15.1.6 Asymmetric Encryption Facts

Asymmetric encryption, also known as public key encryption, uses two keys that are mathematically related. Both keys together are called the key pair.

This lesson covers the following topics:

- Asymmetric encryption features
- Encryption algorithms
- Cryptography tools

Asymmetric Encryption Features

The following table highlights asymmetric encryption features:

Consideration	Description
Functionality	Asymmetric encryption functions in the following manner: ▪ The public key is made available to anyone; the private key is kept secret. ▪ One key encrypts, and the other key decrypts. For example, if data is encrypted with the public key, the private key is used to decrypt the data. ▪ The strength of an asymmetric encryption system lies in the secrecy and security of its private keys. If a private key is discovered, a new key pair must be generated. ▪ Both private and public keys are created on a local machine by a Local Security Authority (the security kernel) and a Cryptographic Service Provider (CSP). ▪ Asymmetric key ciphers are two associated algorithms that are inverses of each other. Both of the algorithms are easy to compute. ▪ It is computationally infeasible to derive the second algorithm from the first without the private key.
Uses	Asymmetric key encryption can provide confidentiality (encryption), strong authentication, and non-repudiation. Asymmetric encryption is used for: ▪ Data encryption to secure data. ▪ Digital signing to confirm the integrity of the message and the authenticity of the sender. ▪ Key exchange to ensure keys are secure during transit. Asymmetric encryption is often used to securely exchange symmetric keys.
Hybrid Cryptography	Operating systems, applications, and other components of information systems typically use a hybrid cryptography system. A hybrid cryptography system combines the strengths of both the symmetric and asymmetric cryptography systems. This means it combines symmetric systems to process large amounts of data relatively fast and asymmetric systems to securely distribute keys. A hybrid cryptography system works as follows: 1. A plain text message is encrypted into ciphertext with a symmetric session key. 2. The symmetric session key is then encrypted with asymmetric cryptography using the public key of the recipient. 3. The encrypted symmetric session key and the ciphertext are sent to the recipient. 4. The recipient decrypts the symmetric session key with asymmetric cryptography using the recipient's private key. 5. The ciphertext is then decrypted into plain text with the decrypted session key.
Implementations	Asymmetric encryption is used with the following protocols: ▪ SSL/TLS ▪ IPsec ▪ VPNs (PPTP, L2TP, SSTP) ▪ S/MIME and PGP for email security ▪ SSH tunnels

Management Considerations	Management considerations in implementing asymmetric key cryptography include the following:
	▪ Keys can be distributed with no prior relationship required. Only the public key needs to be distributed. It does not matter who has access to the public key. The private key is always kept secure and private.
	▪ Asymmetric cryptography is scalable for use in very large, expanding environments where data is frequently exchanged between different communication partners. The number of keys required is two per user.
	▪ Key space typically starts around 1,000 bits and goes as high as 32,000 bits.
	▪ Processing speeds are much slower (about 1000 times slower) than symmetric key cryptography.
	▪ Two mechanisms determine how long a cryptographic key remains in use:
	  ▪ Ephemeral keys are generated every time the key establishment process is executed and exists only for the lifetime of a specific communication session. As such, these keys have a relatively short lifespan.
	  ▪ Static keys can be reused by multiple communication sessions. As such, these keys remain in use for a relatively long period of time.

Encryption Algorithms

All asymmetric cryptographic algorithms are based on a trapdoor function that creates a value that is easy to produce in one direction, but difficult to reverse. The following table describes common asymmetric key cryptography algorithms.

System	Characteristics
Challenge-Handshake Authentication Protocol (CHAP)	The *Challenge-Handshake Authentication Protocol* (CHAP) is a protocol that uses a challenge/response (three-way handshake) mechanism to protect passwords. CHAP provides username and password authentication. CHAP is the only remote access authentication protocol that ensures that the same client or system exists throughout a communication session by repeatedly and randomly re-testing the validated system.
Diffie-Hellman Key Exchange	The *Diffie-Hellman Key Exchange* is a key agreement protocol that generates symmetric keys simultaneously at sender and recipient sites over unsecure channels. The Diffie-Hellman Key Exchange was the first asymmetric algorithm. It was developed by Whitfield Diffie and Martin Hellman in 1976. The Diffie-Hellman key exchange: ▪ Provides for key distribution and does not provide any cryptographic services. ▪ Is based on calculating discreet logarithms in a finite field. ▪ Is used in many algorithms and standards such as DES. ▪ Is subject to man-in-the-middle attacks and requires strong authentication to validate the end points. The Diffie-Hellman model uses a formula with the following values to create the public and private keys as follows: ▪ Y = large number < P (a typical large number is 301 digits). ▪ P = large prime number (a prime number is a whole number greater than or equal to one that has only two natural divisors, one and the number itself). ▪ mod = modulus (the remainder resulting from dividing two numbers). Diffie-Hellman does not use authentication; however, many other authenticated protocols use it as a base. In ephemeral mode (referred to as EDH or DHE), it provides TLS with perfect forward secrecy.
Digital Signature Algorithm (DSA)	The *Digital Signature Algorithm* (DSA) is a digital signature algorithm. DSA was adopted in 1994 as FIPS 186 by the National Institute of Standards and Technology (NIST). DSA signs messages using the signer's private key and the signatures are verified by the signer's corresponding public key. The digital signature algorithm provides message authentication, integrity, and non-repudiation.
Elliptic Curve Cryptography (ECC)	*Elliptic curve cryptography* (ECC) is a public-key cryptography method based on groups of numbers in an elliptical curve. The use of elliptic curves in cryptography was suggested independently by both Neal Koblitz and Victor S. Miller in 1985. Elliptic curve: ▪ Is a more efficient algorithm than other algorithms. ▪ Is used in conjunction with other methods to reduce the key size. ▪ Is suitable for small amounts of data for small devices, such as smart phones and PDAs. ▪ Produces a key of 160 bits that is equivalent to 1024-bit RSA key. The 160-bit key reduces computational power and memory requirements. Elliptic curve Diffie-Hellman (ECDH) is an implementation of the Diffie-Hellman key exchange protocol using elliptic curve cryptography. It allows two parties, each having their own elliptic curve public/private key pair, to generate symmetric keys over an unsecure channel simultaneously .
Extensible Authentication Protocol (EAP)	The *Extensible Authentication Protocol* (EAP) is a framework that provides a standardized method to negotiate wireless authentications between wireless devices. It's not a specific protocol, but rather, a framework in which other protocols work. EAP allows the client and server to negotiate the characteristics of authentication. When a connection is established, the client and server negotiate the authentication type that will be used based on the allowed or required authentication types configured on each device. EAP allows authentication using a variety of methods, including passwords, certificates, and smart cards. It is also used as an alternative to CHAP and PAP authentication protocols because it is more secure and supports different authentication mechanisms.
Message Digest Function (MD5)	*Message Digest Function* (MD5) is an algorithm that produces a value of 128 bits with 32 hexadecimal characters. MD5 is the latest edition, and is more secure than earlier versions. The MD5 algorithm takes a message of arbitrary length as the input and then outputs a 128-bit fingerprint or message digest of the input. MD5 is not collision-resistant, so you should use it with the latest algorithms like SHA-2 and SHA-3. It is still deployed for digital signature applications, file integrity checking, and storing passwords.

Rivest, Shamir, Adleman (RSA)	*Rivest, Shamir, Adleman* (RSA) is a public-key cryptosystem used for secure data transmission. It is based on factoring large numbers into their prime values. It was developed by Rivest, Shamir, and Adleman in 1977. RSA: ▪ Is widely used and is one of the de-facto encryption standard asymmetric cryptosystems. ▪ Is based on the difficulty of factoring N, a product of two large prime numbers (201 digits). ▪ Has a key length ranges from about 512 bits to 8,000 bits (2401 digits). ▪ Uses modular arithmetic and elementary number theories to perform computations using two large prime numbers.
Secure Hashing Algorithm (SHA)	*Secure Hash Algorithm* (SHA) is a cryptographic hash function. It generates a cryptographically secure one-way hash and is frequently used on the internet. SHA was published by the NIST as a US Federal Information Processing Standard. It is available in the following versions: ▪ SHA-1 produces a 160-bit digest from a message with a maximum length of $(2^{64} - 1)$ bits, and it resembles the MD5 algorithm. ▪ SHA-2 is a family of two similar hash functions with different block sizes (namely, SHA-256, which uses 32-bit words, and SHA-512, which uses 64-bit words). SHA-2 is used for security applications and protocols, including TSL and SSL. ▪ SHA-3 uses the sponge construction in which message blocks are XORed into the initial bits of the state.

Cryptography Tools

The following table identifies hashing tools you can use to verify that two files are identical. For example, you can compare the hash value of a file on a website with the hash value of that file after you download it. If the hash values match, you downloaded the unmodified version of the website file.

Tool	Description
MD5 Calculator	The *MD5 Calculator* is a tool used to create the MD5 hash value of the selected file. To use the MD5 calculator, right-click the file and select MD5 Calculator. The program will calculate the MD5 hash.
HashMyFiles	*HashMyFiles* is a free utility that calculates the MD5 and SHA1 hashes of files.