CHEAT SHEET

This cheat sheet is designed as a way for you to quickly study the key points of this chapter.

Choosing an encryption algorithm

- If you need to encrypt data that is used locally, or you have a secure way to distribute the encryption key, use the symmetric encryption.
- If you don't have a secure way to send the encryption key data between parties, then asymmetric encryption is recommended.
- If you need only to ensure integrity of the data, use a hashing algorithm.
- If you need to ensure both integrity and authenticity, choose a MAC algorithm.

Symmetric encryption

- Based on a common key called shared secret.
- It needs an initialization vector (IV) that doesn't need to be secret but is used to encrypt the first block of data.
- You use it by instantiating a symmetric algorithm object and then calling `CreateEncryptor` or `CreateDecryptor`.
- The encryptor/decryptor is then used with either by calling directly the `TransformFinalBlock` method or by sending it to a `CryptoStream`.

Asymmetric encryption

- It is based on a pair of complementary keys. Encrypted data with one key can be decrypted only with the other key.
- One key is kept secret and is called a *private key*; the other one is made available to anyone that wants to encrypt data, or verify encrypted data, and it is called a *public key*.

Hashing

- Mapping binary data of a variable length to a fixed size binary data, called *hash*.
- When you need to make sure that data is not modified while transferred, you can calculate the cryptographic hash and send it together with the data to be verified by the receiving party.
- The two commonly used algorithms are SHA256 and SHA512 with resulting hashes of 256 and 512 bits, respectively (32 and 64 bytes).

Key management

- Symmetric keys can be exchanged using asymmetric algorithms.
- Asymmetric private keys can be secured either by using certificates or by using Crypto Service Providers containers.

Assembly version

- An assembly version is specified by four parts: Major, Minor, Build, and Revision.

Strong name

- An assembly that is digitally signed is called a *strong named assembly*.
- A strong name has five parts: Friendly Name, Version, Culture, Public Key Token, and Processor Architecture.

GAC

- Stands for *Global Assembly Cache*.
- A repository to share .NET assemblies.
- Only strong named assemblies can be deployed on the GAC.
- Several versions of the same assembly can be deployed on the GAC at the same time.

REVIEW OF KEY TERMS

assembly An assembly is the unit of reuse, deployment, versioning, and security.

asymmetric encryption (public key) A cryptographic algorithm that uses two complementary keys, one for encryption and one for decryption. Data encrypted with the public key can only be decrypted using the private key.

Certificate Authority (CA) An entity that issues digital certificates.

Certificate Revocation List (CRL) A list of digital certificates that has been revoked for various reasons. You shouldn't use a certificate if it is revoked.

certificate stores A special storage location on your computer, used to store encryption certificates.

Common Language Runtime (CLR) CLR is the component of .NET Framework responsible for running .NET applications and managing their running environment.

cryptography The practice and study of techniques for secure communication.

decryption The process of decoding previously encrypted data so that it can be used by your application.

encryption The process of encoding data so that it cannot be read by an unauthorized person.

Global Assembly Cache (GAC) GAC is a machine-wide code cache.

hash bucket A data structure that holds items that share the same hash value.

hashing Used to map data structures of variable length, to fixed size data structures. Hashing the same data using the same algorithm will always yield the same hash value.

initialization vector (IV) A data array used by the encryption algorithms to encrypt the first data block. The IV doesn't need to be kept secret.

Intermediate Language (IL) The result of compiling a .NET application from source code.

Just In Time compiler (JIT) A component of the .NET that transforms the IL into binary code that can be run on the target platform.

Message Authentication Code (MAC) A family of cryptographic algorithms used to provide data integrity and authenticity.

private key The public and private keys are a pair of complementary keys used together in the asymmetric encryption. Data encrypted with the private key can only be decrypted using the public key, and data encrypted with the public key can only be decrypted using the private key.

public key See *private key*.

Public Key Infrastructure (PKI) The infrastructure needed to handle digital certificates.

Secured Hash Algorithm (SHA) A family of cryptographic algorithms used to calculate hashes published by NIST.

Secure Socket Layer (SSL) A cryptographic protocol used for secure communication over the Internet.

symmetric encryption (shared secret) A cryptographic algorithm that uses the same key for both encryption and decryption of data.

Transport Layer Security (TLS) A cryptographic protocol used for secure communication over the Internet, the successor of SSL.

EXAM TIPS AND TRICKS

The Review of Key Terms and the Cheat Sheet for this chapter can be printed to help you study. You can find these files in the ZIP file for this chapter at `www.wrox.com/remtitle.cgi?isbn=1118612094` on the Download Code tab.