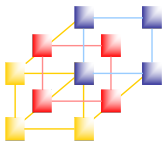
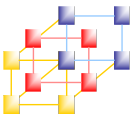
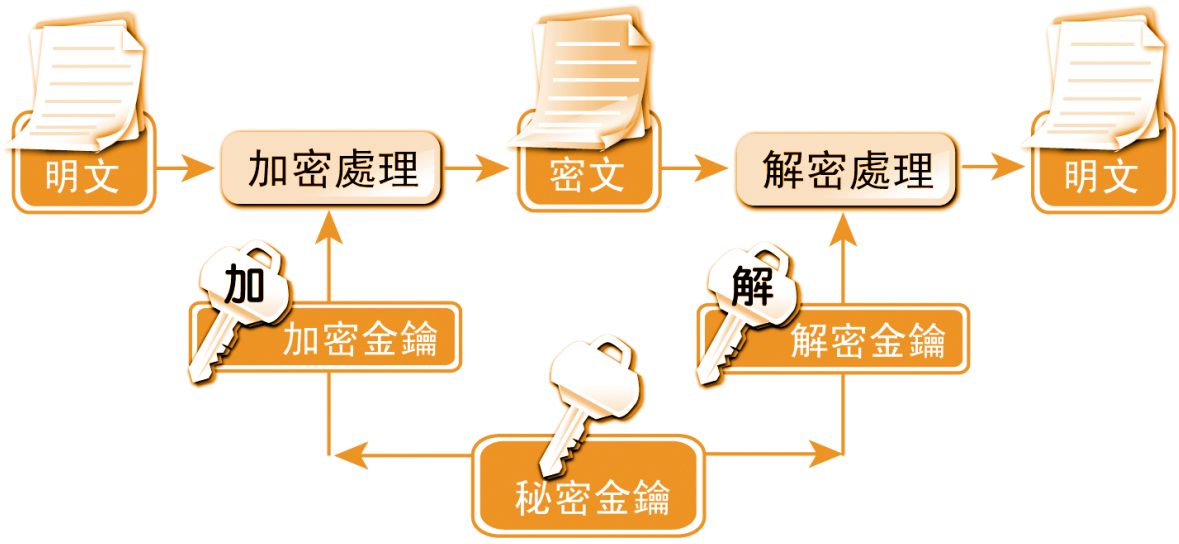


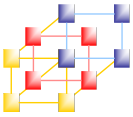
Unit 7
Secure Socket



密碼學基本概念



基本的加解密系統



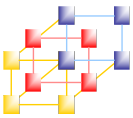
密碼系統的分類

- 對稱性密碼系統(Symmetric Cryptosystems)或秘密金鑰密碼系統(Secret-Key Cryptosystems) 或單金鑰密碼系統(One-Key Cryptosystems)

  加密金鑰及解密金鑰為同一把

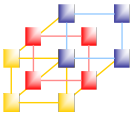
- 非對稱性密碼系統(Asymmetric Cryptosystems)或公開金鑰密碼系統(Public-Key Cryptosystems) 或雙金鑰密碼系統(Two-Key Cryptosystems)

  加密與解密金鑰為不相同的二把金鑰



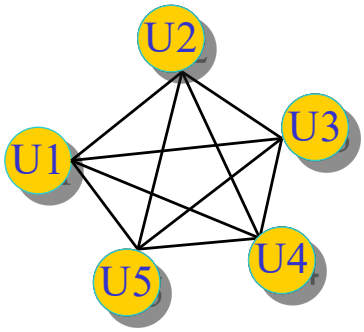
對稱式加解密法

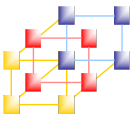
- DES (Data Encryption Standard)
- Triple DES
- AES (Advanced Encryption Standard)
- …



公開金鑰基本概念

- 對稱式密碼系統有金鑰的管理問題
  - 例如要與N個人做秘密通訊，那麼就必須握有N把秘密金鑰
- 為了改善對稱式密碼系統問題，於是便有公開金鑰密碼系統(Public-Key Cryptosystems)的產生

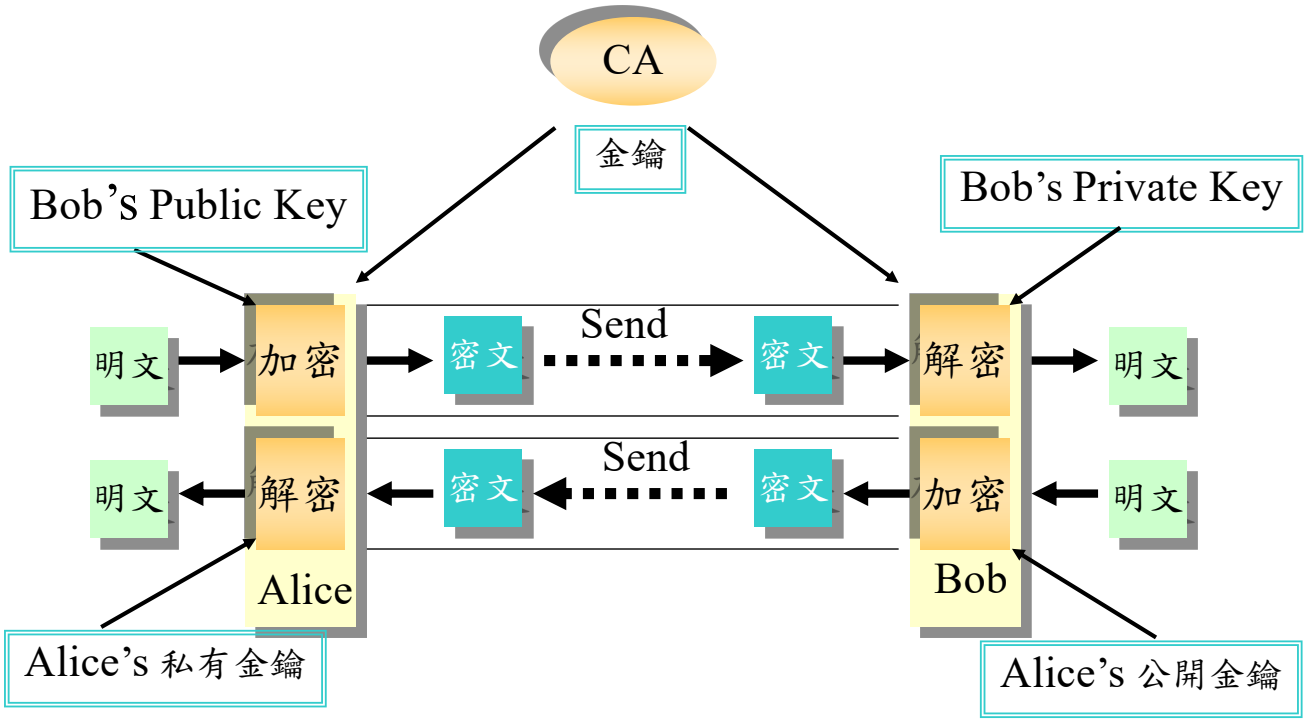




公開金鑰密碼系統

- 著名之公開密碼系統
  - RSA密碼系統
  - ElGamal密碼系統
  - Elliptic Curve Cryptosystem, ECC橢圓曲線的密碼系統
- 公開密碼系統優點
  - 沒有金鑰管理的問題
  - 高安全性
  - 有數位簽章功能
- 公開密碼系統缺點
  - 加解密速度慢

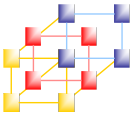
公開金鑰加密系統



RSA 加密法

- 非對稱式密碼系統的一種。
  - 1978年美國麻省理工學院三位教授Rivest、Shamir、Adleman (RSA) 所發展出來的。
- 利用公開金鑰密碼系統作為資料加密的方式，可達到資料加密及數位簽署的功能。

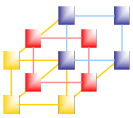
- Encryption
  - RSA 加密演算法，明文加密使用區塊為每次加密的範圍，使用對方公開金鑰 (Public Key) 將明文加密。
- Decryption
  - RSA 解密演算法，必須使用自己的私有金鑰 (Private Key) 才能將密文解出。



RSA 演算法

- 張三選 2 個大質數 p 和 q (至少100位數)，
  令 N = p ・ q
- 再計算Ø(N)=(p-1)(q-1)，並選一個與Ø(N)互質數 e
  - Ø(N)為Euler's Totient函數，其意為與N互質之個數
- (e, N) 即為張三的公開金鑰
- 加密法為 $C = M^e \bmod N$
- 張三選 1 個數 d，滿足 e ・ d mod Ø(N) = 1
- d 即為張三的解密金鑰(亦稱私有金鑰或祕密金鑰)
- 解密法為 $M = C^d \bmod N$

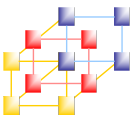
• RSA之安全性取決於質因數分解之困難度
• 要將很大的N因數分解成P跟Q之相乘，是很困難的



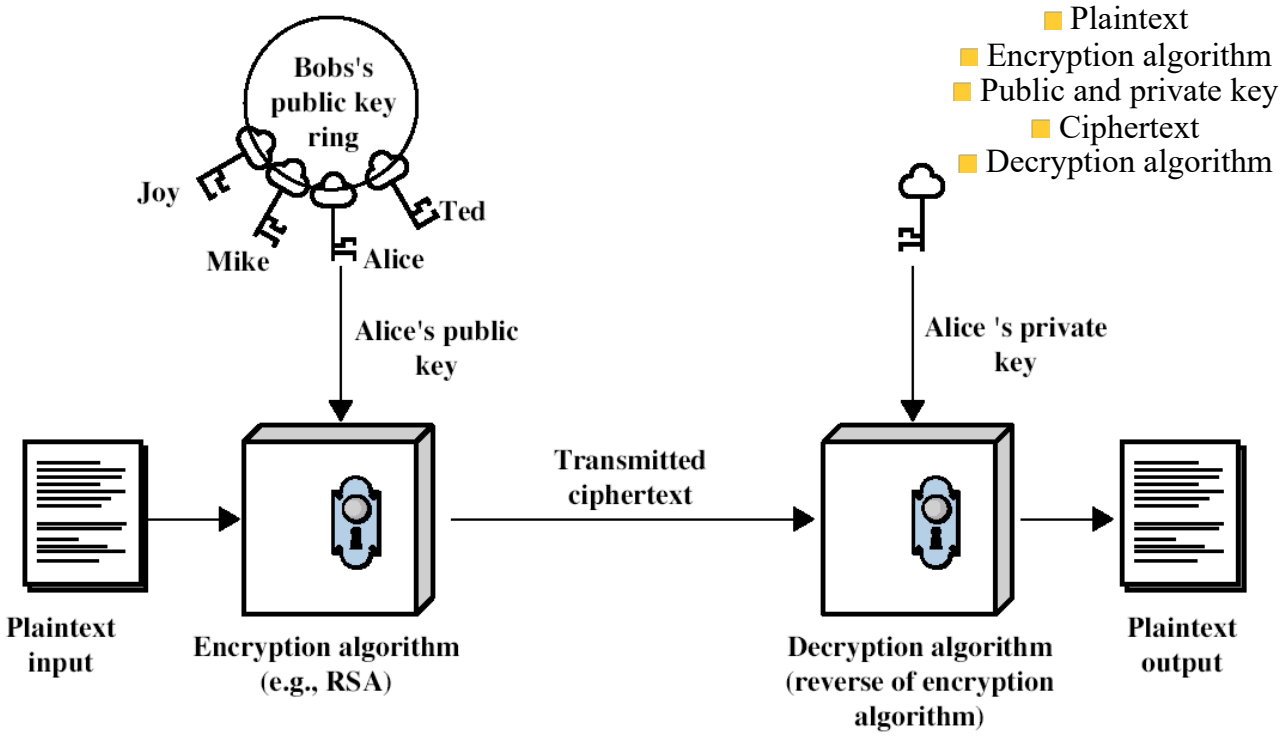
RSA 演算法- 例子

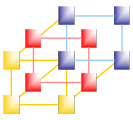
- 張三選 p = 3 ， q = 11
  此時 N = p ・ q = 3 x 11 = 33
- 張三選出一個與 (p-1) x (q-1) = (3-1)(11-1) = 20互質數 e = 3
- (e, N) = (3,33) 即為張三的公開金鑰
- 張三選一個數 d = 7 當作解密金鑰，
  滿足 e ・ d ≡ 1 mod 20 (7 x 3 ≡ 1 mod 20)

- 令明文 M = 19
  - 加密 : $C = M^e \bmod N = 19^3 \bmod 33 = 28$
  - 解密 : $M = C^d \bmod N = 28^7 \bmod 33 = 19$

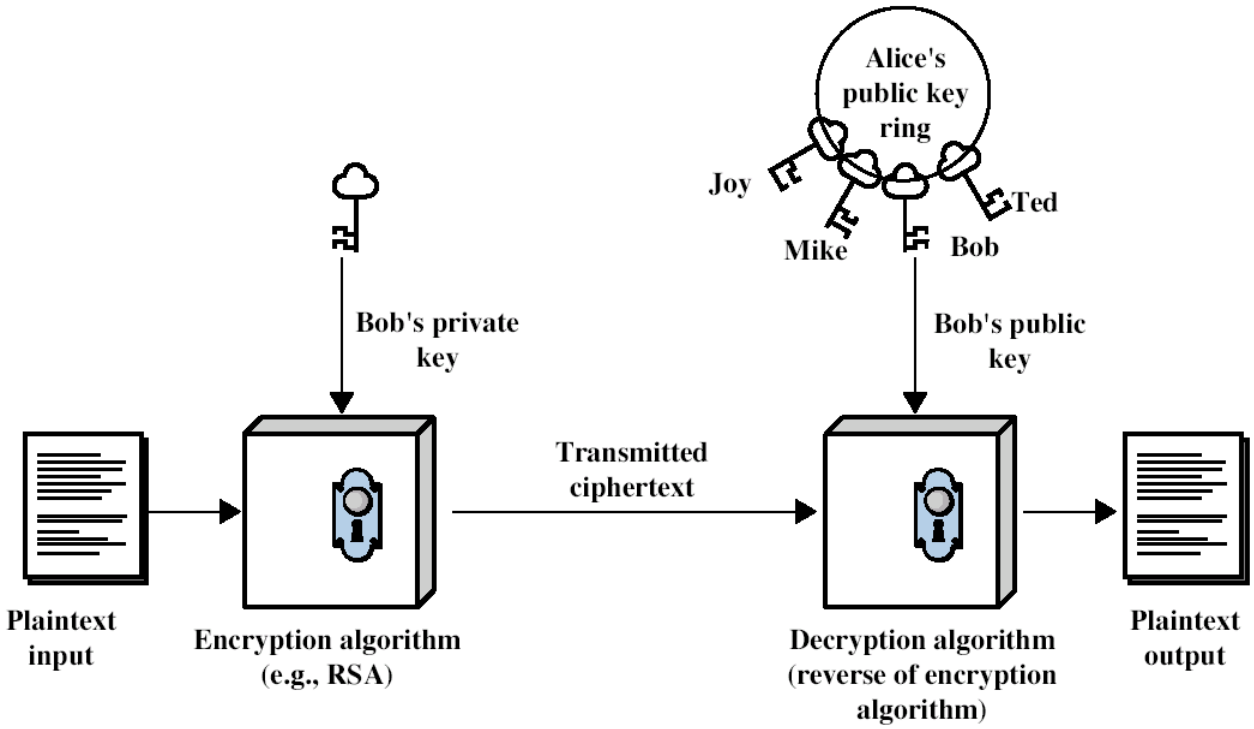


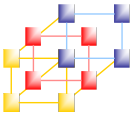
Public-Key Cryptography -- Encryption



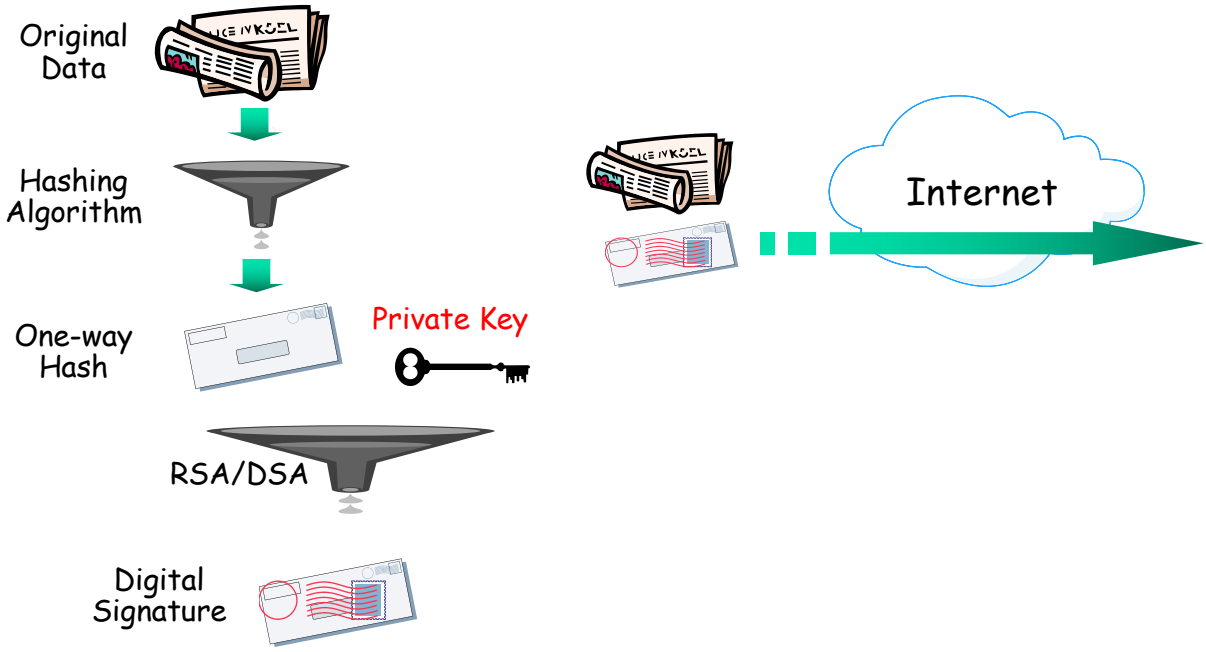


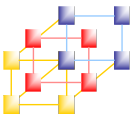
Public-Key Cryptography -- Authentication



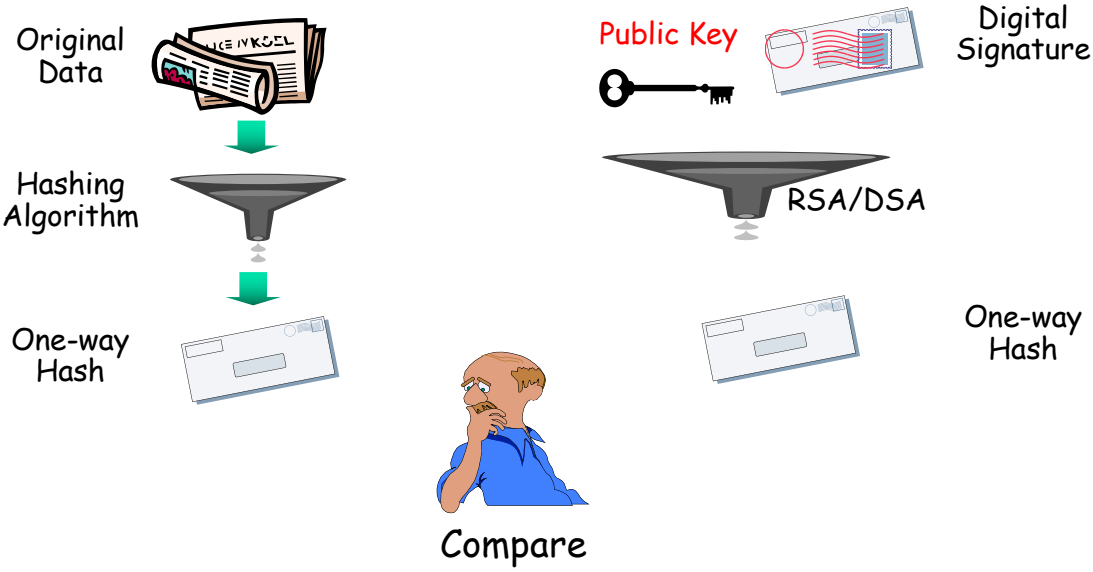


Digital Signature -- Sender

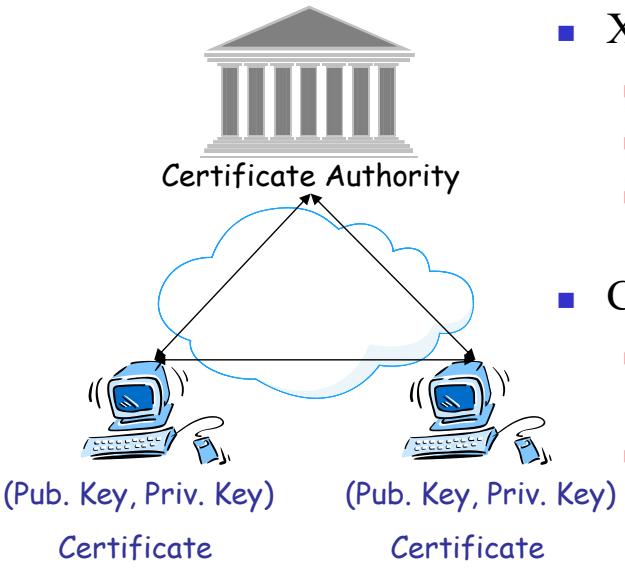




Digital Signature -- Receiver

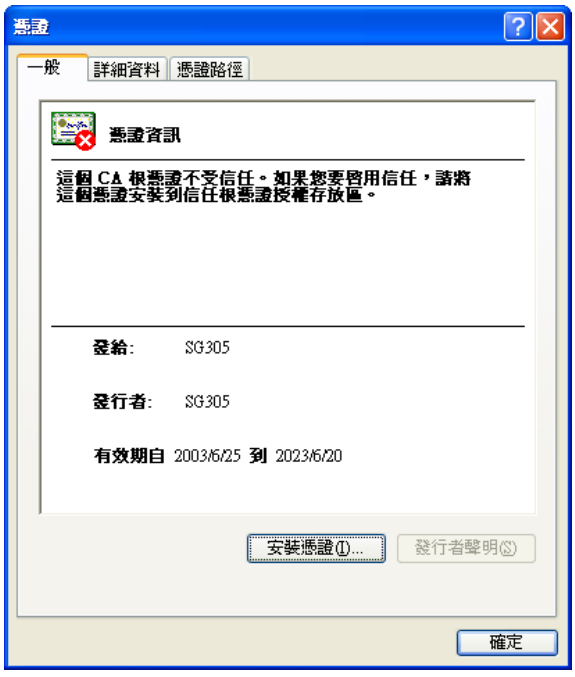


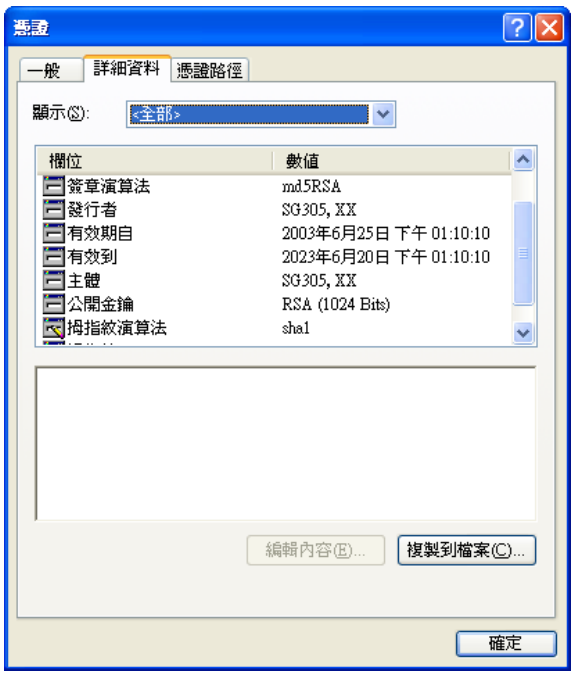
Public-Key Infrastructure

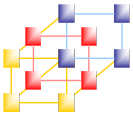


- X.509 (ITU-T)
  - Directory service
  - Authentication Framework
  - Lightweight Directory Access Protocol (LDAP; RFC1777)
- Certification Revocation List (CRL)
  - Lightweight Directory Access Protocol (LDAP; RFC1777)
  - Online Certificate Status Protocol (OCSP;RFC2560)

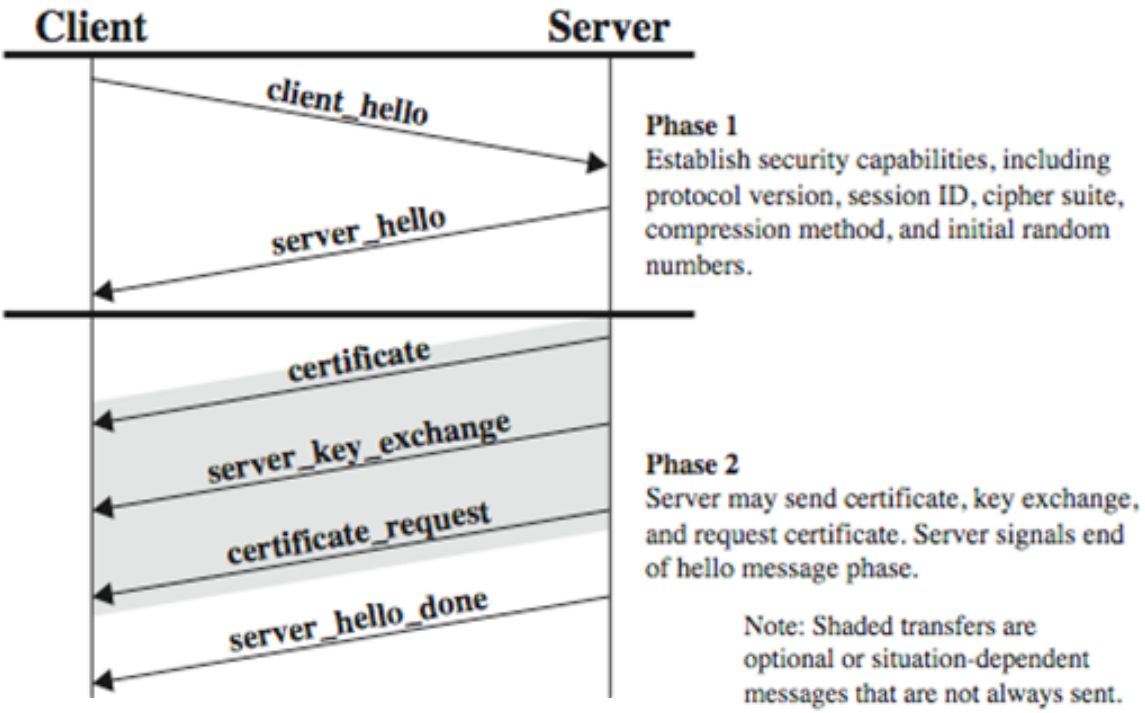
Certificate

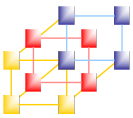




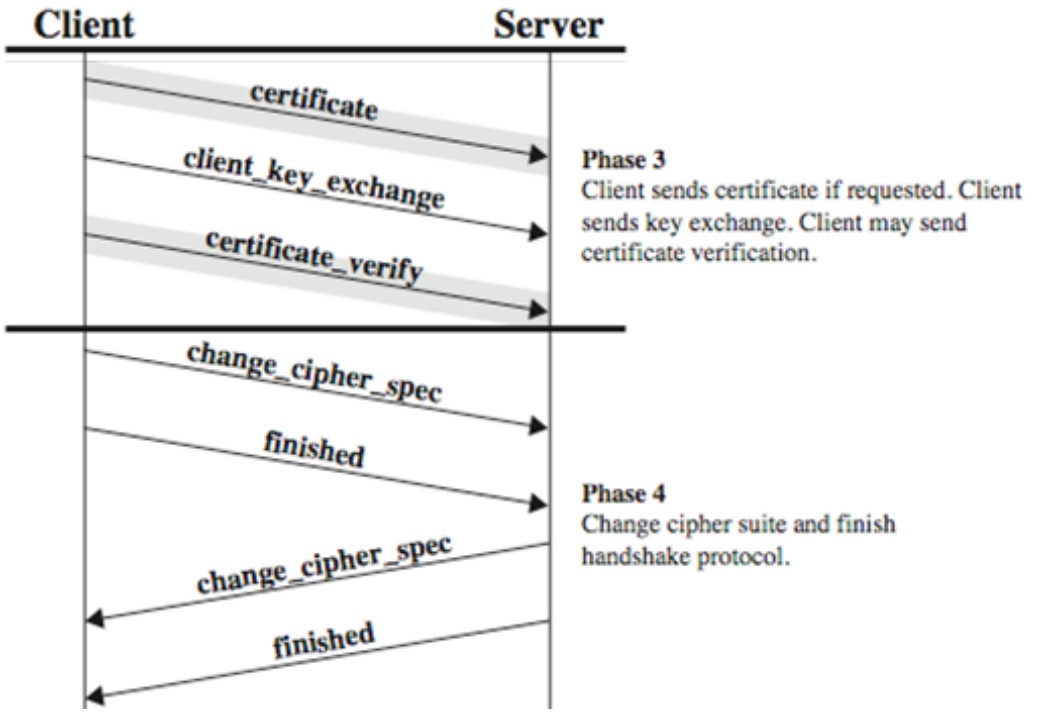


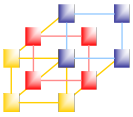
SSL Handshake Protocol (1/2)





SSL Handshake Protocol (2/2)

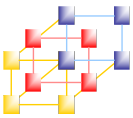




SSL Sockets

- 使用 javax.net.ssl.SSLSocketFactory 來建立 SSL Sockets
  - SSLSocketFactory 為一個抽象類別，透過 getDefault() method 建立此抽象類別的實例
  - 利用createSocket() 來建立 Secure Sockets
  - 利用 setEnabledCipherSuites() 來設定加密套件
- Example

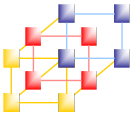
```
SSLSocketFactory factory =
   (SSLSocketFactory)SSLSocketFactory.getDefault();
SSLSocket client =
   (SSLSocket)factory.createSocket(host, port);
client.setEnabledCipherSuites(
 client.getSupportedCipherSuites());
```

Cipher Suites

- Different implementations of the JSSE support different combinations of authentication and encryption algorithms
  - Oracle bundles with Java 7 only supports 128-bit AES encryption
- getSupportedCipherSuites()
  - public abstract String[] getSupportedCipherSuites()
  - tells which combination of algorithms is available on a given socket
- getEnabledCipherSuites()
  - public abstract Strint[] getEnabledCipherSuites()
  - tells which suites this socket is willing to use
- setEnabledCipherSuites()
  - change the suites the client attempts to use
  - Four parts
    - protocol, key exchange algorithm, encryption algorithm, and checksum

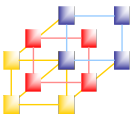
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
- Secure Sockets Layer Version 3;
- Diffie-Hellman method for key agreement;
- no authentication;
- DES encryption with 40-bit keys;
- Cipher Block Chaining, and
- the Secure Hash Algorithm checksum



Session Management

- SSL allows sessions to extend over multiple sockets
  - Multiple sockets within the same session use the same set of public/private keys
- JSSE represents by instances of the SSLSession interface
  - Reuses the session's keys automatically if
    - Multiple secure sockets to one host on one port are opened
    - Within a reasonably short period of time
  - In high security applications, you may want to disallow session-sharing between sockets or force reauthentication of a session
- getSession() method of SSLSocket returns the Session
  - Get various information about the session
- setEnableSessionCreation() method
  - To allow/disallow session
- startHandshake() method
  - To reauthenticate a connection

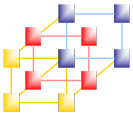
SimpleSSLServer.java
SimpleSSLClient.java



SSL Server Socket

- 使用 javax.net.ssl. SSLServerSocketFactory 來建立 SSL Server Sockets
  - SSLServerSocketFactory 為一個抽象類別，透過 getDefault() method 建立此抽象類別的實例
  - 利用createServerSocket() 來建立 Secure Server Sockets
- Example

```
SSLServerSocketFactory sslserversocketfactory =
       (SSLServerSocketFactory)
       SSLServerSocketFactory.getDefault();
SSLServerSocket sslserversocket = (SSLServerSocket)
       sslserversocketfactory.createServerSocket(port);
SSLSocket sc = (SSLSocket) sslserversocket.accept();
```



Command

- keytool
  - -genkeypair　　產生金鑰組
    - -keystore <keystore>　　金鑰儲存庫名稱
    - -keyalg <keyalg>　　金鑰演算法名稱
  - keytool –genkeypair –keystore ServerKeyStore –keyalg RSA
- java -Djavax.net.ssl.keyStore=ServerKeyStore -Djavax.net.ssl.keyStorePassword=123456 SimpleSSLServer
- java -Djavax.net.ssl.trustStore=ServerKeyStore -Djavax.net.ssl.trustStorePassword=123456 SimpleSSLClient localhost