Data Confidentiality

Context

Data passes between a client and a Web service, sometimes through one or more intermediaries. Messages may also be kept in repositories, such as message queues or databases. Some of the data within the messages is considered to be sensitive in nature. There is a risk that an attacker can gain access to sensitive data, either by eavesdropping on the network or accessing a repository.

Problem

How do you protect data within a message from being disclosed to unintended parties?

Forces

Any of the following conditions justifies using the solution described in this pattern:

- **Disclosure of sensitive data can result in loss or damage, such as identity theft, lawsuits, loss of business, or regulatory fines**. Any data that contains sensitive information must be protected from unauthorized users.
- **Sensitive data may pass across the network**. Sensitive data must be protected from disclosure in transit. An eavesdropper can gain access to sensitive data whenever it leaves a secure area (such as a protected memory space) or crosses a non-secure communication line (such as a public network).
- **Sensitive data may be persisted for short periods of time, such as in a message queue, or over longer periods of time in a database or a file**. Sensitive data must be protected from disclosure in locations where it is persisted.

Solution

Use encryption to protect sensitive data that is contained in a message. Unencrypted data, which is known as plaintext, is converted to encrypted data, which is known as ciphertext. Data is encrypted with an algorithm and a cryptographic key. Ciphertext is then converted back to plaintext at its destination.

Participants

Data confidentiality involves the following participants:

- **Sender**. The sender is the originator of a message. A client can send a request message to a Web service, and a Web service can send a response message back to a client that has sent a request message.
- **Recipient**. The recipient is the entity that receives a message from the sender. A Web service is the recipient of a request message that is sent by a client, and a client is the recipient of a response message that it receives from a Web service.

Process

You can apply data confidentiality in two steps:

1. **Encrypting the data**. In this step, the sender converts plaintext to ciphertext, rendering it unintelligible to parties other than the intended recipient.
2. **Decrypting the data**. In this step, ciphertext is rendered intelligible to the intended recipient by converting it back to plaintext.

You can use two types of cryptography to provide data confidentiality: symmetric and asymmetric. While both symmetric cryptography and asymmetric cryptography follow the same basic process, they each have their own unique characteristics.

Symmetric Cryptography

With symmetric cryptography, both the sender and recipient share a key that is used to perform both encryption and decryption. Symmetric cryptography is commonly used to perform encryption. It also provides data integrity when symmetric keys are used in conjunction with other algorithms to create Message Authentication Codes (MACs). For more information about MACs, see Data Origin Authentication in Chapter 2, "Message Protection Patterns."

Figure 2.2 illustrates the process of encrypting and decrypting data with a shared secret key.

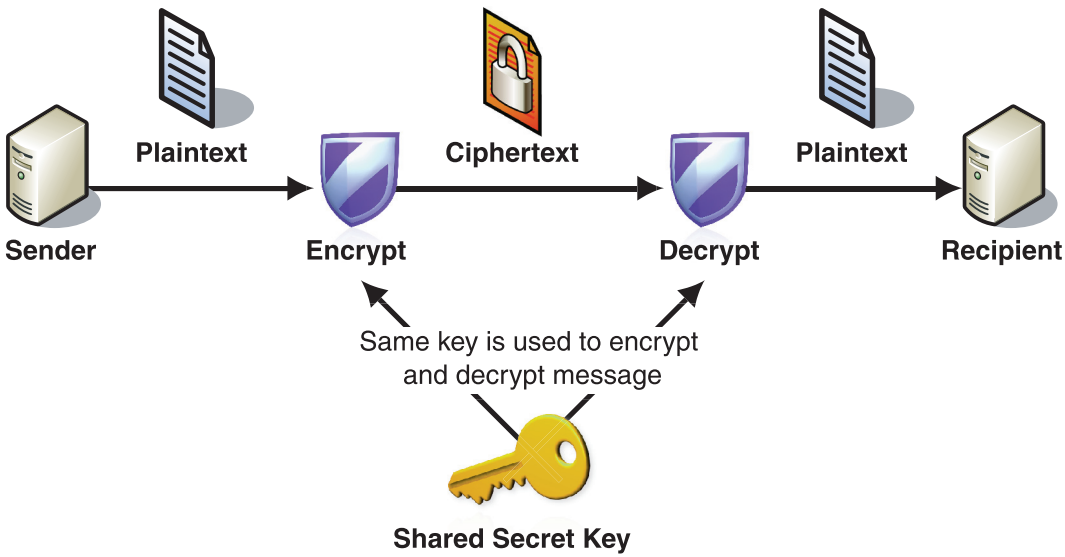

Figure 2.2
The process of symmetric encryption

As illustrated in Figure 2.2, symmetric encryption involves the following steps:

1. The sender creates a ciphertext message by encrypting the plaintext message with a symmetric encryption algorithm and a shared key.
2. The sender sends the ciphertext message to the recipient.
3. The recipient decrypts the ciphertext message back into plaintext with a shared key.

Numerous symmetric algorithms are currently in use. Some of the more common algorithms include Rijndael (AES) and Triple DES (3DES). These algorithms are designed to perform efficiently on common hardware architectures.

Symmetric cryptography is comparatively simple in nature, because the secret key that is used for both encryption and decryption is shared between the sender and the recipient. However, before communication can occur, the sender and the recipient must exchange a shared secret key. In some cases (such as SSL), asymmetric cryptography can be used to ensure that the initial key exchange occurs over a secure channel.

Asymmetric Cryptography

With asymmetric cryptography (also known as public key cryptography), the sender encrypts data with one key, and the recipient uses a different key to decrypt ciphertext. The encryption key and its matching decryption key are often referred to as a public/private key pair.

Note: In addition to providing encryption, you can use public key cryptography to provide digital signatures, facilitating nonrepudiation, and for key management purposes. For more information, see Data Origin Authentication in Chapter 2, "Message Protection Patterns."

The public key of the recipient is used to encrypt data. It can be openly distributed to those who want to encrypt a message to the recipient. The private key of the recipient is used to decrypt messages, and only the recipient must be able to access it.

Figure 2.3 illustrates the process of asymmetric encryption and asymmetric decryption.

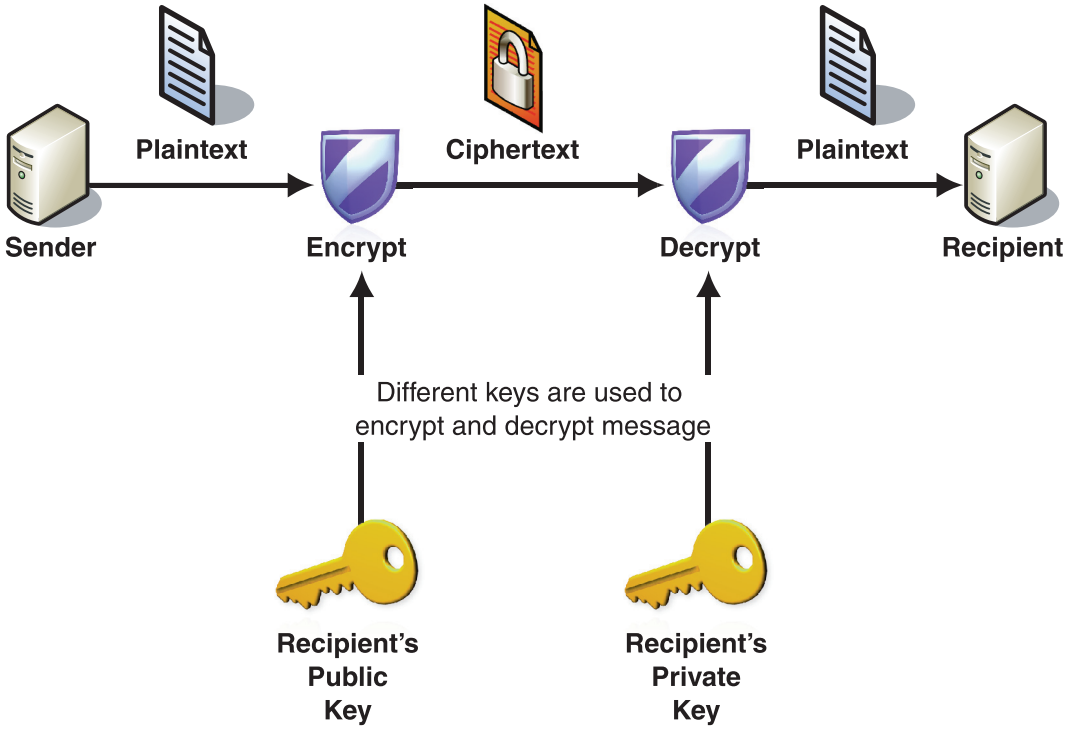

Figure 2.3
The process of asymmetric encryption

As illustrated in Figure 2.3, asymmetric encryption involves the following steps:

1. The sender creates a ciphertext message by encrypting the plaintext message with an asymmetric encryption algorithm and the recipient's public key.
2. The sender sends the ciphertext message to recipient.
3. The recipient decrypts the ciphertext message back to plaintext using the private key that corresponds to the public key that was used to encrypt the message.

Few asymmetric algorithms are currently in use. The most commonly used asymmetric algorithm is the RSA algorithm.

Asymmetric encryption requires more processing resources than symmetric encryption. For this reason, asymmetric encryption is usually optimized by adding a one time high-entropy symmetric key to encrypt a message and then asymmetrically encrypting the shared key. This reduces the size of the data that is asymmetrically encrypted and also improves performance.

In cases where more than one message exchange occurs between two parties, a high-entropy shared secret can be negotiated between a sender and a receiver. In this case, the first exchange includes a shared secret that is encrypted asymmetrically and based on the shared secret, additional message exchanges are performed symmetrically. Key derivation techniques are often used to add variability to shared secrets that are used over multiple message exchanges. For more information, see "Extension 1 — Establishing a Secure Conversation" in Brokered Authentication: Security Token Service (STS) in Chapter 1, "Authentication Patterns."

Example

Global Bank publishes a Web service to provide business customers with the ability to upload payroll account transfers. Direct deposit account information is considered very sensitive for both the business and the customer. Compromising this information can result in unauthorized account activity or disclosure of employee salary information. For this reason, Global Bank requires that any messages containing account data are encrypted as they pass between clients and the Web service to provide data confidentiality.

Resulting Context

This section describes some of the more significant benefits, liabilities, and security considerations of using this pattern.

Note: The information in this section is not intended to be comprehensive. However, it does discuss many of the issues that are most commonly encountered for this pattern.

Benefits

By blocking unauthorized parties from viewing messages, you can prevent financial loss and legal liability because of the disclosure of sensitive information.

Liabilities

The liabilities associated with the Data Confidentiality pattern include the following:

- Cryptography operations are computationally intensive and impact system resource usage. This affects the scalability and performance of the application.
- Key management, which safeguards encryption keys from being compromised, can have significant administrative overhead. Factors that affect the administrative complexity of key management include:
  - The number and type of keys used.
  - The type of encryption used (symmetric or asymmetric).
  - The key management infrastructure in use.

Security Considerations

Security considerations associated with the Data Confidentiality pattern include the following:

- Encryption does not prevent data tampering. For example, a man-in-the-middle attack can replace the bits in transit, which can cause the receiver to decrypt the data to something other than the original plaintext. Without data origin authentication protection, the receiver has no way to verify that decrypted ciphertext is the same as the original plaintext. For this reason, the implementation patterns that implement data confidentiality also implement data origin authentication.
- If too much data is encrypted with the same symmetric key, an attacker can intercept several messages and attempt to cryptographically attack the encrypted messages, with the goal of obtaining the symmetric key. To minimize the risk of this type of attack, you should consider generating session-based encryption keys with a relatively short life span. Typically, these session keys are derived from a master symmetric key, such as a shared identity secret. Usually, the session key is exchanged by using asymmetric encryption during the initial interaction of a sender and recipient. Session keys should be discarded and replaced at regular intervals, based on the amount of data or number of messages that they are used to encrypt.
- Much of the strength of symmetric encryption algorithms comes from the randomness of their encryption keys. If keys originate from a source that is not sufficiently random, attackers may narrow down the number of possible values for the encryption key. This can make it possible for a brute force attack to discover the key value from encrypted messages that the attacker has intercepted. For example, a user password that is used as an encryption key can be very easy to attack because user passwords are typically a non-random value of relatively small size that a user can remember without writing it somewhere.
- You should use published, well-known encryption algorithms that have withstood years of rigorous attacks and scrutiny. Use of encryption algorithms that have not been subjected to rigorous review by trained cryptologists may contain undiscovered flaws that are easily exploited by an attacker.
- Each country may recognize different standards for data privacy/protection. For example, in the U.S., regulations such as Sarbanes-Oxley, HIPAA, and the Privacy Act of 1974 require that measures are taken to prevent disclosure of sensitive personal information or that there is accountability for the management of sensitive data. In the European Union (EU), regulations such as the Data Protection Directive enforce stringent standards for data privacy.

Related Patterns

The following child patterns are related to the Data Confidentiality pattern:

- **Implementing Direct Authentication with UsernameToken in WSE 3.0**. This pattern focuses on using direct authentication at the message layer in WSE 3.0.
- **Implementing Message Layer Security with X.509 Certificates in WSE 3.0**. This pattern provides guidelines for implementing brokered authentication, authorization, data integrity, and data origin authentication with X.509 certificates in WSE 3.0.
- **Implementing Message Layer Security with Kerberos in WSE 3.0**. This pattern provides guidelines for implementing brokered authentication, authorization, data integrity, and data origin authentication with the Kerberos version 5 protocol in WSE 3.0.
- **Implementing Transport Layer Data Confidentiality Using HTTPS**. This reference provides concise information about using data confidentiality and integrity with HTTPS.
- **Implementing Transport Layer Security Using X.509 Certificates and HTTPS**. This reference provides concise information about how to use SSL for data confidentiality and data integrity. It includes information about how to use SSL client certificates for brokered authentication and data origin authentication.