Lecture 10
Asymmetric-Key Encryption

Jason Lin

學習目標

- 區別對稱式金鑰密碼系統與非對稱式金鑰密碼系統
- 介紹單向暗門函數以及其於非對稱式金鑰密碼系統的使用情形
- 討論 RSA 密碼系統
- 討論 Rabin 密碼系統
- 討論 ElGamal 密碼系統

10.1 簡介

- 對稱式金鑰密碼學與非對稱式金鑰密碼學可以同時並存，並且提供服務。我們著實認為它們之間是互補的，其中一個的優點可以補償另外一個的缺點
- 本節討論主題
  - 金鑰
  - 一般概念
  - 兩者皆需要
  - 單向暗門函數

10.1 簡介 (續)

注意

對稱式金鑰密碼技術是基於分享的祕密，非對稱式金鑰密碼技術是基於個人的祕密。

在對稱式金鑰密碼學中，符號被重新排列與取代；在非對稱式金鑰密碼學中，數字被操作處理。

10.1.1 金鑰

- 非對稱式金鑰密碼學使用兩把不同的金鑰：一把為私密金鑰（Private Key），一把為公開金鑰（Public Key）

圖 10.1 非對稱式金鑰密碼系統的上鎖與開鎖

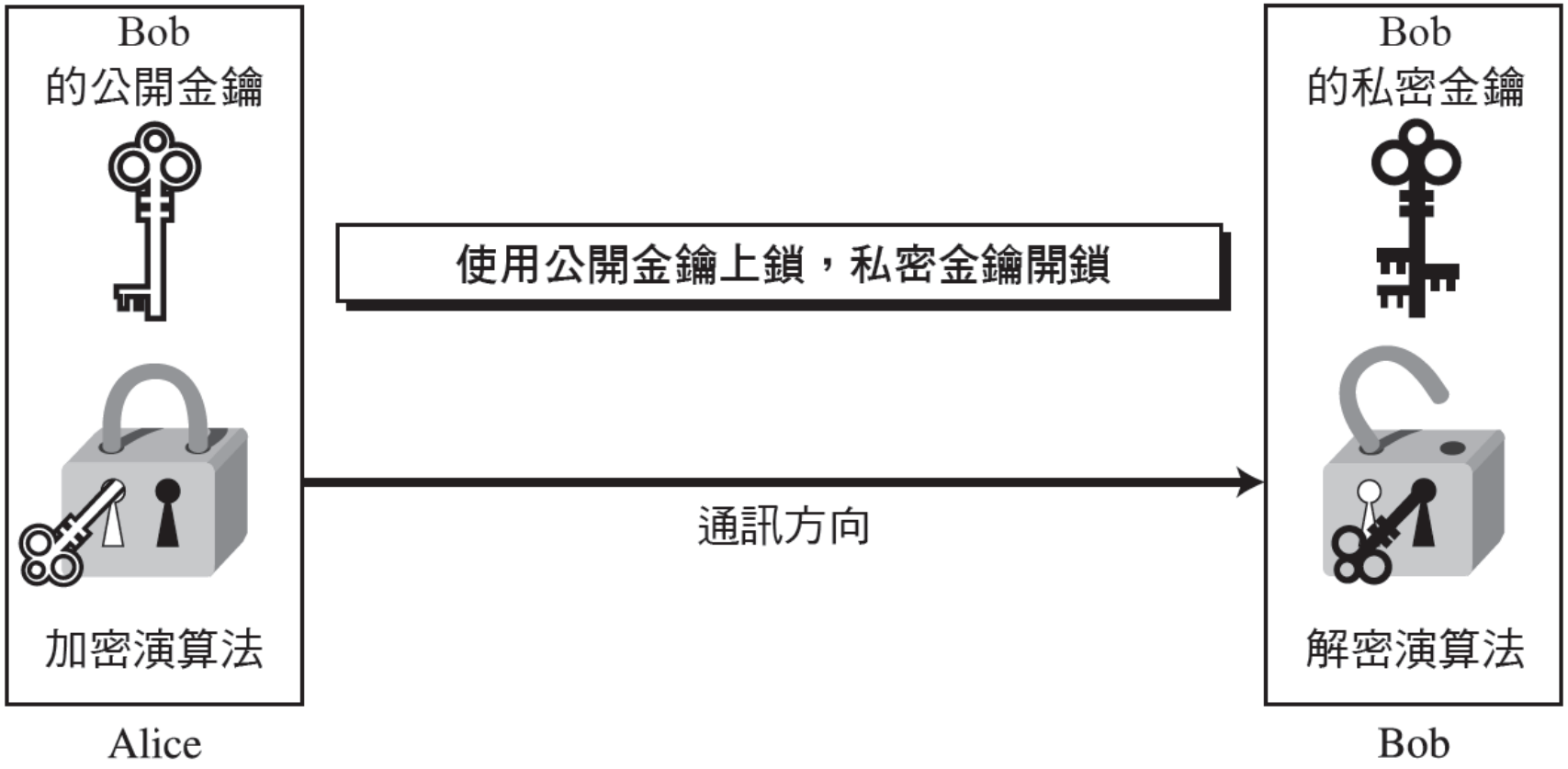
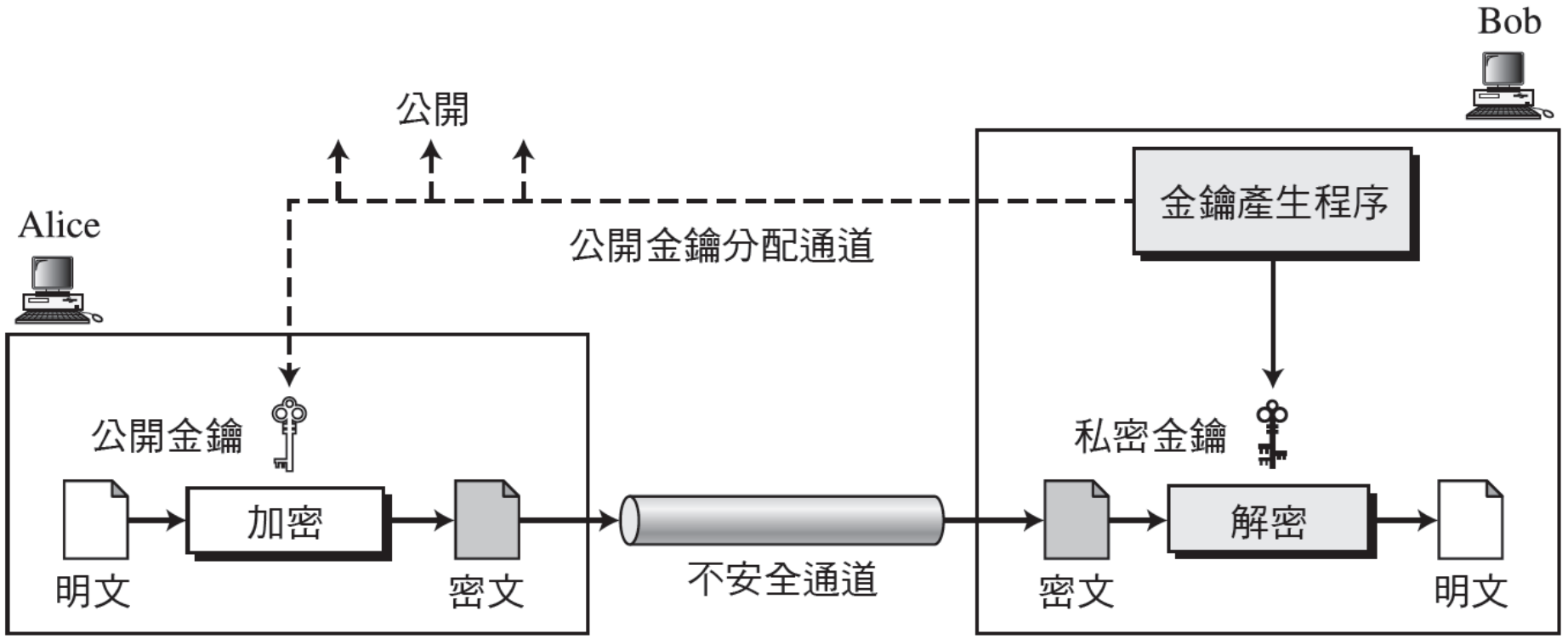


圖 10.2 非對稱式金鑰密碼系統的一般概念



10.1.2 一般概念

- 明文／密文
  - 與對稱式金鑰密碼學不同的是，非對稱式金鑰密碼學將明文（Plaintext）與密文（Ciphertext）當成整數

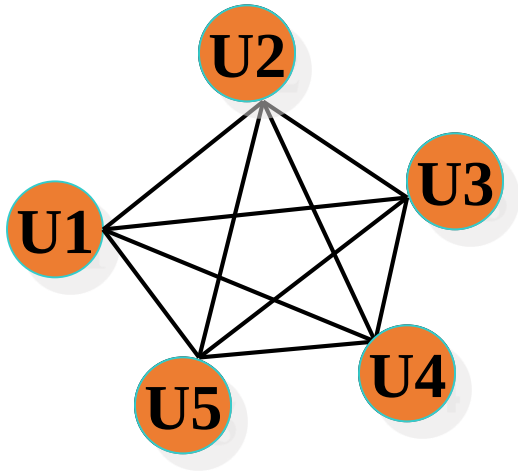
- 加密／解密
  - 密文表示為 $C = \boldsymbol{f}(K_{公開}, P)$
  - 明文表示為 $P = \boldsymbol{f^{-1}}(K_{私密}, C)$

10.1.3 兩者皆需要

- 一件重要但有時卻難以理解的事實是，非對稱式金鑰（公開金鑰）密碼學的來臨並不會抹煞我們對於對稱式金鑰（祕密金鑰）密碼學的需求

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
  - 只需要 1 把私密金鑰跟 1 把公開金鑰

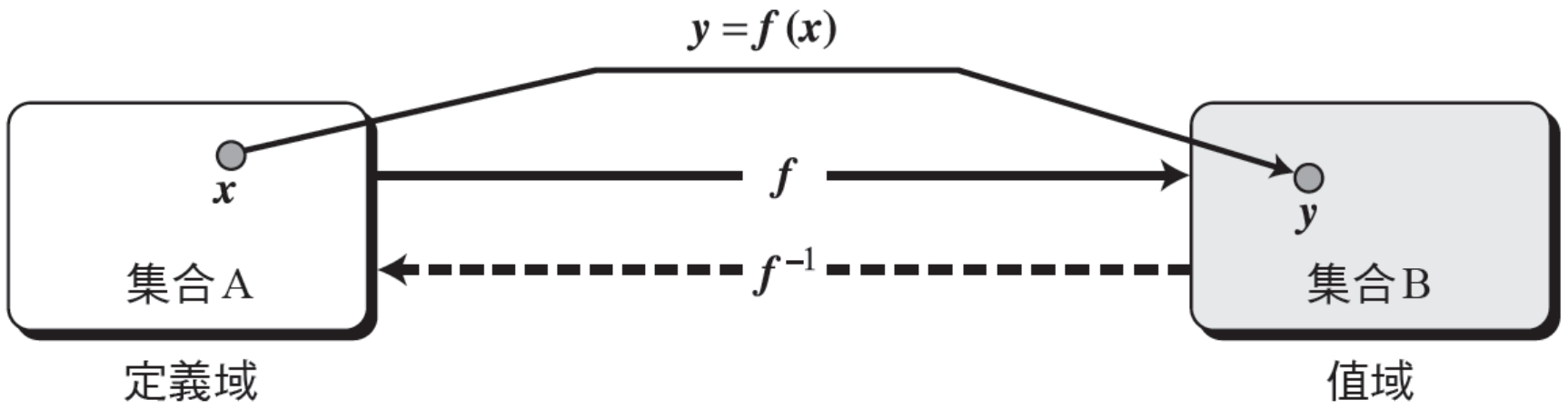


公開金鑰密碼系統

- 著名之公開金鑰密碼系統
  - Diffie-Hellman 概念 1976
  - RSA 密碼系統 1977
  - Rabin 密碼系統 1979
  - ElGamal 密碼系統 1985
- 公開金鑰密碼系統的優點
  - 沒有金鑰管理的問題
  - 高安全性
  - 有數位簽章功能
- 公開金鑰密碼系統的缺點
  - 加解密速度慢

10.1.4 單向暗門函數

- 非對稱式金鑰密碼學的主要概念是單向暗門函數



10.1.4 單向暗門函數 (續)

- 單向函數
  - f 容易計算
  - f^{-1} 很難計算
- 單向暗門函數
  - 給一個 y 以及一個暗門（Trapdoor，祕密），則 x 可以很容易計算出來

範例10.1

- 當 n 值很大，$n = p \times q$ 是一個單向函數。注意，在這個函數內，x 是指由兩個質數所構成的序列 (p, q)，而且 y 值為 n。給定 p 及 q，計算 n 值十分容易；但給 n 值，要計算 p 及 q 則非常困難。這是一個分解因數的問題，我們已於第九章中討論過。在此情形下，計算 $\boldsymbol{f^{-1}}$ 並沒有多項式時間的解法

範例10.2

- 當 n 值很大，函數 $y = x^k \bmod n$ 是一個單向暗門函數。給定 x、k 及 n，要計算 y 是很容易的，可以使用第九章所討論的快速指數運算演算法。但給定 y、k 及 n，要計算 x 是很困難的，這是第九章所討論的離散對數問題。在此情形下，沒有多項式時間的解法來計算 $\boldsymbol{f^{-1}}$。
- 然而，如果我們知道 $k \times k^{-1} = 1 \bmod \phi(n)$ 的暗門 k^{-1}，則我們能使用 $x = y^{k^{-1}} \bmod n$ 來求出 x。這便是著名的 RSA 系統，本章稍後將會討論

Diffie-Hellman Notations

- Alice 的私鑰：$x_a \in [1, p-1]$
- Alice 的公鑰：$y_a = g^{x_a} \bmod p$
- Bob 的私鑰：$x_b \in [1, p-1]$
- Bob 的公鑰：$y_b = g^{x_b} \bmod p$
- 其中 p 跟 g 是可公開的參數

Diffie-Hellman Key Exchange Protocol

Alice

Bob

$y_a \bmod p$ →

← $y_b \bmod p$

$K = (g^{x_b})^{x_a} \bmod p$

$K = (g^{x_a})^{x_b} \bmod p$

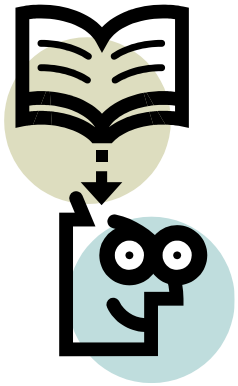
Man-in-the-Middle Attack

Alice

Bob

$g^{x_a} \pmod p$

$g^c \pmod p$



$g^c \pmod p$

$g^{x_b} \pmod p$

Random: $a \in [1, p-1]$

$K_1 = (g^c)^{x_a} \pmod p$

Random: $b \in [1, p-1]$

$K_2 = (g^c)^{x_b} \pmod p$

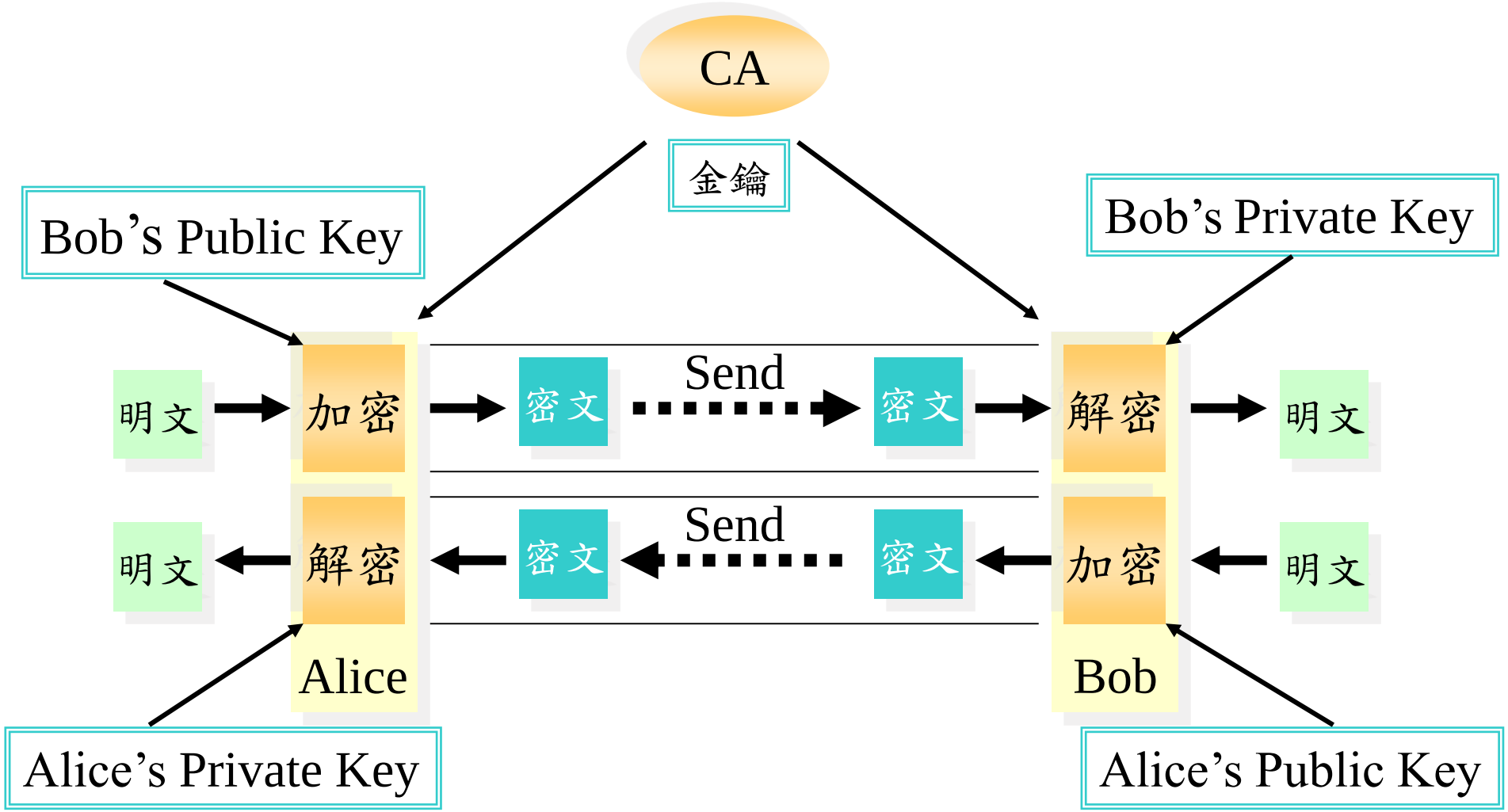
Possible Solutions

- 建立認證的安全通道，利用『前置共享密鑰』（Pre-shared Secret）來交換訊息加密
- 在 Diffie-Hellman 交換訊息之後，緊接著傳送共享數值、名稱、與前置共享密鑰的雜湊值
- 在 Diffie-Hellman 交換訊息之後，緊接著傳送前置共享密鑰與所傳送參數的雜湊值

10.2 RSA 密碼系統

- 屬於非對稱式密碼系統的一種，於 1977 年由美國麻省理工學院的三位教授 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章的功能
- RSA Encryption
  - 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
- RSA Decryption
  - 必須使用自己的私密金鑰（Private Key）才能將密文解出

公開金鑰加密系統


CA
金鑰
Bob's Public Key
Bob's Private Key
明文
加密
密文
Send
密文
解密
明文
明文
解密
密文
Send
密文
加密
明文
Alice
Bob
Alice's Private Key
Alice's Public Key

演算法10.2 RSA 金鑰產生

- 建議 p 及 q 必須至少為 512 位元；n 必須至少為 1024 位元

RSA_Key_Generation

{

Select two large primes p and q such that $p \neq q$.

$n \leftarrow p \times q$

$\phi(n) \leftarrow (p-1) \times (q-1)$

Select e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$

$d \leftarrow e^{-1} \bmod \phi(n)$ //d是e在模$\phi(n)$底下的乘法反元素

Public_key $\leftarrow$ (e, n) // 公開宣布

Private_key $\leftarrow d$ // 保持祕密

return Public_key and Private_key

}

演算法10.3 RSA 加密

```
RSA_Encryption (P, e, n)                    // P 是在 Z_n 下的明文且 P < n
{
    C ← Fast_Exponentiation (P, e, n)       //計算 (P^e mod n)
    return C
}
```

演算法10.4 RSA 解密

```
RSA_Decryption (C, d, n)                    //C是在Z_n下的密文
{
    P ← Fast_Exponentiation (C, d, n)       //計算(C^d mod n)
    return P
}
```

RSA 演算法

- Bob 選 2 個大質數 p 和 q（至少 100 個位數），令 $n = p \cdot q$，再計算 $\phi(n) = (p-1)(q-1)$，並選一個與 $\phi(n)$ 互質的數 e
  - $\phi(n)$ 為 Euler's totient 函數，其意為與 n 互質之個數
  - (e, n) 即為 Bob 的**公開金鑰**
- Bob 選 1 個數 d，滿足 $e \cdot d \bmod \phi(n) = 1$
  - d 即為 Bob 的解密金鑰（亦稱**私密金鑰**）
- 加密法為 $\mathrm{C} = \mathrm{P}^e \bmod n$
- 解密法為 $\mathrm{P} = \mathrm{C}^d \bmod n$

- **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 的證明

如果 $n = p \times q$，$a < n$，而且 k 是一個整數，則 $a^{k \times \phi(n)+1} \equiv a \pmod{n}$

$\mathrm{P}_1 = \mathrm{C}^d \bmod n = (\mathrm{P}^e \bmod n)^d \bmod n = \mathrm{P}^{ed} \bmod n$

$ed = k\phi(n) + 1$ // d 與 e 在模 $\phi(n)$ 底下互為乘法反元素

$\mathrm{P}_1 = \mathrm{P}^{ed} \bmod n$

$\rightarrow \mathrm{P}_1 = \mathrm{P}^{k\phi(n)+1} \bmod n = \mathrm{P} \bmod n$ // 尤拉定理（第二種版本）

RSA 演算法 - 例子

- Bob 選 $p = 3$，$q = 11$
  此時 $n = p \cdot q = 3 \times 11 = 33$
- Bob 選出一個與 $(p-1) \times (q-1) = (3-1)(11-1) = 20$ 互質的數 $e = 3$
- $(e, n) = (3, 33)$ 即為 Bob 的公開金鑰
- Bob 選一個數 $d = 7$ 當作解密金鑰，
  並滿足 $e \cdot d \equiv 1 \bmod 20$ $(7 \times 3 \equiv 1 \bmod 20)$
- 令明文 $\mathrm{P} = 19$
  - 加密：$\mathrm{C} = \mathrm{P}^e \bmod n = 19^3 \bmod 33 = 28$
  - 解密：$\mathrm{P} = \mathrm{C}^d \bmod n = 28^7 \bmod 33 = 19$

安全基礎—因數分解問題

- $\phi(n) = (p-1)(q-1)$，如果知悉公開參數 n 的兩個質因數 (p, q)，則 $\phi(n)$ 的值將無秘密可言
- 由於 e 為公開訊息，並且滿足 $ed = 1 \bmod \phi(n)$，因此如果 $\phi(n)$ 為外人所知悉，則可利用歐式演算法計算出私鑰 d 的值
- 換言之，如果可以有效率地執行因數分解，則 RSA 密碼系統毫無安全性可言

因數分解問題

15 =

3×5

91 =

7×13

2173 =

41×53

3776111 =

1889×1999

1522605027922533360535618378132637429718068114961380688657908
4945801229632589528976540003506920061 39 = ?

RSA 數

- RSA 數（RSA Numbers）為 RSA 質因數分解挑戰問題所使用大半質數的集合，其中從 RSA-100 到 RSA-500，數字代表的是十進制的位數，而從 RSA-576 開始，數字是以二進制的位數來命名的
- 這項挑戰從 1991 年 3 月開始，旨在鼓勵整數分解的相關研究，獎金最高可達美金 200,000 元，挑戰已於 2007 年結束

RSA-200

- 27997833911221327870829467638722601621070446786955428537
  56000992932612840010760934567105295536085606182235191095
  13657886371059544820065767750985805576135790987349501441
  78863178946295187237869221823983
- 200 digits (663 bits)
- May 9, 2005, Franke et al.
  - 2.2 GHz CPU---55 years
  - 80 2.2 GHz CPU---3 months
  - RSA 768 bits---8 months (December 12, 2009)

RSA-200

- Factors =

  35324619344027701212726049781984643686197400197625023649
  30346877612125367942320005854795652808834 9

and

  79258699544783330333470858414800596877379758573642199607
  34330341455767872818152135381409304740185467

30,000 USD. RSA-704

- 740375634795617128280467960974295731425931888892312890849362326389727650340282662768919964196251178439958943305021275853701189680982867331732731089309005525051168770632990723963807867100860969625379346505637963 59
- 704 bits (212 digits)
- July 2, 2012, Shi et al.

RSA-704

- Factors =

  909121352959781887844065830260043748589260831032835872042851216896041152864093336782495078836795675680614 1

and

  814385925911004526572780912628442933587789900216762788320091417242932436013300411670200324082877797025249 9

100,000 USD. RSA-1024

- 135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563
- 1024 bits (309 digits)

200,000 USD. RSA-2048

- 25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774447920739934236584823824281198163815010674810451660377306056201619676256133844143603833904414952634432190114657544541784240209246165157233507787077498171257724679629263863563732899121548314381678998850404536402352738195137863656439121201039712282212072035 7
- 2048 bits (617 digits)

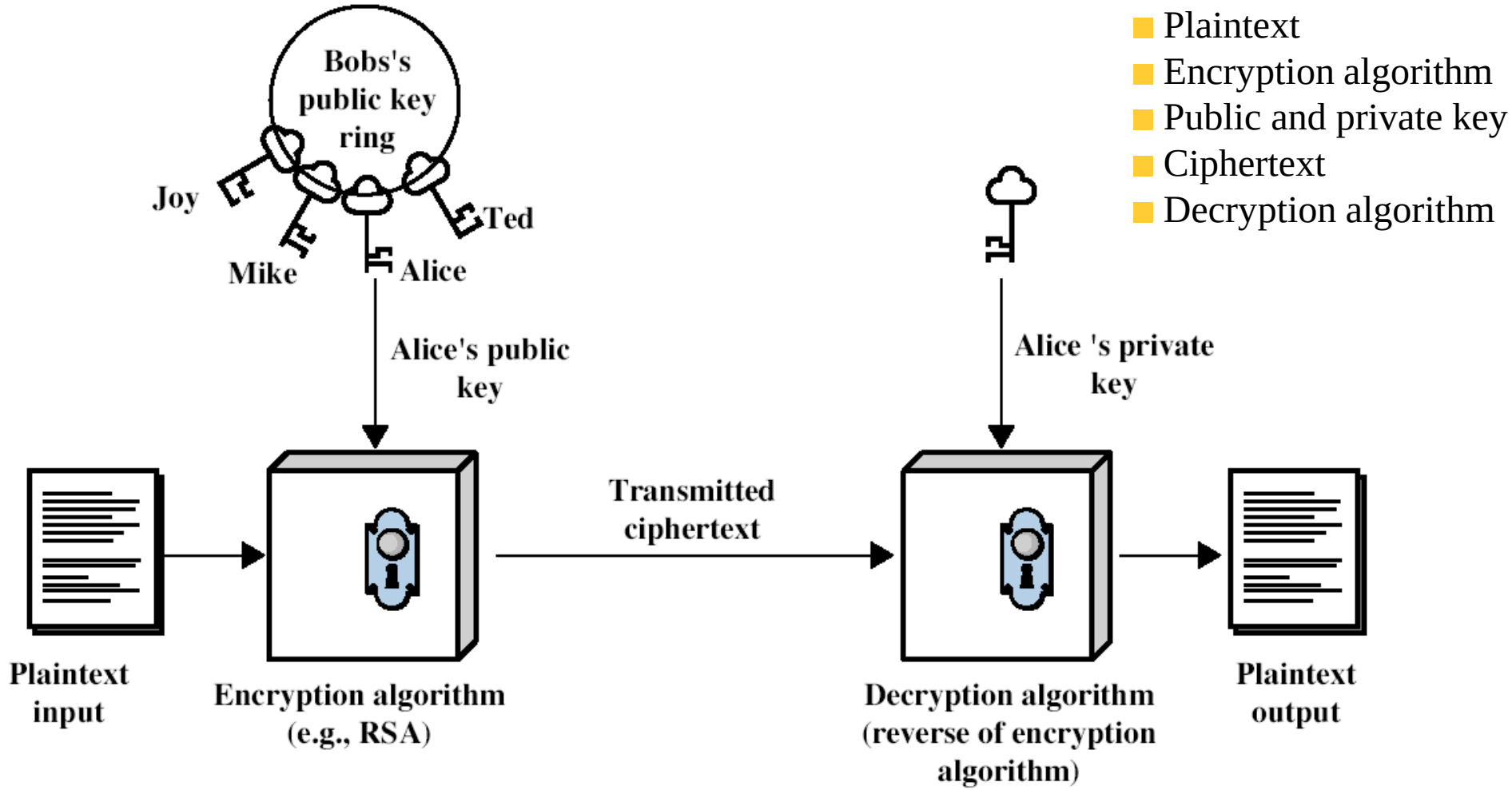
Public-Key Cryptosystems 之特性 (1/2)

1. 可還原性：$D(d, E(e, \mathrm{P})) = \mathrm{P}$
2. 金鑰產生容易：d 和 e 很容易求得
3. 單向暗門：若公開(e, n)，別人很難從(e, n)去求得 d，即只有自己知道如何解以 e 加的密
4. 可驗證性：$D(e, E(d, \mathrm{P})) = \mathrm{P}$

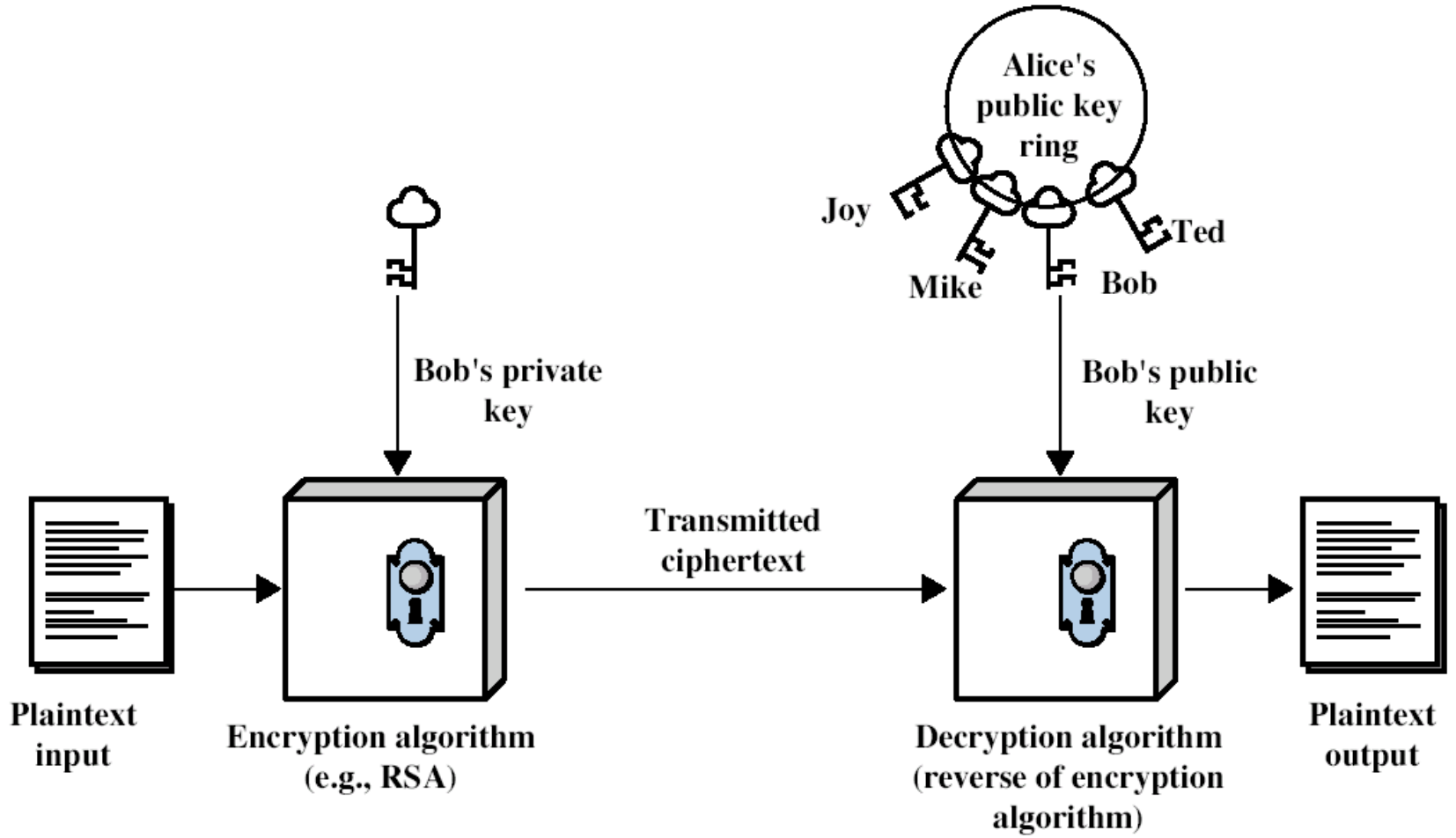
Public-Key Cryptosystems 之特性 (2/2)

- 滿足 1 ~ 3 項稱之為 Trapdoor One-Way Function
  - “One-Way”因易加密而不易解密
  - “Trapdoor”若知一些特別資訊即可解密
- 滿足 1 ~ 4 項稱之為 Trapdoor One-Way Permutation
- 1 ~ 3 項為 Public-Key Cryptosystems 之要求
- 若同時滿足第 4 項要求，則該保密法可用來製作數位簽章

Public-Key Cryptography -- Encryption



Public-Key Cryptography -- Authentication



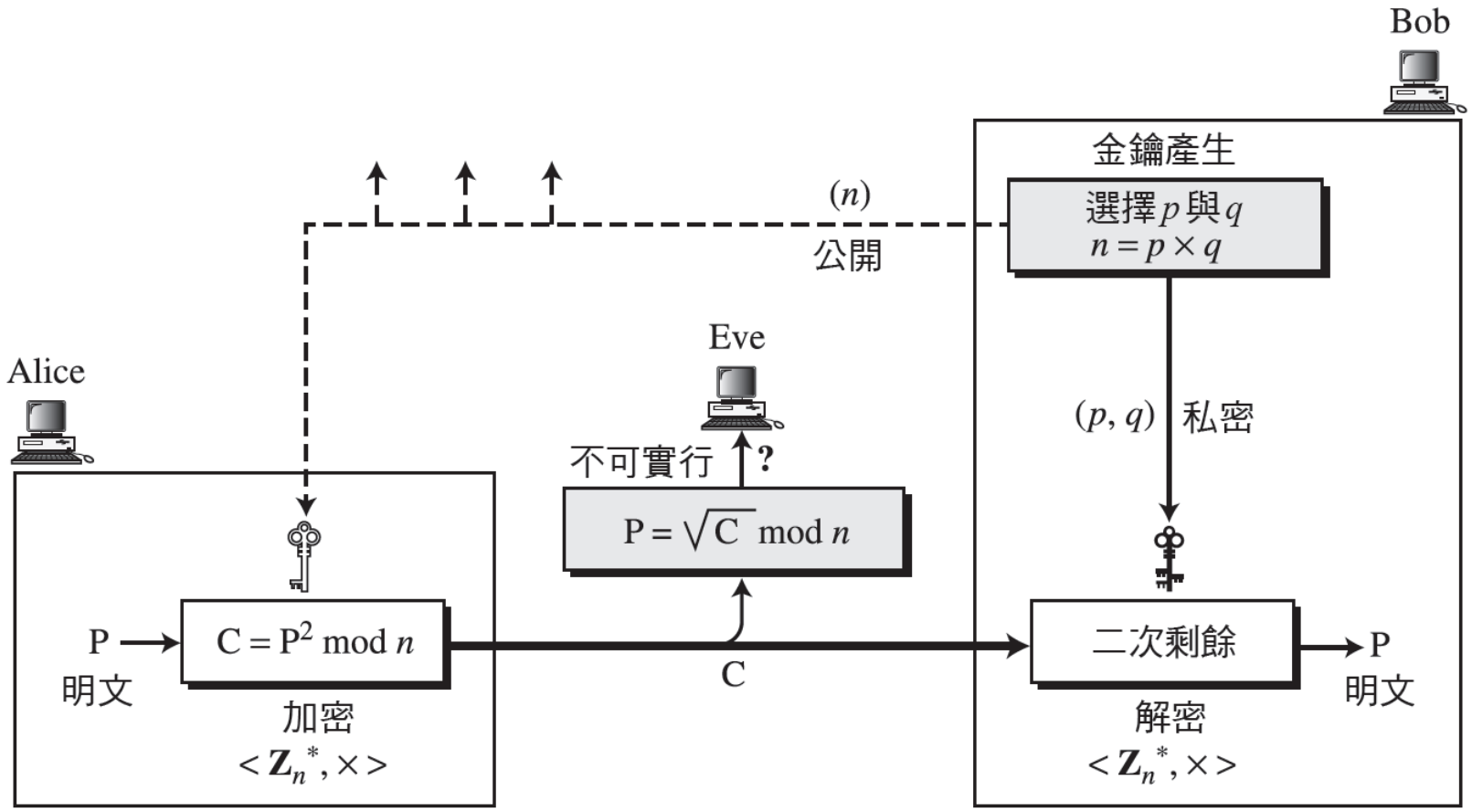
Public-Key Characteristics

- Public-key algorithms rely on two keys with the characteristics that it is:
  - Computationally infeasible to find decryption key knowing only algorithm & encryption key
  - Computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - Either of the two related keys can be used for encryption, with the other used for decryption (in some schemes)

10.3 Rabin 密碼系統

- Rabin 密碼系統可想成一個將 e 及 d 固定 的 RSA 密碼系統。加密是 $\mathrm{C} \equiv \mathrm{P}^2 \pmod{n}$，而解密是 $\mathrm{P} \equiv \mathrm{C}^{1/2} \pmod{n}$
- 本節討論主題
  - 程序
  - Rabin 系統的安全性

圖 10.10 Rabin 密碼系統的加密、解密及金鑰產生



演算法 10.6 Rabin 密碼系統的金鑰產生

- 與 RSA 系統相同，接收者 Bob 任選兩大質數 p 及 q 並求出 $n = pq$
- 公開金鑰：n
- 私密金鑰：p、q

Rabin_Key_Generation

{

Choose two large primes p and q in the form $4k + 3$ and $p \neq q$.

$n \leftarrow p \times q$

Public_key $\leftarrow$ n //公開宣布

Private_key $\leftarrow$ (q, n) //保持祕密

return Public_key and Private_key

}

演算法 10.7 Rabin 密碼系統加密

- 若 Alice 欲傳送明文 P 給 Bob，則她首先必須先取得 Bob 之公開金鑰 n（其中 $0 \leq P < n$），並求出密文 $C = P^2 \bmod n$，傳送給 Bob

```
Rabin_Encryption (n, P)          //n 是公開金鑰；P 是在 Z_n* 下的明文
{
    C ← P^2 mod n                //C 是密文
    return C
}
```

演算法 10.8 Rabin 密碼系統解密

- Bob 收到密文後，求出 C 在模 n 之四個平方根 P_1、P_2、P_3 及 P_4，則 P 必為此四根之一，可由 P 的預設格式（如後面加許多 0）來判定何者為真正明文

Rabin_Decryption (p, q, C) //C是密文；p及q是私密金鑰

{

$a_1 \leftarrow +(C^{(p+1)/4}) \bmod p$

$a_2 \leftarrow -(C^{(p+1)/4}) \bmod p$

$b_1 \leftarrow +(C^{(q+1)/4}) \bmod q$

$b_2 \leftarrow -(C^{(q+1)/4}) \bmod q$ //呼叫四次中國餘數定理演算法

$P_1 \leftarrow$ Chinese_Remainder (a_1, b_1, p, q)

$P_2 \leftarrow$ Chinese_Remainder (a_1, b_2, p, q)

$P_3 \leftarrow$ Chinese_Remainder (a_2, b_1, p, q)

$P_4 \leftarrow$ Chinese_Remainder (a_2, b_2, p, q)

return P_1, P_2, P_3, and P_4

}

10.3 Rabin 密碼系統 (續)

注意

Rabin 密碼系統是非確定式的：解密得到四個同樣都可能的明文。

範例 10.8

- 以一個非常顯而易見的例子來說明此概念
  - Bob 選擇 $p = 23$ 及 $q = 7$，注意兩個都是在模 4 下同餘為 3
  - Bob 計算 $n = p \times q = 161$
  - Bob 公布 n；他保持 p 及 q 為私密
  - Alice 想要傳送明文 $\mathrm{P} = 24$。注意 161 與 24 互質；24 在 $\boldsymbol{Z}_{161^*}$ 中。她計算 $\mathrm{C} = 24^2 = 93 \bmod 161$，並將密文 93 傳送給 Bob

範例 10.8 (續)

- Bob 收到 93，並計算下面四個值：
  - $a_1 = +(93^{(23+1)/4}) \bmod 23 = 1 \bmod 23$
  - $a_2 = -(93^{(23+1)/4}) \bmod 23 = 22 \bmod 23$
  - $b_1 = +(93^{(7+1)/4}) \bmod 7 = 4 \bmod 7$
  - $b_2 = -(93^{(7+1)/4}) \bmod 7 = 3 \bmod 7$

範例 10.8 (續)

- Bob 得到四個可能的答案 (a_1, b_1)、(a_1, b_2)、(a_2, b_1) 及 (a_2, b_2)，並使用中國餘數定理找到四個可能的明文 116、24、137 及 45（全部都與 161 互質）。注意，只有第二個答案是 Alice 的明文。Bob 必須根據情況來決定。再次注意，所有四個解答在平方再模 n 之後，都會得到 Alice 送出的密文 93

範例 10.9

- 選取 $p = 277$，$p = 331$，則 $n = 91687$
- 若要加密 $\mathrm{P} = 1001111001$，可在後方複製部分原文使之能辨識
  - 例如 $\mathrm{P} = (1001111001\mathbf{111001})_2 = (40569)_{10}$
- 密文 $\mathrm{C} = 40569^2 \bmod 91678 = 62111$

範例 10.9 (續)

- 解密時可解 $P^2 = 62111 \bmod 91678$
  - $P_1 = 1000100000010110$
  - $P_2 = 101011000010001$
  - $P_3 = 100111100\mathbf{111001}$
  - $P_4 = 1100011110101110$
- 故可知 P_3 為正確的解

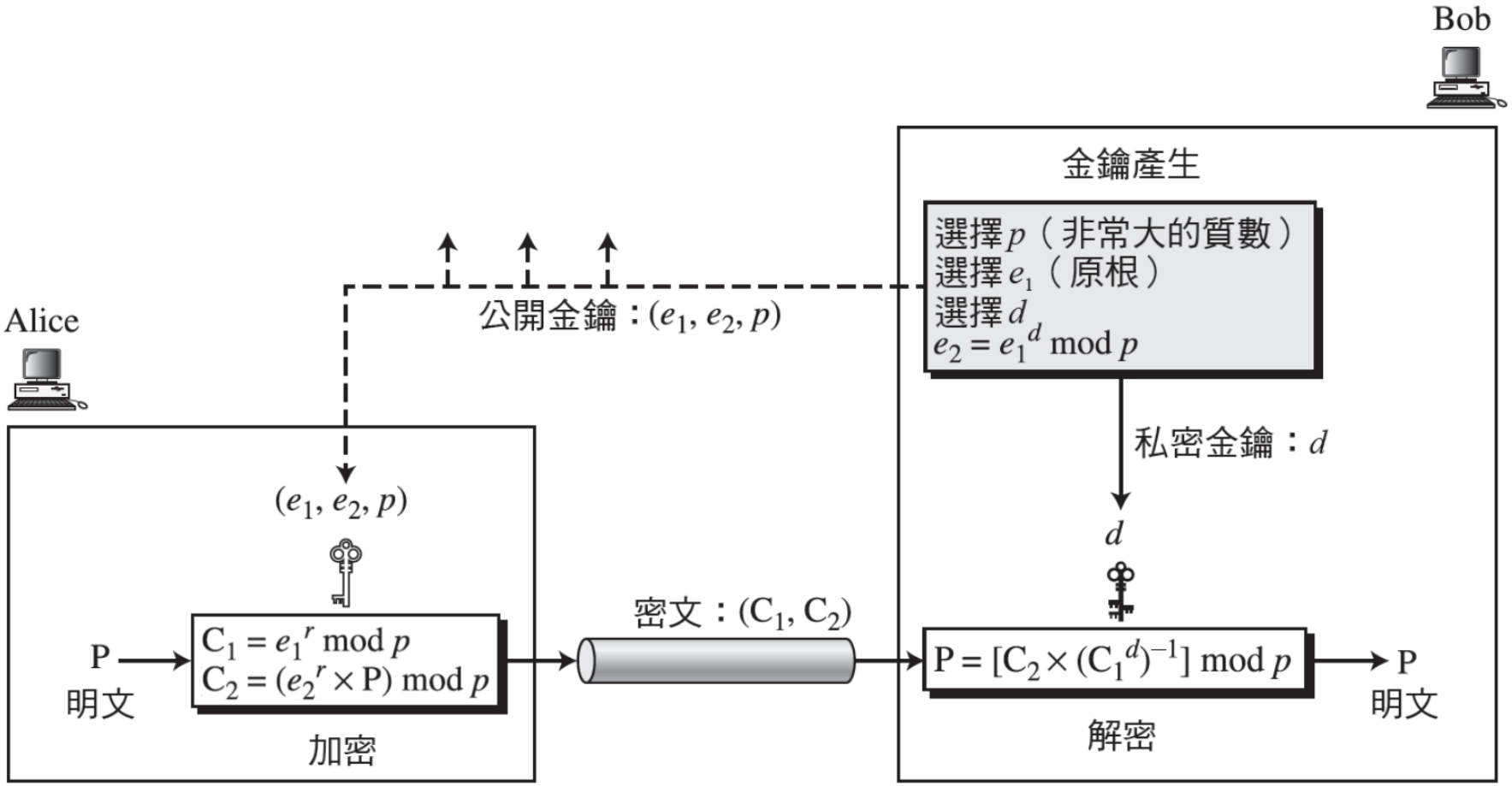
Rabin 密碼系統的安全性分析

- 本系統之安全性可證明等於分解因數，但存在選擇密文攻擊法
  - 破譯者任選一亂數 x 並求出 $x^2 = \mathrm{C} \bmod n$，破譯者將 C 送給 Bob 並要求 Bob 解密
  - 若 Bob 解密得一明文 P 送回給破密者且 $\mathrm{P} \neq \pm x \bmod n$（其機率為 $\frac{1}{2}$），則破密者可獲得 C 之 4 個平方根之兩獨立解，破解者可解因數 n，故此系統可被破解

10.4 ElGamal 密碼系統

- 除了 RSA 與 Rabin，另外一個公開金鑰密碼系統是 ElGamal 密碼系統（ElGamal Cryptosystem），根據發明者 Taher ElGamal 命名。ElGamal 是基於解離散對數問題
- 本節討論主題
  - ElGamal 密碼系統
  - 程序
  - 證明
  - 分析
  - ElGamal 的安全性
  - 應用

圖 10.11 ElGamal 系統的金鑰產生、加密及解密程序



演算法 10.9 ElGamal 系統的金鑰產生

ElGamal_Key_Generation

{

Select a large prime p

Select d to be a member of the group $\mathbf{G} = < \mathbf{Z}_p^*, \times >$ such that $1 \le d \le p - 2$

Select e_1 to be a primitive root in the group $\mathbf{G} = < \mathbf{Z}_p^*, \times >$

$e_2 \leftarrow e_1^{\,d} \bmod p$

Public_key $\leftarrow$ (e_1, e_2, p) // 公開宣布

Private_key $\leftarrow d$ // 保持祕密

return Public_key and Private_key

}

演算法 10.10 ElGamal 加密

ElGamal_Encryption (e_1, e_2, p, P) //P 是一個明文

{

Select a random integer r in the group $\mathbf{G} = < \mathbf{Z}_p^*, \times >$

$C_1 \leftarrow e_1^{\,r} \bmod p$

$C_2 \leftarrow (P \times e_2^{\,r}) \bmod p$ //C_1 及 C_2 是密文

return C_1 and C_2

}

10.4.1 ElGamal 密碼系統

注意

在 ElGamal 密碼系統中的加密或解密位元運算複雜度是多項式型態的。

範例 10.10

- 這是一個顯而易見的例子。Bob 選擇 p 為 11，於是選擇 $e_1 = 2$。注意，2 是在 $\boldsymbol{Z_{11}}^*$ 底下的原根
- 然後，他選擇 $d = 3$ 並計算 $e_2 = e_1^d = 8$。因此公開金鑰為 (2, 8, 11)，而私密金鑰為 3。Alice 選擇 $r = 4$，並計算明文 7 的密文 C_1 及 C_2

範例 10.10 (續)

明文：7

$C_1 = e_1^r \bmod 11 = 16 \bmod 11 = 5 \bmod 11$

$C_2 = (P \times e_2^r) \bmod 11 = (7 \times 4096) \bmod 11 = 6 \bmod 11$

密文：(5, 6)

- Bob 收到密文（5 及 6）並計算明文

密文： $[C_2 \times (C_1^d)^{-1}] \bmod 11 = 6 \times (5^3)^{-1} \bmod 11 = 6 \times 3 \bmod 11 = 7 \bmod 11$

明文：7

範例 10.11

- 不使用 $\mathrm{P} = [\mathrm{C}_2 \times (\mathrm{C}_1^d)^{-1}] \bmod p$ 來解密，我們可以避免計算乘法反元素，使用 $\mathrm{P} = [\mathrm{C}_2 \times \mathrm{C}_1^{p-1-d}] \bmod p$（參見上一章之費瑪小定理）。在範例 10.10 中，我們能計算 $\mathrm{P} = [6 \times 5^{11-1-3}] \bmod 11 = 7 \bmod 11$

10.4.5 ElGamal 的安全性

注意

若希望 ElGamal 密碼系統是安全的，p 必須至少 300 位數字，而且對每一次加密，r 都必須採用一個全新的數值。

範例 10.12

- 這裡有一個較為實際的例子。Bob 使用一個 512 位元的隨機整數（理想值是 1024 位元）。整數 p 為 155 位數字（理想值是 300 位數字）。於是，Bob 選擇 e_1、d 並計算 e_2，如以下所示：Bob 宣稱 (e_1, e_2, p) 為他的公開金鑰，並保持 d 為私密金鑰

$p =$	11534899272561676244925313717014331740490094532609834959814346921905689869862264593212975473787189514436889176526473093615929993728061165964347353440008577
$e_1 =$	2
$d =$	1007
$e_2 =$	97886413043009189508766856938097739043880062887337687610022062233255450707415618921231831770461014167336015088413294085724853770315820660100725587074 55

範例 10.12 (續)

- Alice 將明文 P = 3200 傳送給 Bob。她收到 $r = 545131$，計算 C_1 與 C_2，並將它們傳送給 Bob

P =	3200
***r* =**	545131
C_1 =	88729706938352847102257047149227566312026006725656212501818835142941722359971268111410536366170517305158153318916540097373635508029573678785690606191528881
C_2 =	70845433304892994457701601238079499956743602183619244696177450692124469615516580077945559308034588961440240859952591957920972162887968135058277956643029 50

範例 10.12 (續)

- Bob 計算明文 $\mathrm{P} = \mathrm{C}_2 \times (\mathrm{C}_1)^{p-1-d}$

P =	3200

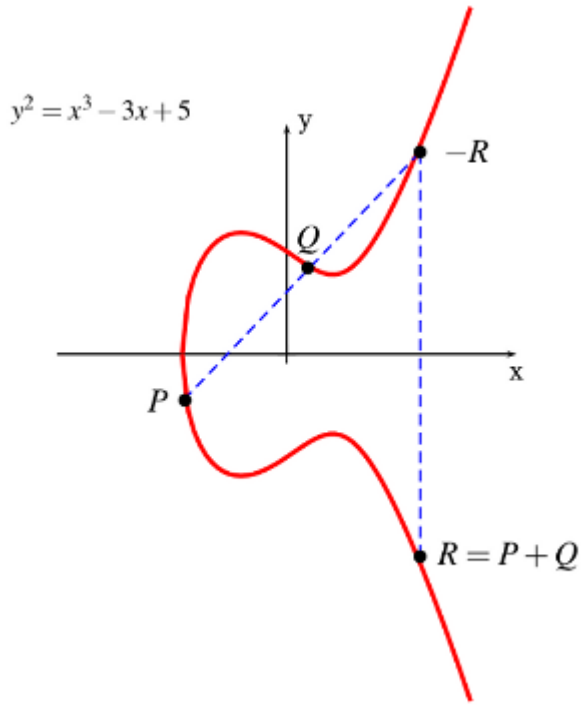
橢圓曲線密碼學

- 橢圓曲線密碼學（Elliptic Curve Cryptography，ECC）是一種基於橢圓曲線數學的公開金鑰加密演算法，於 1985 年分別由 Neal Koblitz 與 Victor Miller 各自提出的
- ECC 的主要優勢是可以使用比 RSA 演算法還要小的金鑰長度，並且提供相當等級的安全性，適合用於計算效能相對較差的輕量級裝置上

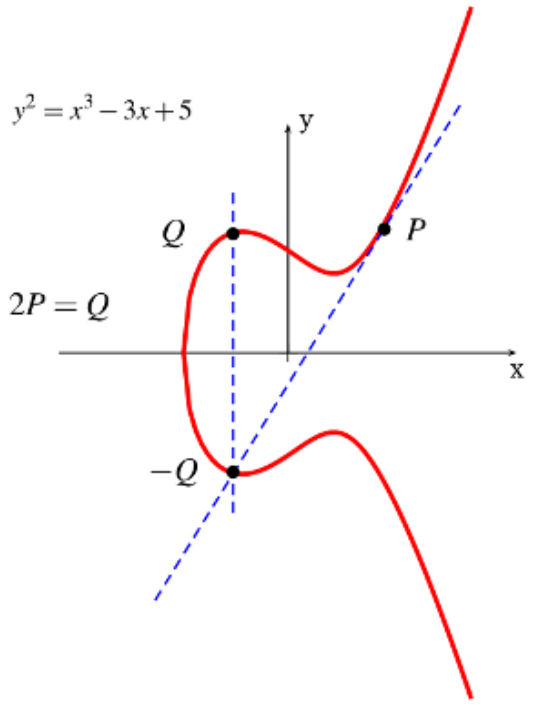
Elliptic Curve 橢圓曲線 (1/2)

- 橢圓曲線所蘊含豐富且深奧的理論已被數學家研究超過 150 年
- 橢圓曲線方程式：$y^2 = x^3 + ax + b$
  - 此曲線剛好對稱於 x 軸（$y = 0$ 這條直線）
  - 參數 a 和 b 必須滿足 $4a^3 + 27b^2 \neq 0$，才能確保沒有重根，具有唯一解
  - 加法單位元素 O 為一無窮遠的點，滿足 $O = -O$，亦須滿足橢圓曲線上某三點共線的合為 O
- 橢圓曲線的特性
  - 曲線上的任何點都以 x 軸反射（$y = 0$），並且仍是同樣的曲線（奇特的對稱性）
  - 任何不垂直的線穿過曲線最多只會有三個交點

Elliptic Curve 橢圓曲線 (2/2)



Point Addition



Point Doubling

Image Credit: http://www.embedded.com/design/safety-and-security/4396040/An-Introduction-to-Elliptic-Curve-Cryptography

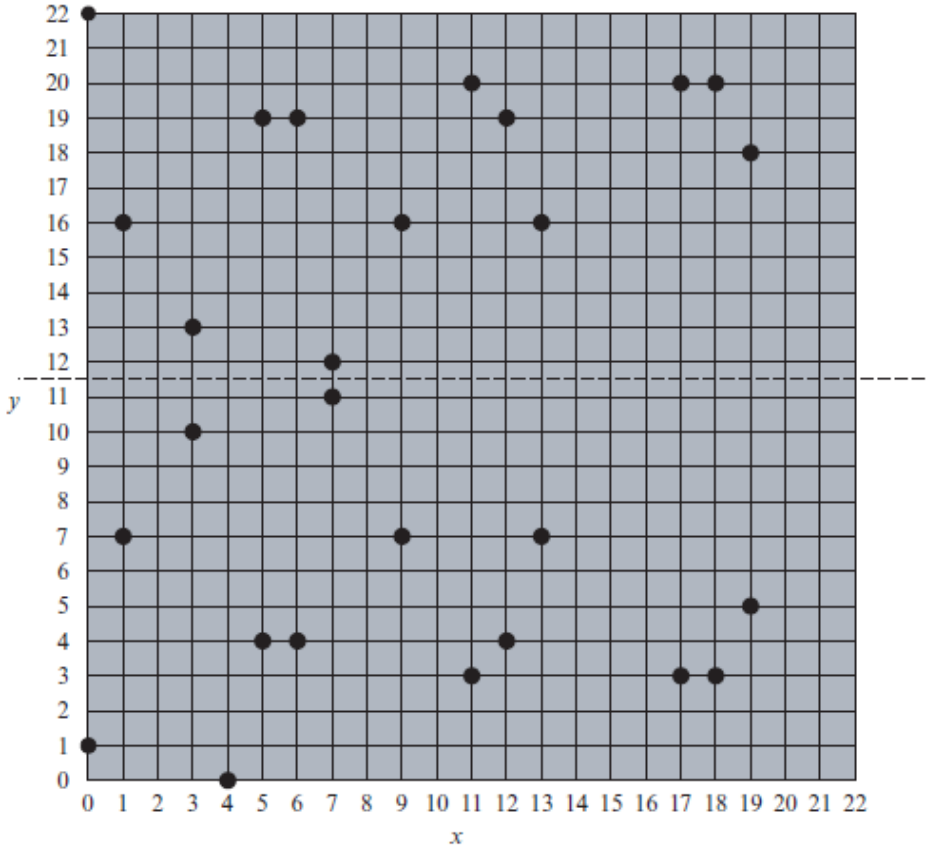
有限體和離散對數 (1/2)

- 因為橢圓曲線是連續的，容易被推算，因此，並不適合用於加密
- 把橢圓曲線定義在有限體上，這時就會用到以質數為模數的蓋洛瓦體 $GF(p)$
  - 假設橢圓曲線為 $y^2 = x^3 + x + 1$，其在有限體 $GF(p)$ 上時，可表示為：$y^2 = x^3 + x + 1 \pmod{23}$，也記作 $E_{23}(1,1)$
  - 此時，橢圓曲線不再是一條光滑連續曲線，而是一些離散的點

有限體和離散對數 (2/2)

- $E_{23}(1,1)$ 的點如下表與右圖所示

(0, 1)	**(6, 4)**	**(12, 19)**
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)



橢圓曲線在有限體上的算術計算 (1/2)

- Point Addition:
  - 給定兩個點 $P = (x_1, y_1)$ 和 $Q = (x_2, y_2)$，求 $P + Q$
  - 若 $P = Q$，則使用 Doubling 的公式；否則，計算斜率 $\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod p$
  - 假設 Addition 後的新點為 $(x_3, y_3) = P + Q = (x_1, y_1) + (x_2, y_2)$，其計算方式如下：

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \bmod p \\ y_3 = \lambda(x_1 - x_3) - y_1 \bmod p \end{cases}$$

  - 特殊情況
    - 點 P 與無窮遠點 O 相加，其結果為點 P 本身（$P + O = P$）
    - 點 P 與其負點 $-P$ 相加，其結果為無窮遠點 O（$P + (-P) = O$）

橢圓曲線在有限體上的算術計算 (2/2)

- Point Doubling:
  - 給定一個點 $P = (x_1, y_1)$
  - 計算斜率 $\lambda = \frac{3x_1^2 + a}{2y_1} \bmod p$
  - 假設 Doubling 後的新點為 $(x_3, y_3) = 2P$，其計算方式如下

$$\begin{cases} x_3 = \lambda^2 - 2x_1 \bmod p \\ y_3 = \lambda(x_1 - x_3) - y_1 \bmod p \end{cases}$$

橢圓曲線加解密演算法

- 設私鑰與公鑰分別為 k、Q，即 $Q = kG$，其中 G 為基點
- 公鑰加密：
  - 選擇隨機數 r，將訊息 M 生成密文 C，該密文是一對點，即 $C = \{rG, M + rQ\}$，其中 Q 為公鑰
- 私鑰解密：
  - $M + rQ - k(rG) = M + r(kG) - k(rG) = M$，其中 k 與 Q 分別為私鑰與公鑰
- 在橢圓曲線上，若已知 G 和 kG，則求 k 是非常困難的，此即為解橢圓曲線離散對數（Elliptic Curve Discrete Logarithm Problem，ECDLP）的難題

Security of Elliptic Curve Cryptography

- ECC 的安全性是基於解橢圓曲線離散對數的難題
  - 在給定橢圓曲線上的一點 P 和另一點 $Q = kP$，已知 Q 和 P，求解 k
- 目前已知的快速解法為 1975 年 John Pollard 所提的 Pollard's rho 演算法
- 在與因數分解難題相同的安全級別下，可以使用比 RSA 小很多的金鑰大小

各加密演算法金鑰大小與破解困難度

- NIST SP-800-57

Symmetric Key Algorithms	Diffie–Hellman, Digital Signature Algorithm	RSA (size of n in bits)	ECC (modulus size in bits)
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15{,}360$ $N = 512$	15,360	512 +

Note: L = size of public key, N = size of private key.

總結 ECC 的優勢

- 安全性更高
  - 160 位元 的 ECC 與 1024 位元的 RSA 有相同的安全强度
- 處理速度更快
  - 在計算速度上，ECC 比 RSA 快得多
- 頻寬要求更低
- 儲存空間更小
  - ECC 的密鑰大小參數，與 RSA 相比要小得多