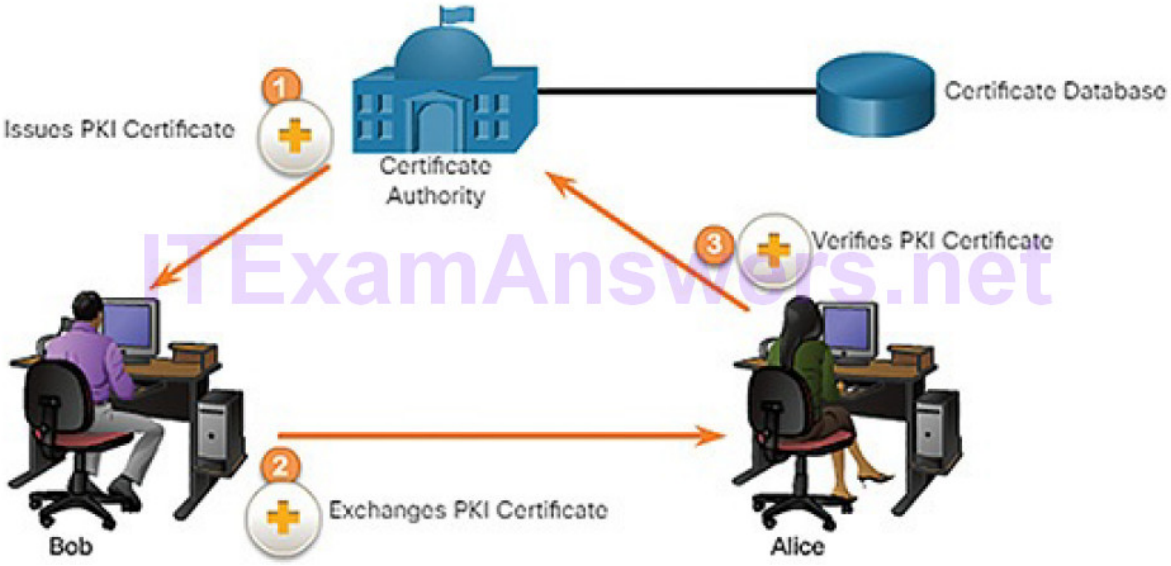
PKI certificate authority (CA): The CA is a trusted third party that issues PKI certificates to entities and individuals after verifying their identity. It signs these certificates using its private key.

Certificate database: The certificate database stores all certificates approved by the CA.

PKI certificate: Certificates contain an entity's or individual's public key, its purpose, the CA that validated and issued the certificate, the date range during which the certificate can be considered valid, and the algorithm used to create the signature.

Certificate store: The certificate store resides on a local computer and stores issued certificates and private keys.

Figure 9-48 shows an example of PKI, the steps of which are described as follows:



ITExamAnswers.net

Figure 9-48 PKI Example

1. Bob initially requests a certificate from the CA. The CA authenticates Bob and stores Bob's PKI certificate in the certificate database.

2. Bob communicates with Alice using his PKI certificate.

3. Alice communicates with the trusted CA using the CA's public key. The CA refers to the certificate database to validate Bob's PKI certificate.

Note
Not all PKI certificates are directly received from a CA. A registration authority (RA) is a subordinate CA and is certified by a root CA to issue certificates for specific uses.

The PKI Authorities System (9.2.2.3)

Many vendors provide CA servers as a managed service or as an end-user product. Some of these vendors include Symantec Group (VeriSign), Comodo, Go Daddy Group, GlobalSign, and DigiCert, among others.

Organizations may also implement private PKIs using Microsoft Server or Open SSL.

CAs, especially those that are outsourced, issue certificates based on classes which determine how trusted a certificate is.

Table 9-10 provides a description of the classes. The class number is determined by how rigorous the procedure was that verified the identity of the holder when the certificate was issued. The higher the class number, the more trusted the certificate. Therefore, a class 5 certificate is trusted much more than a lower class certificate.

Table 9-10 Classes of Certificates

Class	Description
0	Used for testing purposes in which no checks have been performed.
1	Used for individuals with a focus on verification of email.
2	Used for organizations for which proof of identity is required.
3	Used for servers and software signing for which independent verification and checking of identity and authority is done by the issuing certificate authority.
4	Used for online business transactions between companies.
5	Used for private organizations or governmental security.

For example, a class 1 certificate might require an email reply from the holder to confirm that they wish to enroll. This kind of confirmation is a weak authentication of the holder. For a class 3 or 4 certificate, the future holder must prove identity and authenticate the public key by showing up in person with at least two official ID documents.

Some CA public keys are preloaded, such as those listed in web browsers. Figure 9-49 displays various VeriSign certificates contained in the certificate store on the host. Any certificates signed by any of the CAs in the list will be seen by the browser as legitimate and will be trusted automatically.

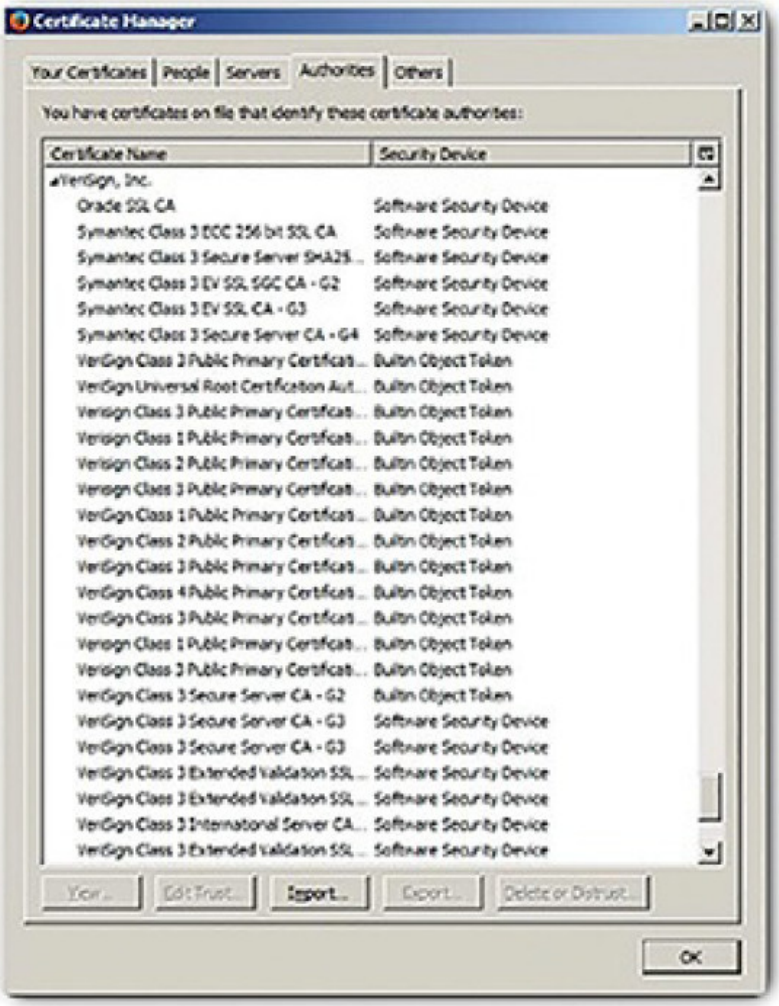


Figure 9-49 Sample VeriSign Certificates

Note
An enterprise can also implement PKI for internal use. PKI can be used to authenticate employees who are accessing the network. In this case, the enterprise is its own CA.

The PKI Trust System (9.2.2.4)

PKIs can form different topologies of trust. The simplest is the single-root PKI topology.

As shown in Figure 9-50, a single CA, called the root CA, issues all the certificates to the end users, which are usually within the same organization.

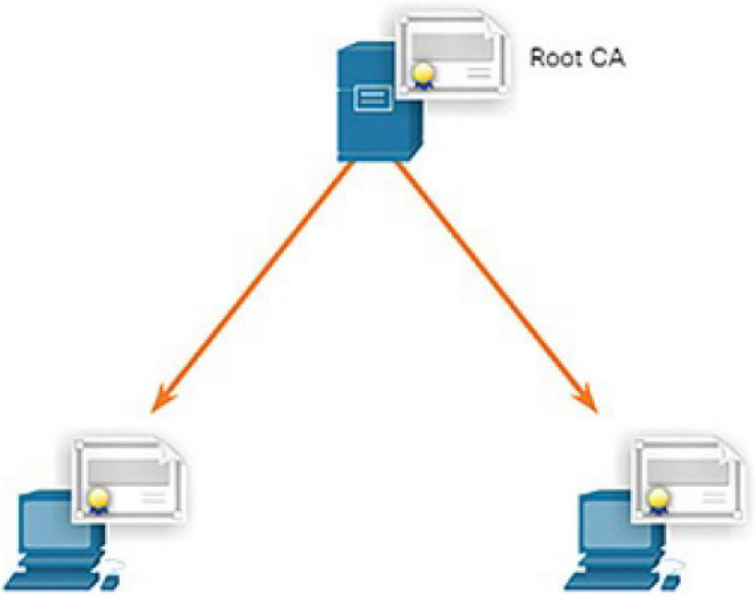
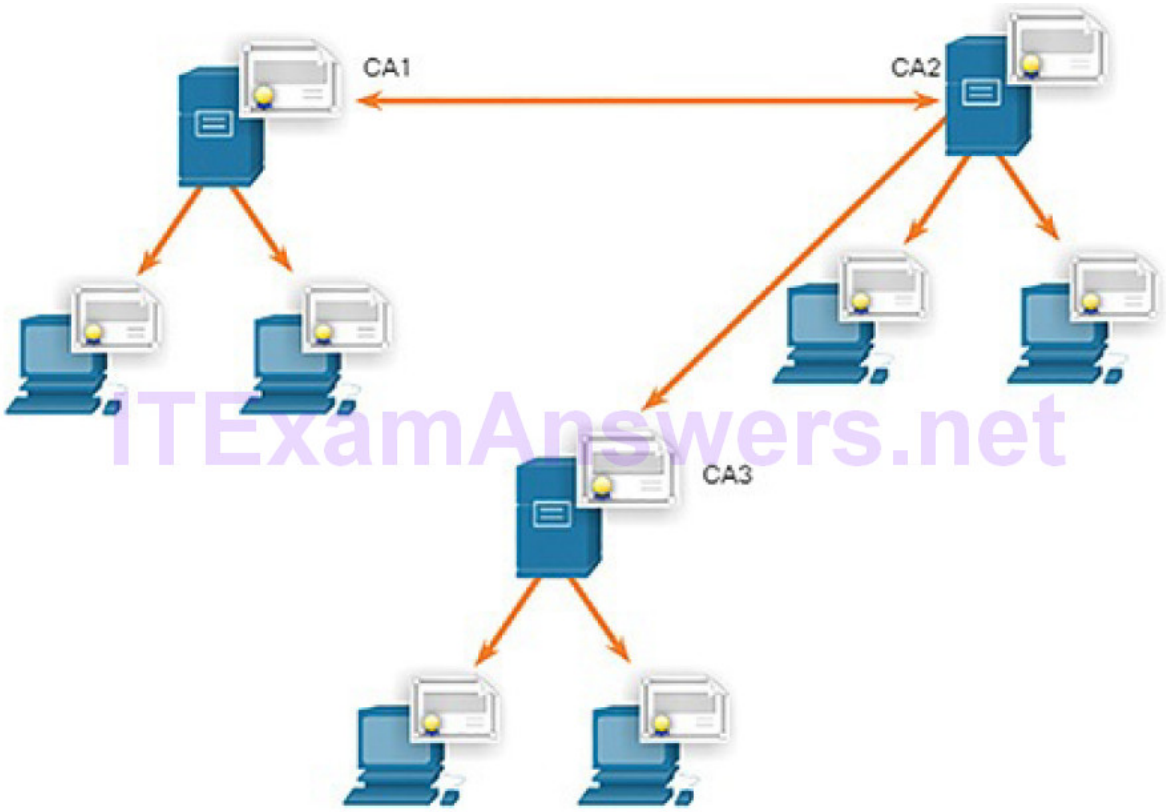


Figure 9-50 Single-Root PKI Topology

The benefit to this approach is its simplicity. However, it is difficult to scale to a large environment because it requires a strictly centralized administration, which creates a single point of failure.

On larger networks, PKI CAs may be linked using two basic architectures:

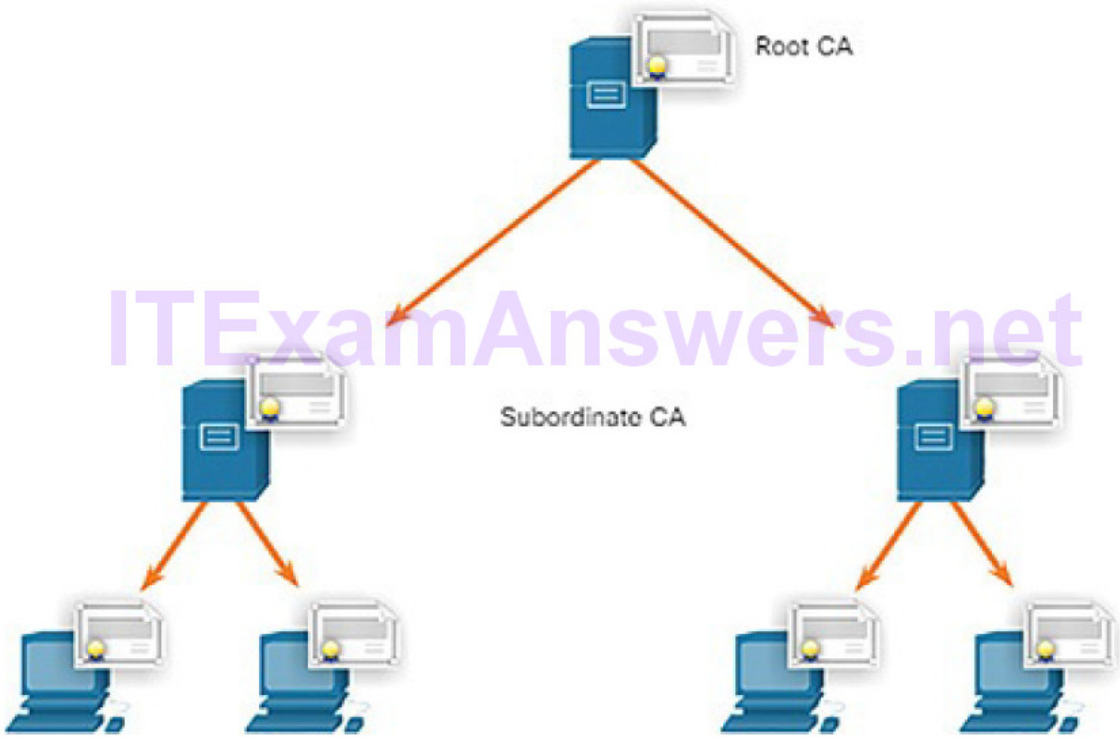
Cross-certified CA topologies: As shown in Figure 9-51, this is a peer-to-peer model in which individual CAs establish trust relationships with otherCAs by cross-certifying CA certificates. Users in either CA domain are also assured that they can trust each other. This provides redundancy and eliminates the single point of failure.



ITExamAnswers.net

Figure 9-51 Cross-Certified CA

Hierarchical CA topologies: As shown in Figure 9-52, the highest level CA is called the root CA. It can issue certificates to end users and to a subordinate CA. The sub-CAs could be created to support various business units, domains, or communities of trust. The root CA maintains the established “community of trust” by ensuring that each entity in the hierarchy conforms to a minimum set of practices. The benefits of this topology include increased scalability and manageability. This topology works well in most large organizations. However, it can be difficult to determine the chain of the signing process.



ITExamAnswers.net

Figure 9-52 Hierarchical CA

A hierarchical and cross-certification topology can be combined to create a hybrid infrastructure. An example would be when two hierarchical communities want to cross-certify each other in order for members of each community to trust each other.

Interoperability of Different PKI Vendors (9.2.2.5)

Interoperability between a PKI and its supporting services, such as Lightweight Directory Access Protocol (LDAP) and X.500 directories, is a concern because many CA vendors have proposed and implemented proprietary solutions instead of waiting for standards to develop.

Note
LDAP and X.500 are protocols that are used to query a directory service, such as Microsoft Active Directory, to verify a username and password.

To address this interoperability concern, the IETF published the Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework (RFC 2527). The X.509 version 3 (X.509v3) standard defines the format of a digital certificate.

As shown Figure 9-53, the X.509 format is already extensively used in theinfrastructure of the Internet.

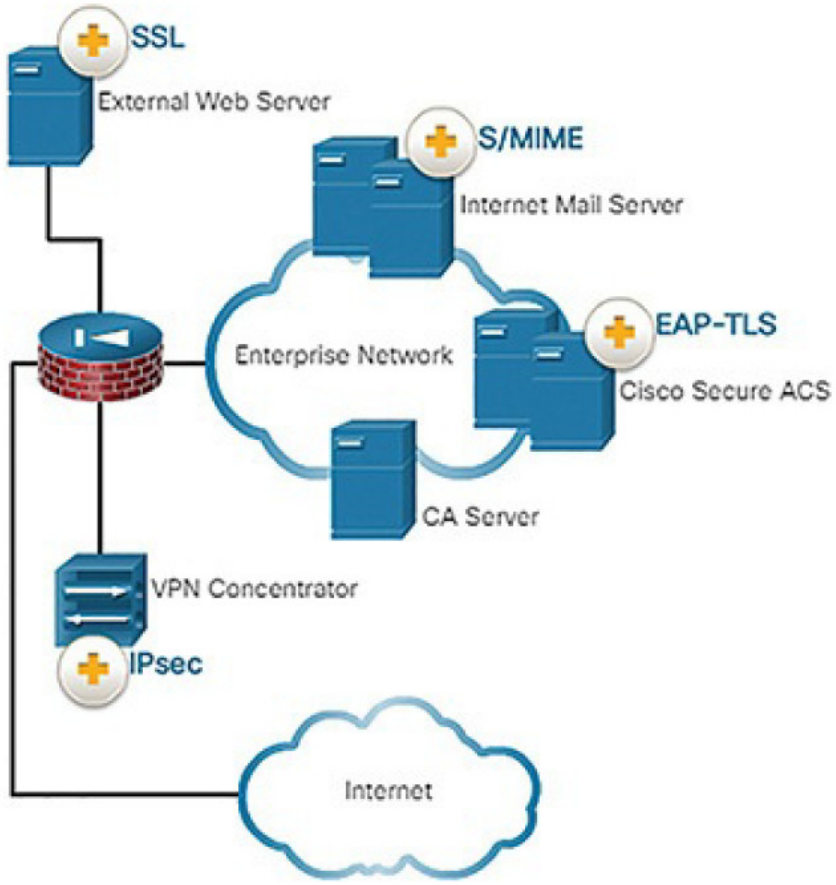


Figure 9-53 X.509v3 Applications

The applications in Figure 9-53 include the following:

SSL: Secure web servers use X.509v3 for website authentication in the SSL and TLS protocols, while web browsers use X.509v3 to implement HTTPS client certificates. SSL is the most widely used certificate-based authentication.

S/MIME: User mail agents that support mail protection using the Secure/ Multipurpose Internet Mail Extensions (S/MIME) protocol use X.509.

EAP-TLS: Cisco switches can use certificates to authenticate end devices connecting to LAN ports using 802.1X between the adjacent devices. The authentication can be proxied to a central ACS via the Extensible Authentication Protocol with TLS (EAP-TLS).

IPsec: IPsec VPNs where certificates can be used as a public key distribution mechanism for IKE RSA-based authentication use X.509.

Certificate Enrollment, Authentication, and Revocation (9.2.2.6)

In the CA authentication procedure, the first step when contacting the PKI is to securely obtain a copy of the CA's public key. All systems that leverage the PKI must have the CA's public key, called the self-signed certificate. The CA public key verifies all the certificates

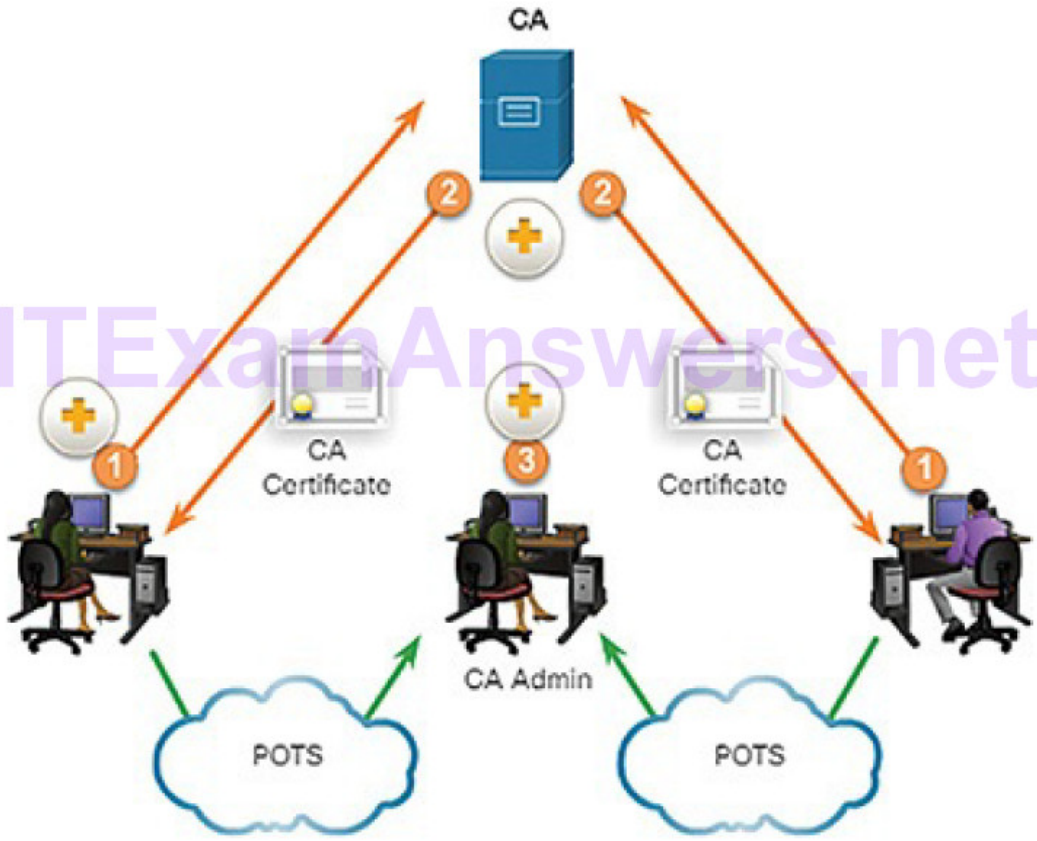
issued by the CA and is vital for the proper operation of the PKI.

Note
Only a root CA can issue a self-signed certificate.

For many systems such as web browsers, the distribution of CA certificates is handled automatically. The web browser comes preinstalled with a set of public CA root certificates. Organizations also push their private CA root certificate to clients through various software distribution methods.

The certificate enrollment process is used by a host system to enroll with a PKI. To do so, CA certificates are retrieved in-band over a network, and the authentication is done out-of-band (OOB) using the telephone. The system enrolling with the PKI contacts a CA to request and obtain a digital identity certificate for itself and to get the CA's self-signed certificate. The final stage verifies that the CA certificate was authentic and is performed using an OOB method such as the Plain Old Telephone System (POTS) to obtain the fingerprint of the valid CA identity certificate.

The first part of the certificate enrollment process consists of acquiring the CA's self-signed certificate.
Figure 9-54 demonstrates how CA certificates are retrieved, as described in the following steps:



ITExamAnswers.net

Figure 9-54 Retrieving CA Certificates

1. Alice and Bob request the CA certificate that contains the CA public key.

2. Upon receipt of the CA certificate, each requesting system verifies the validity of the certificate using public key cryptography.

3. Alice and Bob follow up the technical verification done by their system by telephoning the CA administrator and verifying the public key and serial number of the certificate.

After retrieving the CA certificate, Alice and Bob submit certificate requests to the CA, as shown in Figure 9-55 and described next:

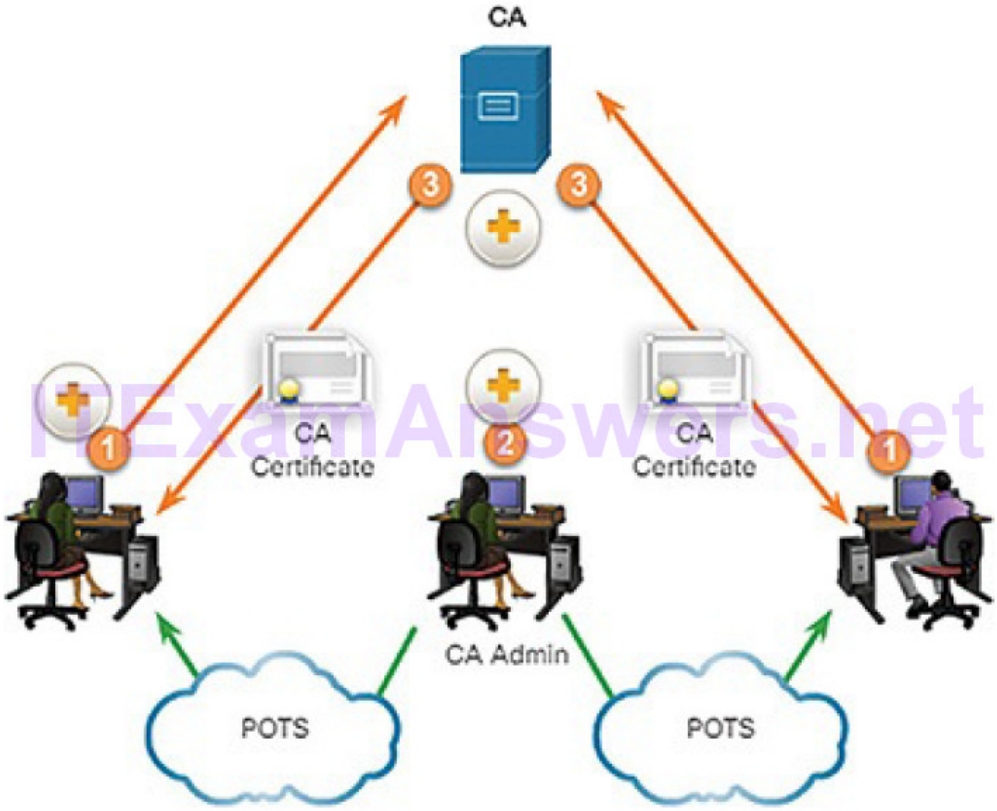


Figure 9-55 Submitting Certificate Requests to the CA

1. Both systems forward a certificate request that includes their public key along with some identifying information. All of this information is encrypted using the public key of the CA.

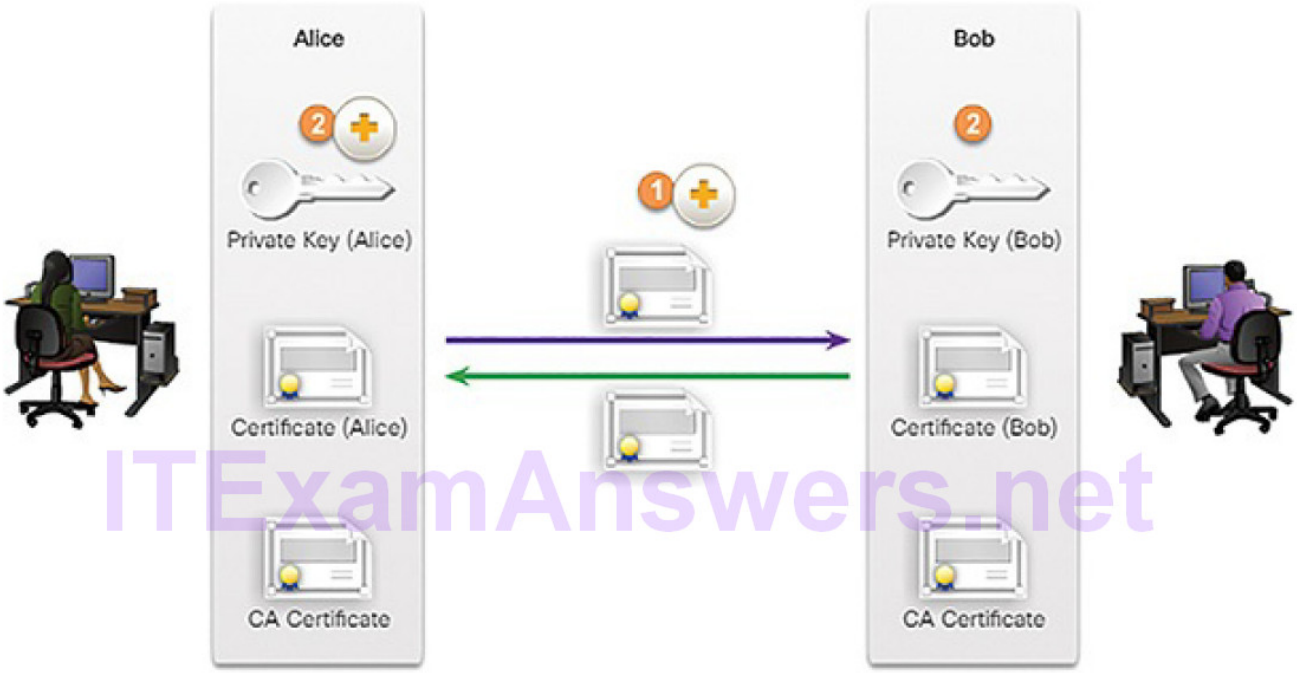
2. Upon the receipt of the certificate requests on the CA server, the CA administrator telephones Alice and Bob to confirm their submittal and the public key. The CA administrator issues the certificate by adding some additional data to the certificate request and digitally signing it all.

3. Either the end user manually retrieves the certificate or SCEP automatically retrieves the certificate, and the certificate is installed onto the system.

Having installed certificates signed by the same CA, Bob and Alice are now ready to authenticate each other, as shown in Figure 9-56 and described next:

1. Bob and Alice exchange certificates. The CA is no longer involved.

2. Each party verifies the digital signature on the certificate by hashing the plaintext portion of the certificate, decrypting the digital signature using the CA public key, and comparing the results. If the results match, the certificate is verified as being signed by a trusted third party and the verification by the CA that Bob is Bob and Alice is Alice is accepted.



ITExamAnswers.net

Figure 9-56 Peers Authenticate Each Other

Authentication no longer requires the presence of the CA server, and each user exchanges their certificates containing public keys.

Certificate must sometimes be revoked. For example, a digital certificate can be revoked if a key is compromised or if it is no longer needed.

Here are two of the most common methods of revocation:

Certificate revocation list (CRL): A list of revoked certificate serial numbers that have been invalidated because they expired. PKI entities regularly poll the CRL repository to receive the current CRL.

Online Certificate Status Protocol (OCSP): An Internet protocol used to query an OCSP server for the revocation status of an X.509 digital certificate. Revocation information is immediately pushed to an online database.

Lab 9.2.2.7: Certificate Authority Stores

In this lab, you will complete the following objectives:

- Certificates Trusted by Your Browser
- Checking for Man-In-Middle

Applications and Impacts of Cryptography (9.2.3)

In this topic, you will learn how the use of cryptography affects cybersecurity operations.

PKI Applications (9.2.3.1)

Where can PKI be used by an enterprise? The following provides a short list of common use of PKIs:

- SSL/TLS certificate-based peer authentication
- Secure network traffic using IPsec VPNs
- Secure HTTPS web traffic
- Control access to the network using 802.1X authentication
- Secure email using the S/MIME protocol
- Secure instant messaging
- Approve and authorize applications with code signing
- Protect user data with the Encrypting File System (EFS)
- Implement two-factor authentication with smart cards
- Secure USB storage devices

Encrypting Network Transactions (9.2.3.2)

A security analysis must be able to recognize and solve potential problems related to permitting PKI-related solutions on the enterprise network.

Consider how the increase of SSL/TLS traffic poses a major security risk to enterprises because the traffic is encrypted and cannot be intercepted and monitored by normal means. Users can introduce malware or leak confidential information over an SSL/TLS connection.

Threat actors can use SSL/TLS to introduce regulatory compliance violations, viruses, malware, data loss, and intrusion attempts in a network.

Other SSL/TLS-related issues may be associated with validating the certificate of a web server. When this occurs, web browsers will display a security warning.PKI-related issues that are associated with security warnings include

Validity date range: The X.509v3 certificates specify “not before” and “not after” dates. If the current date is outside the range, the web browser displays a message. Expired certificates may simply be the result of administrator oversight, but they may also reflect more serious conditions.

Signature validation error: If a browser cannot validate the signature on the certificate, there is no assurance that the public key in the certificate is authentic. Signature validation will fail if the root certificate of the CA hierarchy is not available in the browser's certificate store.

Figure 9-57 displays an example of a signature validation error with the Cisco AnyConnect Secure Mobility Client.

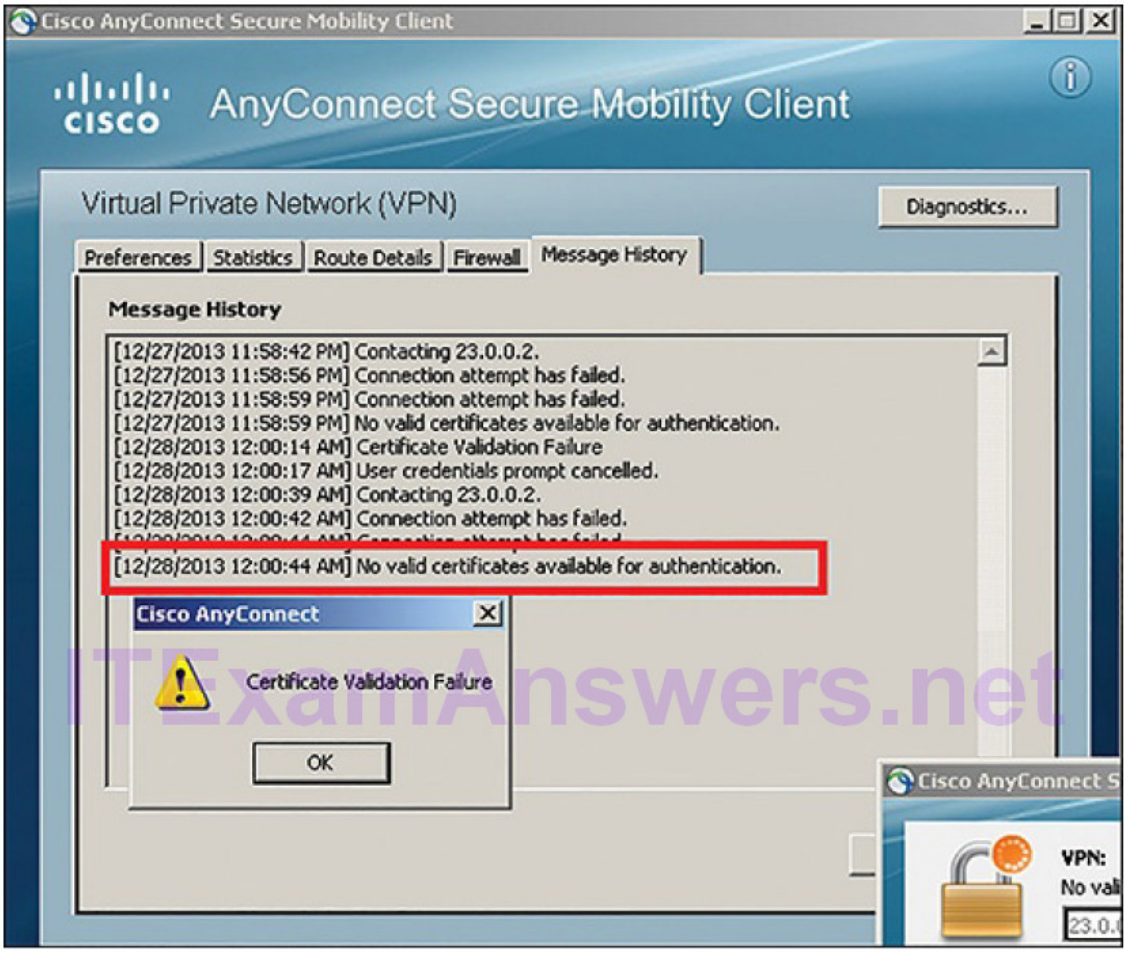


Figure 9-57 Signature Validation Error

Some of these issues can be avoided due to the fact that the SSL/TLS protocols are extensible and modular. This is known as a cipher suite. The key components of the cipher suite are the message authentication code (MAC) algorithm, the encryption algorithm, the key exchange algorithm, and the authentication algorithm. These can be changed without replacing the entire protocol. This is very helpful because the different algorithms continue to evolve. As cryptanalysis continues to reveal flaws in these algorithms, the cipher suite can be updated to patch these flaws. When the protocol versions within the cipher suite change, the version number of SSL/TLS changes as well.

Encryption and Security Monitoring (9.2.3.3)

Network monitoring becomes more challenging when packets are encrypted. However, security analysts must be aware of those challenges and address them as best as possible. For instance, when site-to-site VPNs are used, the IPS should be positioned so it can monitor unencrypted traffic.

However, the increase of HTTPS in the enterprise network introduces new challenges. Since HTTPS introduces end-to-end encrypted HTTP traffic (via TLS/SSL), it is not as easy to peek into user traffic.

Security analysts must know how to circumvent and solve these issues. Here is a list of some of the things that a security analyst could do:

- Configure rules to distinguish between SSL and non-SSL traffic, HTTPS and non-HTTPS SSL traffic.
- Enhance security through server certificate validation using CRLs and OCSP.
- Implement antimalware protection and URL filtering of HTTPS content.
- Deploy a Cisco SSL Appliance (Figure 9-58) to decrypt SSL traffic and send it to IPS appliances to identify risks normally hidden by SSL.

Figure 9-58 Cisco SSL Appliance

Cryptography is dynamic and always changing. A security analyst must maintain a good understanding of cryptographic algorithms and operations to be able to investigate cryptography-related security incidents.

There are two main ways in which cryptography impacts security investigations. First, attacks can be directed to specifically target the encryption algorithms themselves. After the algorithm has been cracked and the attacker has obtained the keys, any encrypted data that has been captured can be decrypted by the attacker and read, thus exposing private data. The security investigation is also affected because data can be hidden in plain sight by encrypting it. For example, command and control traffic that is encrypted with TLS/SSL most likely cannot be seen by a firewall. The command and control traffic between a command and control server and an infected computer in a secure network cannot be stopped if it cannot be seen and understood. The attacker would be able to continue using encrypted commands to

infect more computers and possibly create a botnet. This type of traffic can be detected by decrypting the traffic and comparing it with known attack signatures, or by detecting anomalous TLS/SSL traffic. This is either very difficult and time consuming or a hit-or-miss process.

Summary (9.3)

In this chapter, you learned about the various cryptographic techniques and how their use affects network security monitoring. Securing communications with cryptography consists of four elements:

- Data confidentiality to guarantee that only authorized users can read themessage
- Data integrity to guarantee that the message was not altered
- Origin authentication to guarantee that the message is not a forgery and does actually come from whom it states
- Data non-repudiation to guarantee that the sender cannot repudiate, or refute, the validity of a message sent

One of the main ways to verify and ensure data integrity is through the use of hash functions. Hash functions make it computationally infeasible for two different sets of data to come up with the same hash output. Three well-known hash functions include a;

- MD5 with a 128-bit digest
- SHA-1
- SHA-2

To include authentication along with message integrity, an HMAC is added as an input to a hash function. If two parties share a secret key and use HMAC functions for authentication, a properly constructed HMAC digest of a message that a party has received indicates that the other party was the originator of the message.

Confidentiality of the data is ensured through one of two types of encryption: symmetric or asymmetric.
Symmetric encryption uses the same key to encrypt and decrypt data. Well-known symmetric encryption algorithms include a;

- Data Encryption Standard (DES)
- 3DES (Triple DES)
- Advanced Encryption Standard (AES)
- Software-Optimized Encryption Algorithm (SEAL)
- Rivest Ciphers (RC)

Asymmetric encryption uses different keys to encrypt and decrypt data. Examples of protocols that use asymmetric key algorithms include:

- Internet Key Exchange (IKE)
- Secure Sockets Layer (SSL)
- Secure Shell (SSH)
- Pretty Good Privacy (PGP)

Diffie-Hellman (DH) is an asymmetric mathematical algorithm that allows two computers to generate an identical shared secret key without having communicated before. The new shared secret key is never actually exchanged between the sender and receiver.

The Public Key Infrastructure (PKI) relies on digital certificates. Digital certificates provide a digital signature that authenticates the source, guarantees the integrity of the data, and provides non-repudiation for the transaction. The three Digital Signature Standard (DSS) algorithms are:

- Digital Signature Algorithm (DSA)
- Rivest-Shamir-Adleman Algorithm (RSA)
- Elliptic Curve Digital Signature Algorithm (ECDSA)

Trusted third-party organizations such as VeriSign validate the authenticity of digital certificates. Digital certificates are used in a variety of PKI applications, including:

- IPsec VPNs
- HTTPS traffic
- 802.1X authentication
- Email and instant message security
- Code signing
- Encrypting File System (EFS)
- Two-factor authentication
- USB device security

Practice

The following activities provide practice with the topics introduced in this chapter. The Labs and Class Activities are available in the companion CCNA Cybersecurity Operations Lab Manual (ISBN: 9781587134388).

Class Activities

Class Activity 9.0.1.2: Creating Codes

Labs

Lab 9.1.1.6: Encrypting and Decrypting Data Using OpenSSL
Lab 9.1.1.7: Encrypting and Decrypting Data Using a Hacker Tool
Lab 9.1.1.8: Examining Telnet and SSH in Wireshark
Lab 9.1.2.5: Hashing Things Out
Lab 9.2.2.7: Certificate Authority Stores