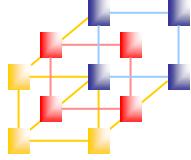


Unit 2

Cryptography



Quiz 2-討論



- 有關Keyless Relay Attack是主動攻擊還是被動攻擊？

11101 主動攻擊派：47票 / 被動攻擊派：19票

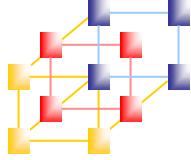
11102 主動攻擊派：36票 / 被動攻擊派：20票

11201 主動攻擊派：56票 / 被動攻擊派：9票

11202 主動攻擊派：25票 / 被動攻擊派：33票

11301 主動攻擊派：47票 / 被動攻擊派：22票

11302 主動攻擊派：35票 / 被動攻擊派：22票



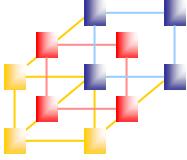
Quiz 2-討論



- 有關Keyless Relay Attack是主動攻擊還是被動攻擊？

主動攻擊派：35票（另有1票兩者）

- 因為雖然被動攔截訊號，但仍有主動擴散出去
- 因為攻擊者透過特定方式取的非公開資訊解鎖
- 因為攻擊者主動攔截訊號，並重新發射
- 因為兩人一組鎖定，直接與被害者接觸
- 因為主動靠近被攻擊方
- 因為主動取得受害者特徵，並非單純旁聽
- 因為攻擊者欺騙鑰匙與汽車使兩者誤認彼此靠近



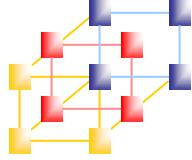
Quiz 2-討論



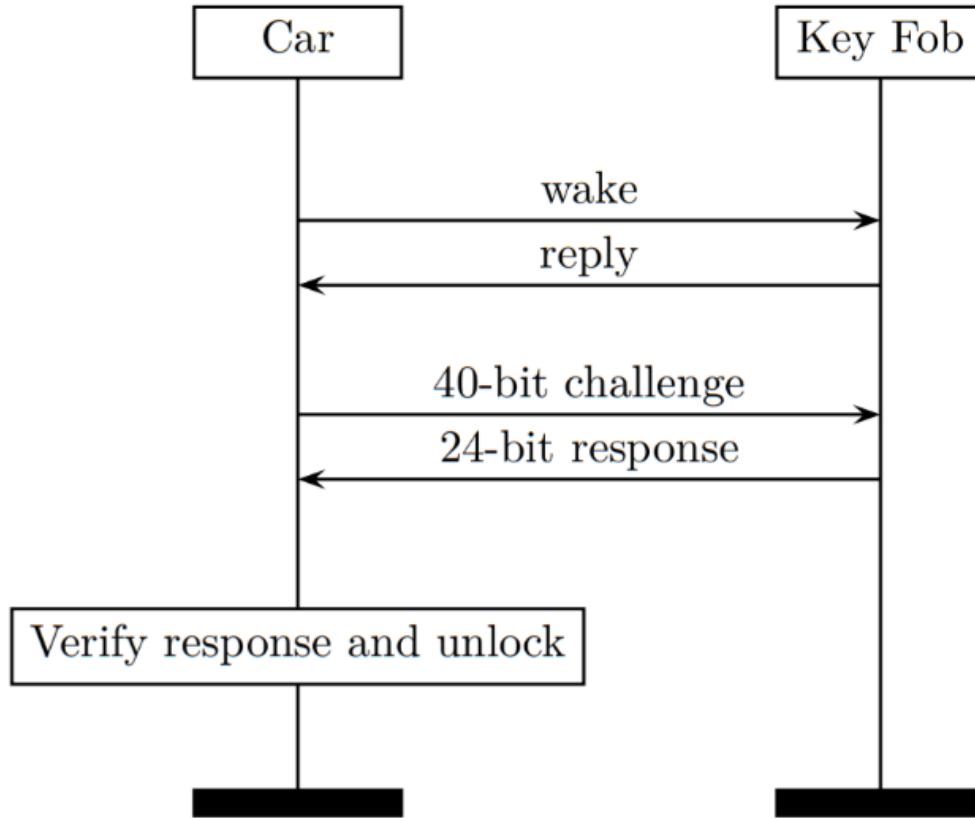
- 有關Keyless Relay Attack是主動攻擊還是被動攻擊？

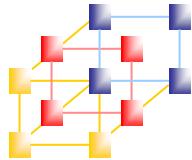
被動攻擊派：22票（另有1票兩者）

- 因為過程中沒有對訊號進行竄改或損毀
- 因為未對鑰匙或車子進行竄改
- 因為沒有破壞，受害者也未察覺，主要是攔截訊號
- 因為只有竊取資訊沒有竄改
- 因為是接收而非發起
- 因為沒有偽造或修改訊號本身
- 因為沒有對於鑰匙進行改動



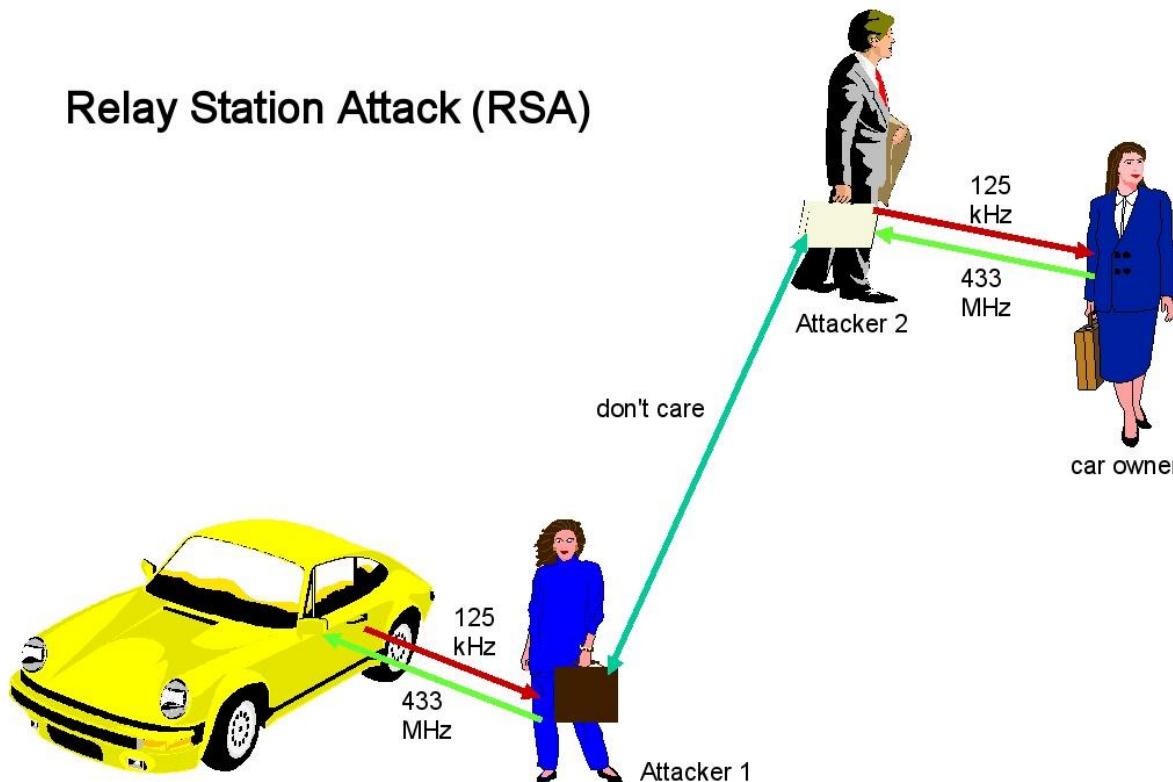
Keyless的通訊協定

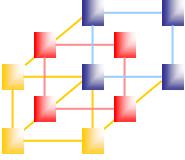




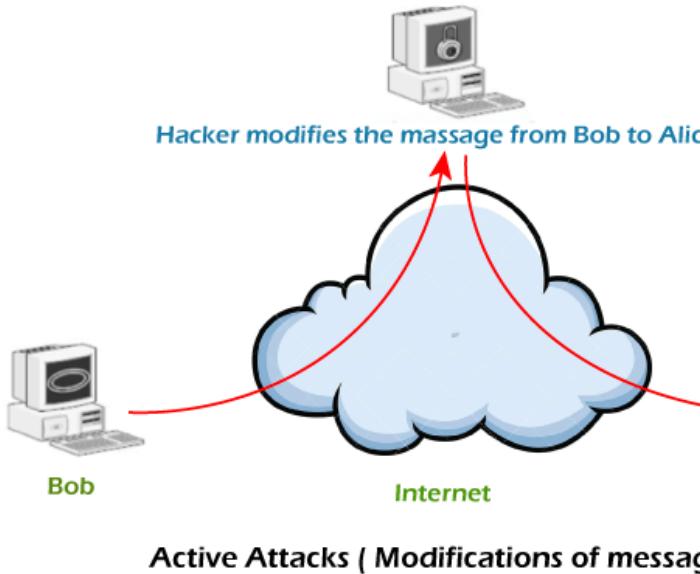
Relay Attack and Replay Attack

- https://en.wikipedia.org/wiki/Relay_attack

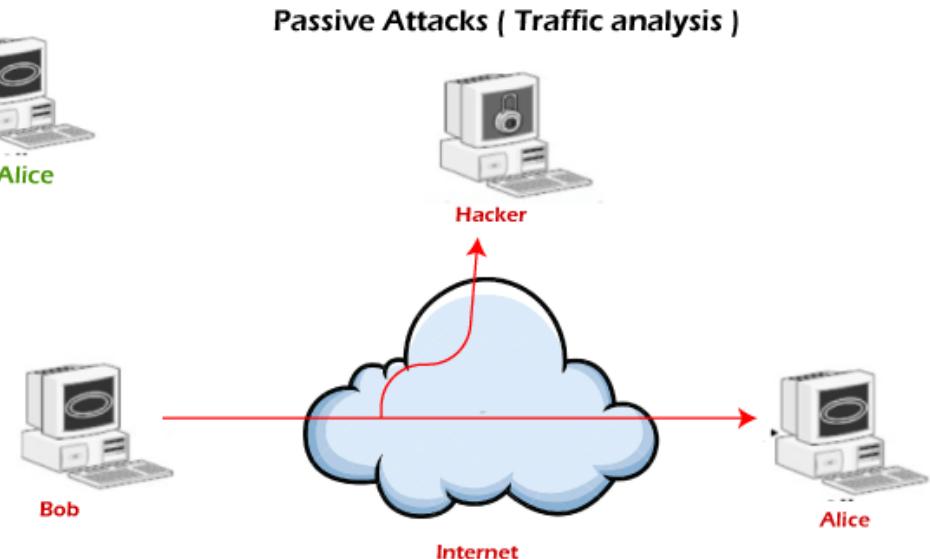




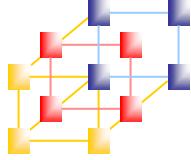
複習一下



1. 把Key發出的訊息收下來，變大，丟出去給Car
2. 把Car回應的訊息收下來，變大，丟給Key



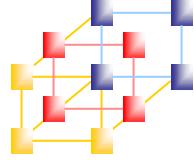
(Source: <https://www.javatpoint.com/active-attack-vs-passive-attack>)
Information and Network Security



另一種攻擊方式



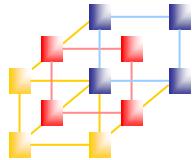
- <https://www.youtube.com/watch?v=aV1YuPzmJoY>
- <https://www.esat.kuleuven.be/cosic/news/fast-furious-and-insecure-passive-keyless-entry-and-start-in-modern-supercars/>



Quiz 2-延伸

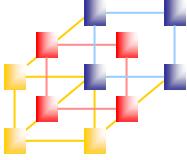


- 這是主動攻擊還是被動攻擊？



Outline

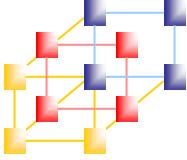
- 加密的基本概念
- 對稱式加解密法
- 非對稱式加解密法
- 雜湊函數
- 數位簽章



提了很多次的三大資安需求

- CIA

- 機密性(Confidentiality)
 - 避免非法的人看到資料
- 完整性(Integrity)
 - 避免非法的人竄改資料
- 可用性(Availability)
 - 讓合法的人想用就可以用



簽章與加密



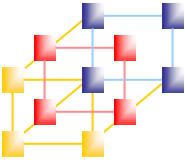
- 使用密碼學技術達到以下目標：

- 機密性(Confidentiality) → 加密
- 完整性(Integrity) → 簽章

How to do it?!

我們要戲說從頭...

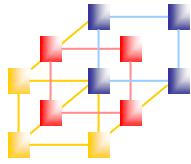




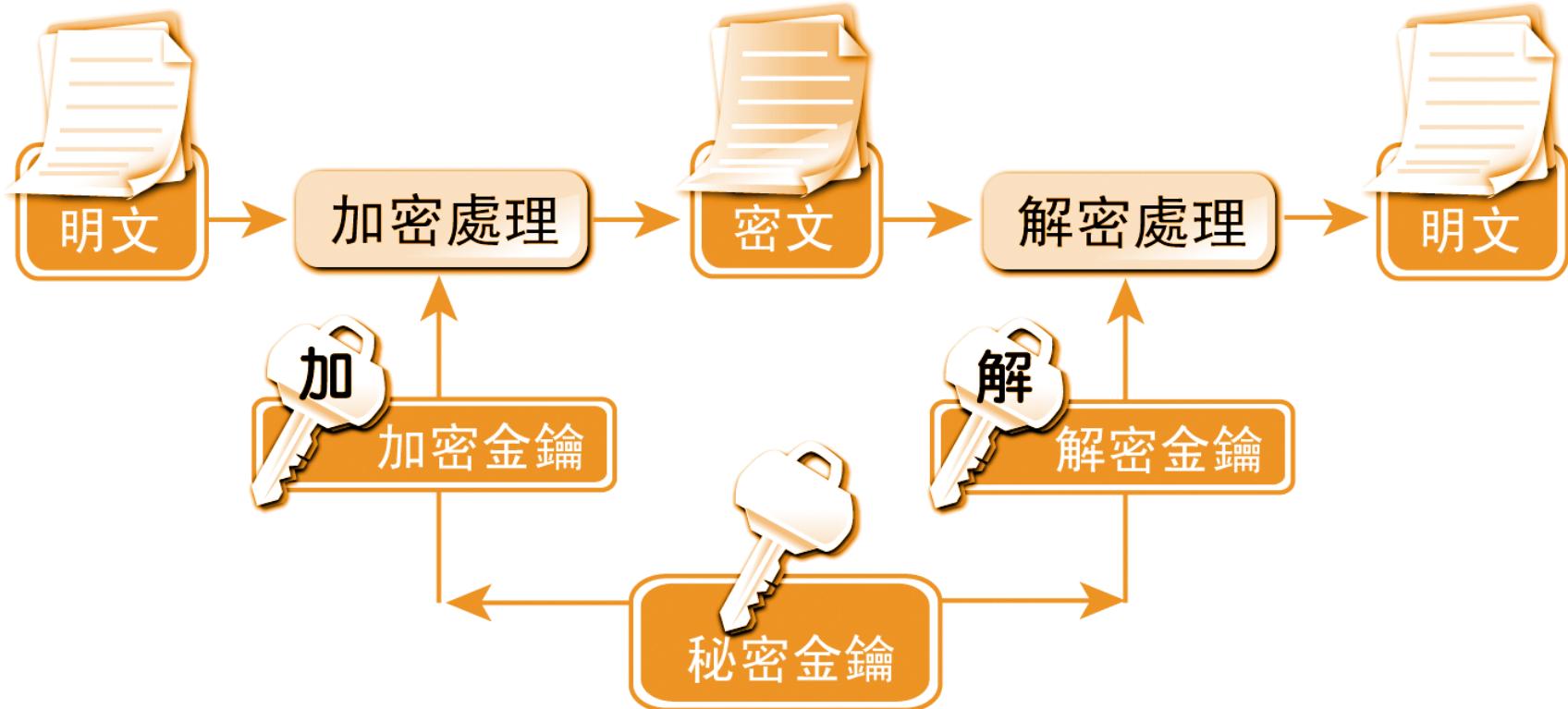
加密的基本概念

■ 加密

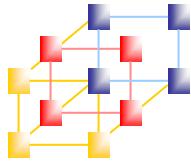
- 將資訊 (Information) 打散、避免無權檢視資訊內容的人看到資訊的內容的一種方式，而獲得授權的人，可看到資訊的內容。
- 所謂『獲得授權的人』是指擁有解密金鑰 (key) 的人。
- 透過網路來傳遞機密資訊，很容易就被竊聽。因此，對於機密資料存入於磁碟、備援磁帶、或經由網路傳遞前，應先加密成密文，使一般未經授權人員不能得知其內容。



密碼學基本概念



基本的加解密系統



密碼學基本概念

E ：加密演算法

D ：解密演算法

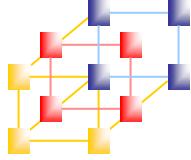
K ：金鑰

C ：密文

M ：明文

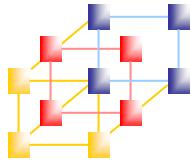
加密公式： $C = E_K(M)$

解密公式： $D_K(C) = D_K(E_K(M)) = M$



公開的加密方法或演算法

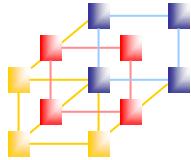
- 較不佔空間
 - 不需再儲存相關的加解密程式
- 即使為保密的加解密演算法也難保其安全
- 相容性的問題
 - 不需為每一對加解密的使用者準備一組加解密程式



密碼系統之安全性程度

- 無條件安全(Unconditionally Secure)
 - 非法使用者不管截獲多少個密文，用盡各種方法還是沒有足夠資訊可以導出明文機密資料。
 - Ex: 一次性密碼系統 (One-Time Pad)
- 計算安全(Computationally Secure)
 - 目前或未來預測之科技、以合理之資源設備下，要破解密碼系統需要一段相當長的時間(例如數百年)

基本上目前看到的大部分加密系統都只能達到計算安全

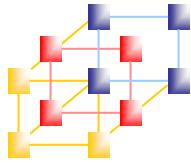


什麼是好的加解密方法？

- 前一份投影片有提到...
 - 柯克霍夫原則 (Kerckhoff's principle)

A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

一個加密系統中就算除了金鑰外的所有細節都已公開，也仍要維持其安全性



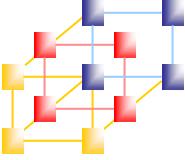
密碼系統的分類

- 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

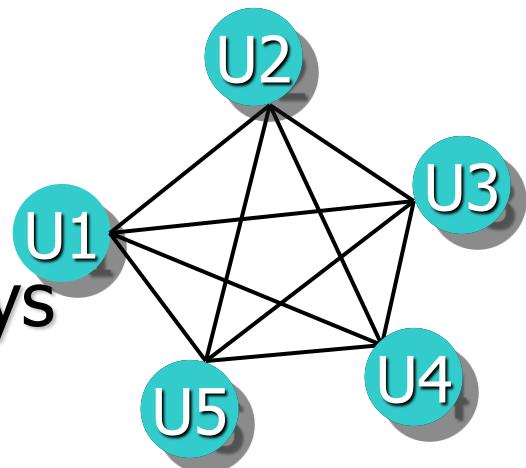


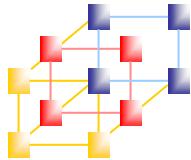
對稱性密碼系統

秘密金鑰密碼系統(Secret-Key Cryptosystems)
又稱單金鑰密碼系統(One-Key Cryptosystems)
也稱對稱密碼系統(Symmetric Cryptosystems)

優點：加解密速度快

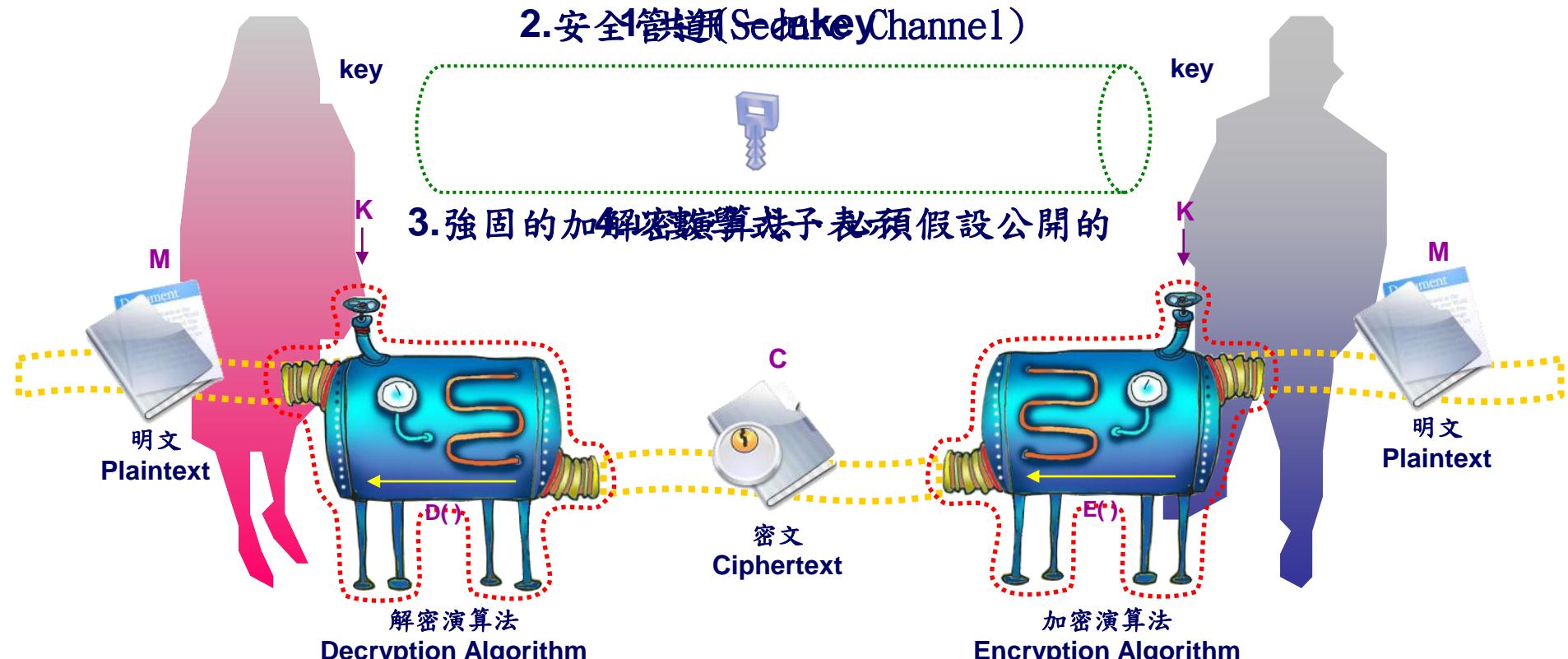
缺點：有金鑰管理的問題
每位使用者需儲存 $n-1$ 把Keys





對稱式加密(Symmetric Encryption)

又稱為Conventional / Private-key / Single-key Encryption

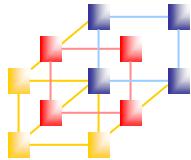


1.共用一把key

2.安全管道

3.強固加密演算法、必須假設是公開的

4.以數學公式表示之 $C=E_K(M), M=D_K(C)$



無條件安全密碼系統

One-time Pad 加密方法

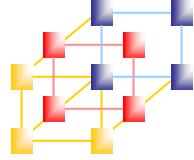
加密

$$\begin{aligned} c &= k \oplus m \\ &= 001111000111000 \oplus 1001101101010011 \\ &= 1010010101101011 \end{aligned}$$

解密

$$\begin{aligned} m &= k \oplus c \\ &= 001111000111000 \oplus 1010010101101011 \\ &= 1001101101010011 \end{aligned}$$

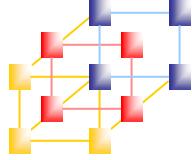
1. 加解密金鑰使用一次即丟
2. 需擁有一份與明文長度相同或更長的金鑰



One Time Pad



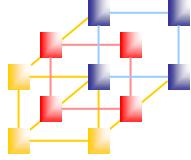
- <https://www.youtube.com/watch?v=F1IG3TvQCBQ>



運作方式



- Block Cipher vs Stream Cipher



古典密碼系統



凱薩加密(ROT 13)

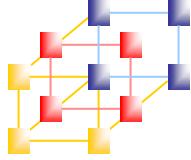
→ Substitution Cipher

- <https://www.youtube.com/watch?v=pIt4Q68J00A>

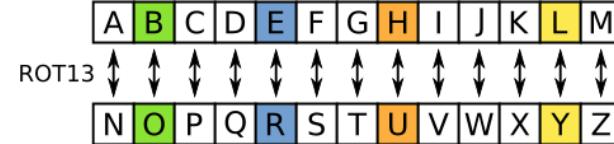
移位加密

→ Permutation Cipher

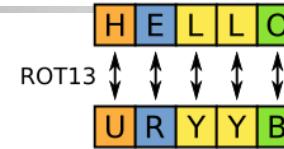
- https://www.youtube.com/watch?v=0um-_4SvPg0
- <https://www.youtube.com/watch?v=bcyUJK1BvHw>



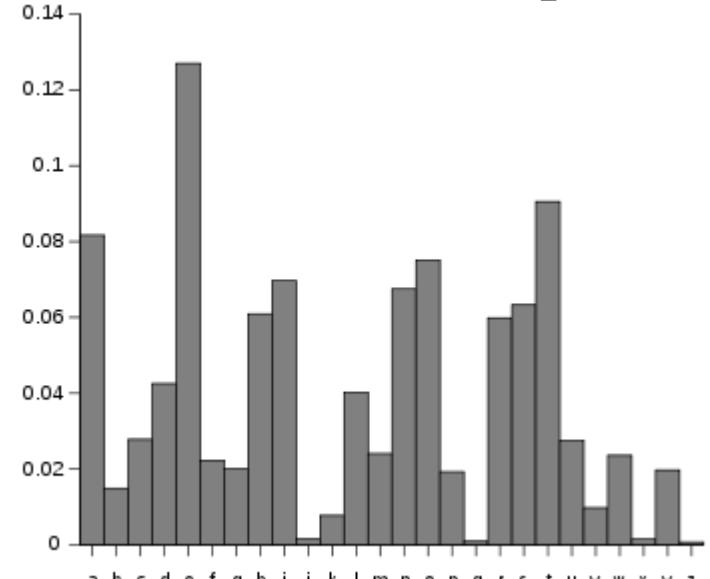
古典密碼系統



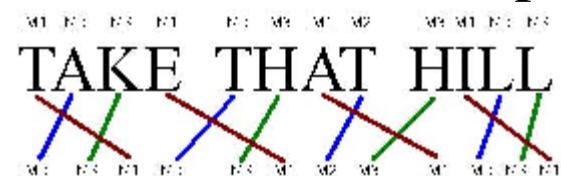
- 替換 (Substitution)
 - 把字換掉



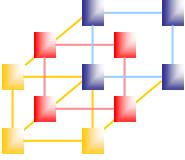
Source: Wikipedia



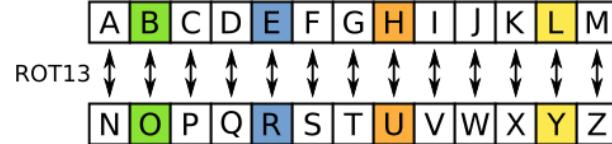
Source: Wikipedia



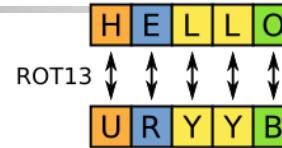
Source: KMU AKTT HETH ALLI



Quiz 3

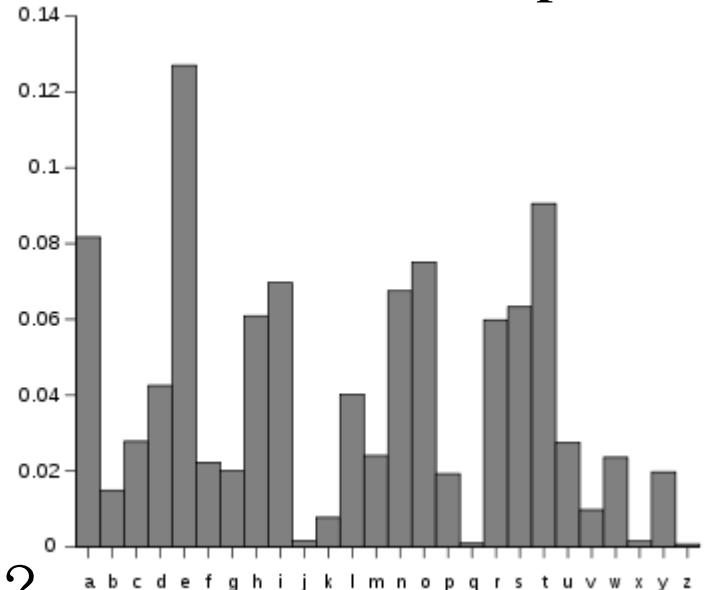


- 替換 (Substitution)
 - 把字換掉



Source: Wikipedia

- 移位 (Permutation)
 - 把順序換掉
 - 藏頭詩

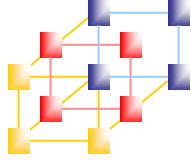


Source: Wikipedia

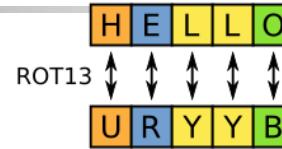
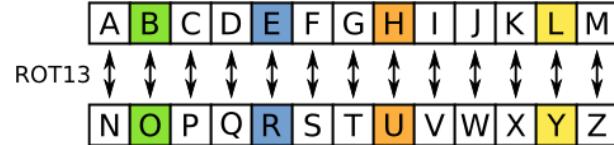
- 您會怎摸破解這兩種加密法？
- 請將答案寫在紙條上
 - ((請記得寫上班級姓名學號

TAKE THAT HILL
M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12 M13

Source: KMU AKTT HETH ALLI

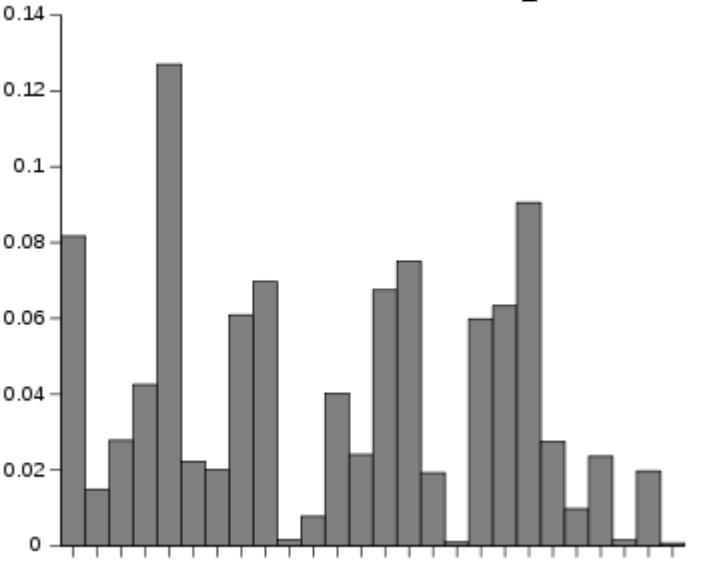


古典密碼系統



Source: Wikipedia

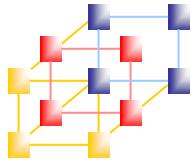
- 替換 (Substitution)
 - 把字換掉
→ 破解方式：統計出現機率
- 移位 (Permutation)
 - 把順序換掉
 - 藏頭詩
 - 破解方式：重新排列組合



Source: Wikipedia



Source: KMU AKTT HETH ALLI

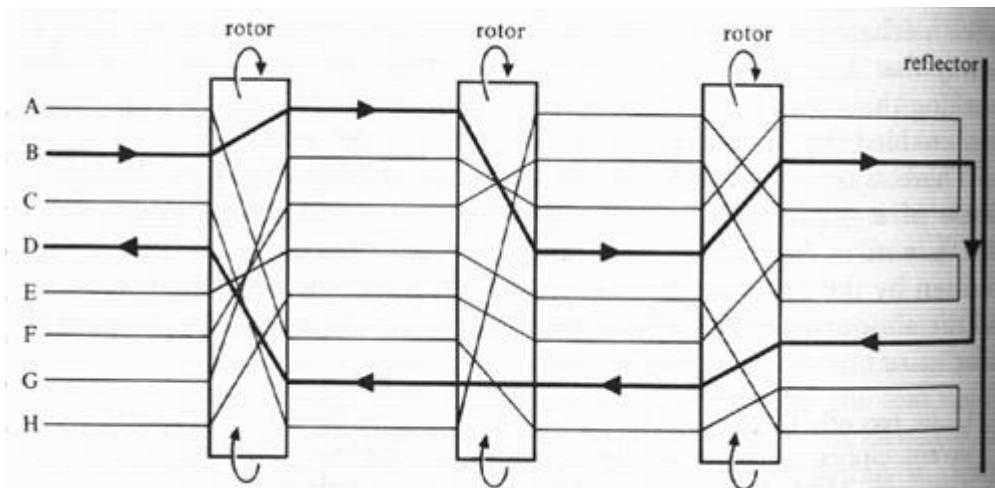


古典密碼系統

■ 混合加密法 (Product Cipher)

■ MIX!!!

旋轉加密機 / Enigma



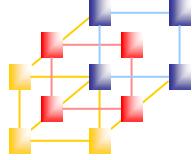
Source:cs.trincoll.edu



Source:Wikipedia

<https://www.youtube.com/watch?v=faRfab9Yyk8>
<https://www.youtube.com/watch?v=-qcOCBfRRzg>

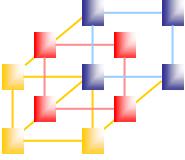
PS. 有空可以去看模彷遊戲
Information and Network Security



Enigma



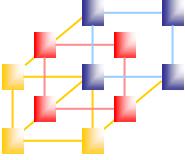
- <https://www.youtube.com/watch?v=QwQVMqfoB2E>



古代到近代



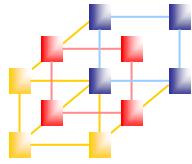
- 傳統數據保密技術並不能保證系統的安全，而且應用上也受到相當大之限制
 - Ex: 中文系統就很難用換位法以達到保密效果
 - “今天我不回家” → “回不天家我今”
 - 該密文一看就可重組成明文而不需藉其它工具來破解，所以我們需要使用一些以“位元”為處理單位的加密系統。



古代到近代

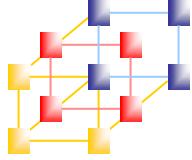


- 看起來混合加密法 (Product Cipher)有很多好處
 - 沒辦法從統計資訊看到秘密
 - 同一個字母加密多次會產生不同密文
 - 把排列方式換掉又是一條好漢
- 仍然是混合加密法 (Product Cipher)
 - 取代 (Substitution) + 移位 (Permutation)
 - 加密解密同一把Key，也叫對稱式加密系統



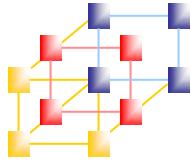
對稱式加解密法

- DES (Data Encryption Standard)
- Triple DES => DES做三次！ !
- AES (Advanced Encryption Standard)
- ...



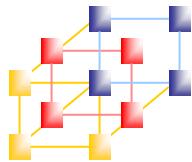
DES加密系統 (1/2)

- DES (Data Encryption Standard)
 - 對稱式加密系統之代表
 - 1970年代中期由IBM公司所發展
 - 美國國家標準局公佈為資料加密標準的一種區塊加密法(Block Cipher)
 - DES 屬於區塊加密法，而區塊加密法就是對一定大小的明文或密文來做加密或解密動作
 - 每次加密解密的區塊大小均為 64 位元(Bits)



DES加密系統 (2/2)

- DES (Data Encryption Standard)
 - 就一般資料而言，資料通常大於64位元。
 - 只要將明/密文中每64位元當作一個區塊加以切割，再將每個區塊做加密或解密即可
 - 最後一個區塊大小可能小於64位元，此時就要將此區塊附加 ”0” 位元，直到區塊大小成為64位元為止
 - DES 所用加密或解密金鑰也是64位元大小。但其中有8個位元是用來做錯誤更正，**真正的金鑰有效長度只有 56 位元**



DES 加密流程

Final Permutation

40	08	48	16	56	24	64	31
39	07	47	15	55	23	53	30
38	06	46	14	54	22	62	30
37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28
36	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26
33	01	41	09	49	17	57	25

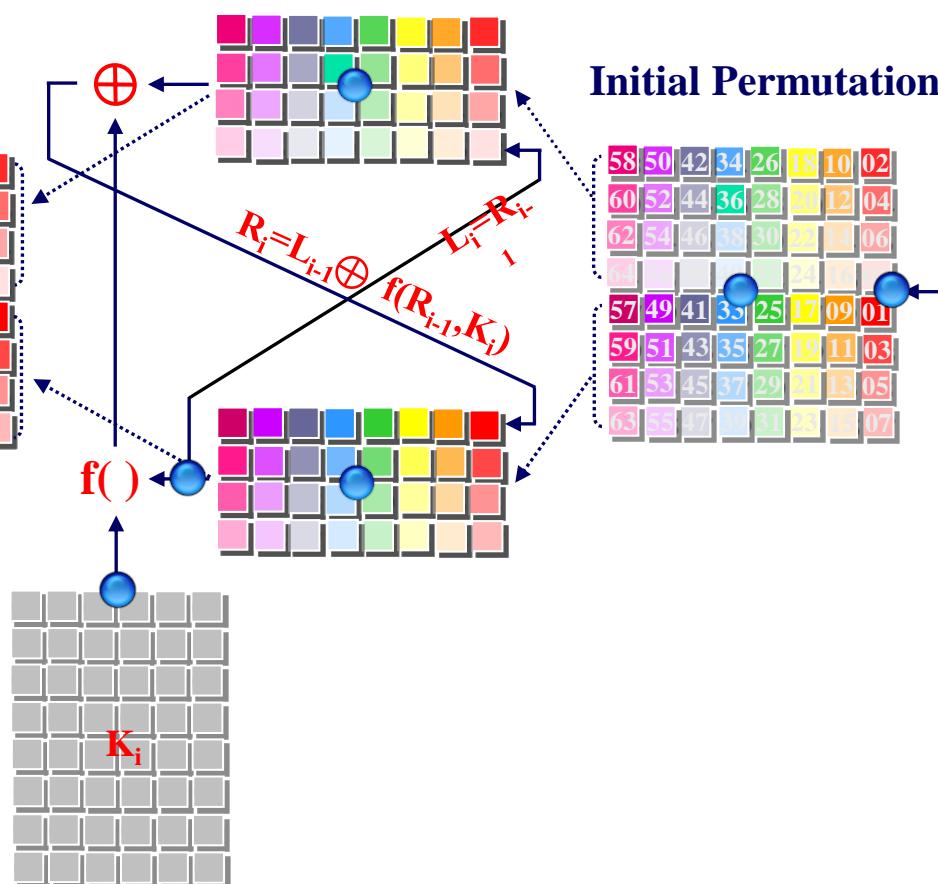
第*i*回合Feistel加密

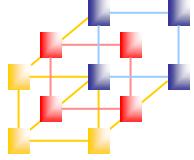
Initial Permutation

58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
57	49	41	35	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

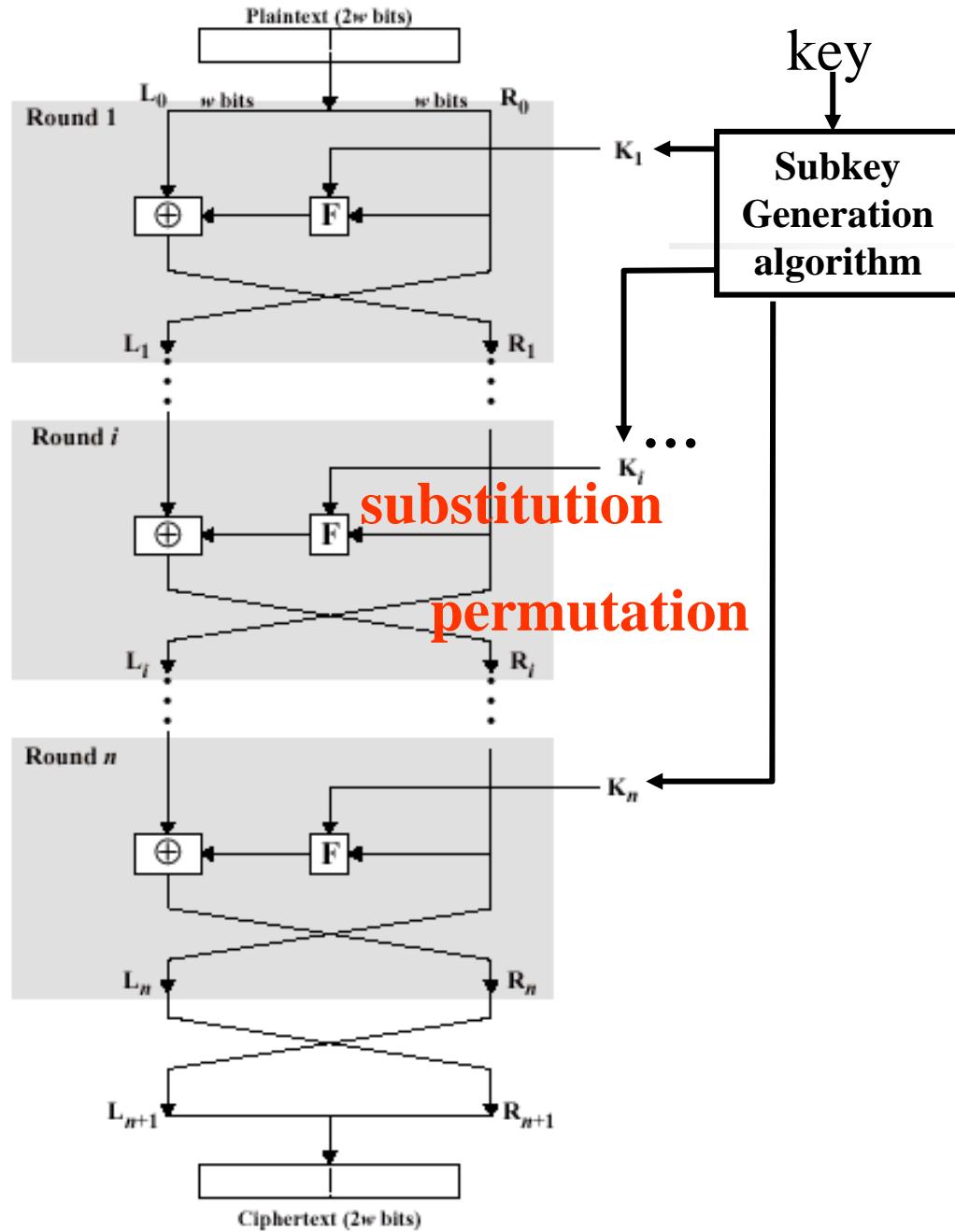
Input

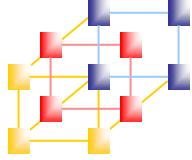
01	02	03	04	05	06	07	18
09	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64





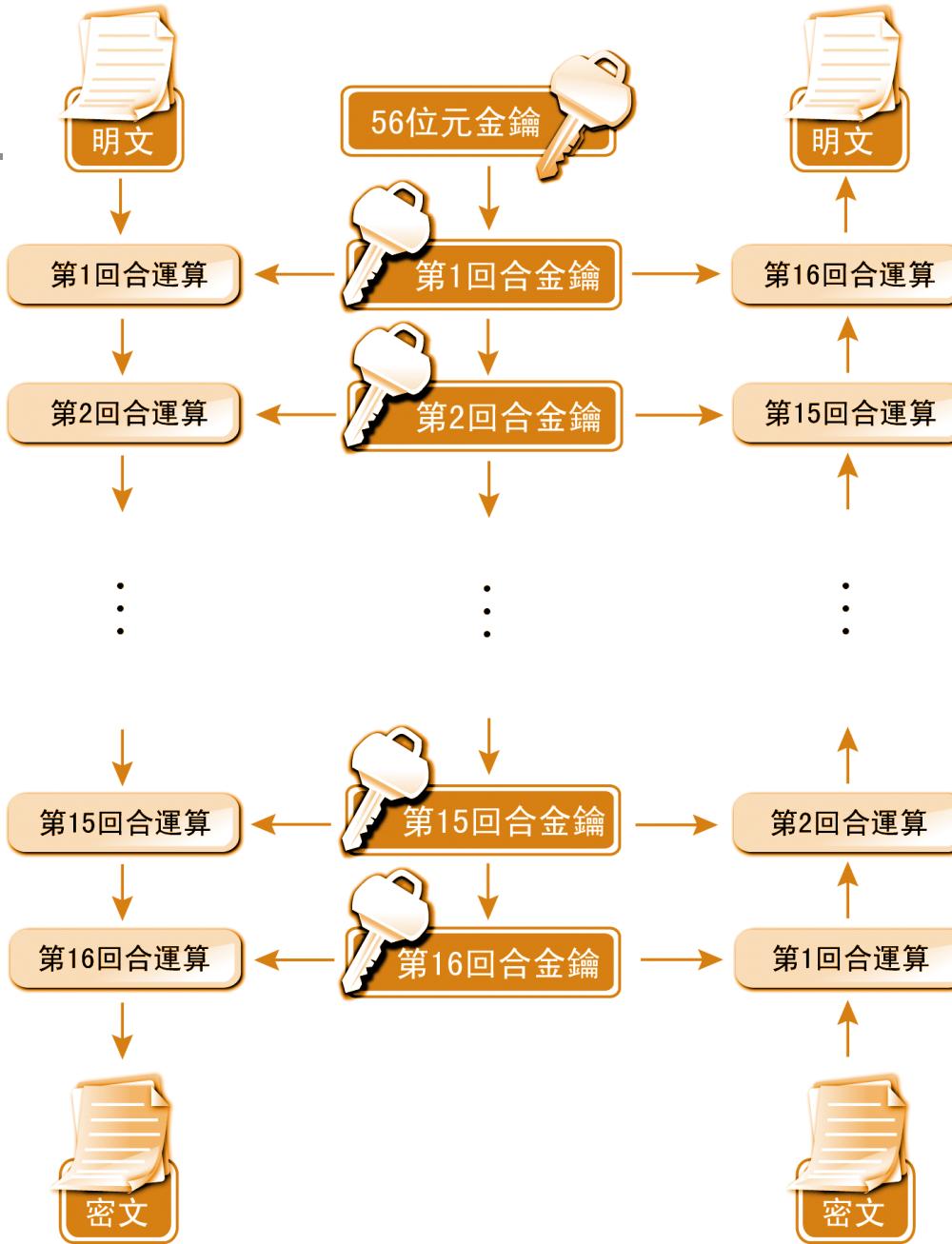
另一張圖

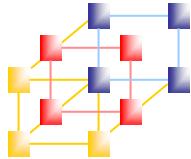




DES 加解密 架構

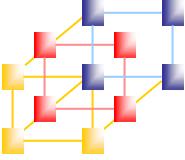
加密 產生回合金鑰 解密





DES的變革

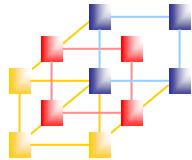
- 1997年，RSA安全贊助了一系列的競賽，獎勵第一個成功破解以DES加密的訊息的隊伍1萬美元，Rocke Verser、Matt Curtin、Justin Dolske領導的DESCHALL計劃獲勝，該計劃使用了數千台連接到網路的電腦的閒置計算能力。
- 1998年，電子前哨基金會（EFF）製造了一台DES破解器，造價約\$250,000。該破解器可以用稍多於2天的時間暴力破解一個密鑰，它顯示了迅速破解DES的可能性。
- 下一個成功的DES破解器是2006年由德國的魯爾大學與基爾大學的工作組建造的COPACOBANA。有趣的是一台COPACOBANA的造價大約是\$10,000，是EFF設備的25分之一。
- 2007年，COPACOBANA的兩個計畫參與者組建的SciEngines公司改進了COPACOBANA，並發展了它的下一代。
- 2008年，他們的COPACOBANA RIVYERA將破解DES的時間減少到了1天以內，使用128片Spartan-3 5000型FPGA。



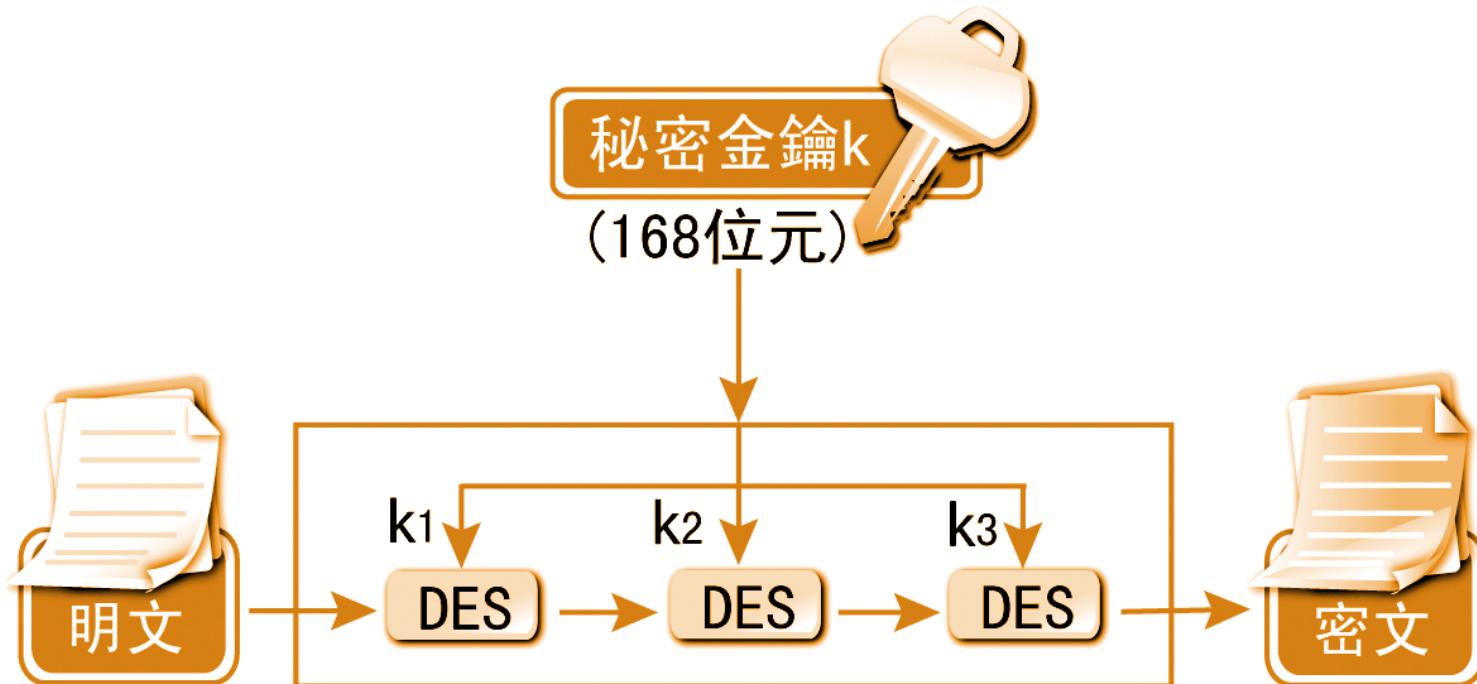
DES的變革



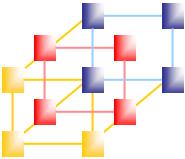
- 電腦軟硬體大幅改進
 - 電腦運算處理速度的提升
 - 平行運算技術(格網運算、雲端運算、...)
- 56bits的金鑰長度太短了
 - 鑰匙的窮舉搜尋法(暴力破解)已被證明可以破解 DES
- 內部架構設計列為機密，使用者無法確定DES內是否還有一些隱藏的弱點，讓攻擊者不需要鑰匙就能解開密文。
- 改進DES
 - Triple DES



Triple DES

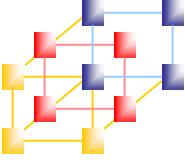


Triple DES 加密架構



Triple DES使用方式

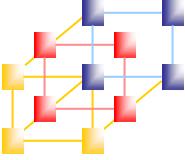
- **EEE3**：用三把不同秘密金鑰（即金鑰長度為168位元）並以加密-加密-加密依序處理產生密文
- **EDE3**：用三把不同秘密金鑰，並以加密-解密-加密依序處理產生密文
- **EEE2**：用二把不同秘密金鑰（即金鑰長度為112位元）任選二個DES金鑰設為相同（例如，第一個及第二個DES金鑰相同，但與第三個DES金鑰不同），並以加密-加密-加密依序處理產生密文
- **EDE2**：用二把不同秘密金鑰（即金鑰長度為112位元）第一個DES金鑰與第三個DES金鑰相同，但與第二個DES金鑰不同，並以加密-解密-加密依序處理產生密文



DES重點整理

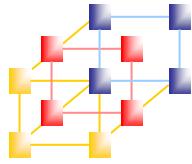


- IBM的產品
- 曾經的美國國家標準
- 一次加密64Bits，金鑰長56Bits (另8Bits為校驗碼)
- 可以連續做三次可增強安全性
- EEE3、EDE3、EEE2、EDE2算法，若用EDE1相容DES
- S-Box(替換表)的外星科技 (<https://ppt.cc/fSm8sx>)



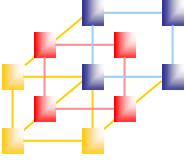
AES (Advanced Encryption Standard)

- 為了取代DES，NIST在1997年公布AES (Advanced Encryption Standard，高等加密標準) 徵選活動。
- 在2000十月，NIST宣佈來自比利時的兩位密碼學家 — Joan Daemon和Vincent Rijmen，他們提出的Rijndael演算法贏得這項競賽。
- 並於2001年十一月完成評估，發佈為FIPS PUB 197標準。
- AES牢靠度高、適用於高速網路、可在硬體設備建置等因素，都是這個演算法獲選的原因。



AES 的特色

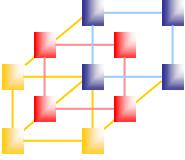
- 採用私密金鑰的對稱式區段加密法。
- Rijndael 演算法支援多種分組及金鑰長度，介於 128-256 之間所有 32 的倍數均可
 - 最小支援128位元，最大256位元，共25種組合
- AES 標準支援的分組大小固定為128位元，金鑰長度有3種選擇：**128位元、192位元及256位元**
- 運算速度比 Triple-DES 更強更快。
- 具有完備的規格與設計細節供參考。
- AES可採用 C 與 Java 來實作。



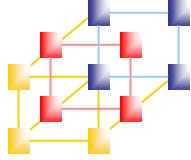
AES



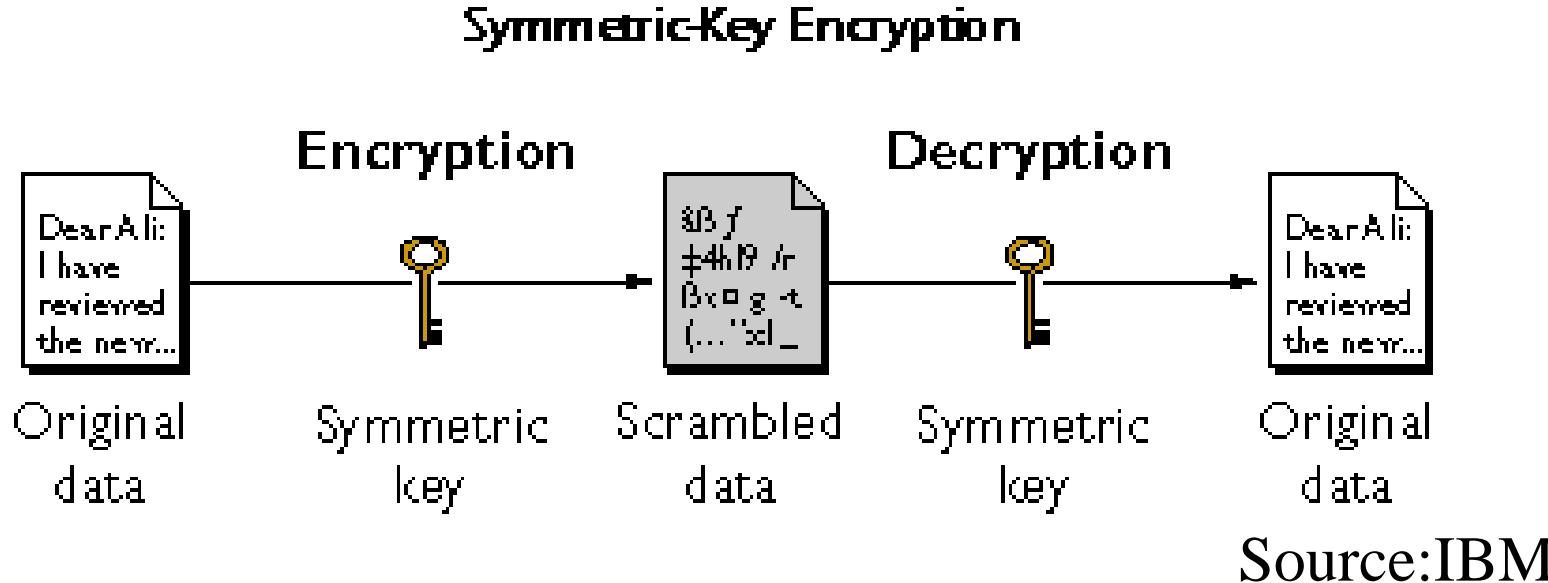
- 1997年1月2日由NIST經由公開程序對外徵求
- 1997年4月15日舉辦AES研討會，研討制訂AES之功能需求
- NIST於1997年9月12日，正式公佈AES功能規格標準需求：
 - 為一秘密加密演算法
 - 為一區塊加密演算法
 - 進行加密之區塊最小為128位元
 - 秘密金鑰之長度是變動的，可以為128、192 或 256位元
 - 可以同時由硬體及軟體來實作
 - 沒有專利的限制，可以自由使用



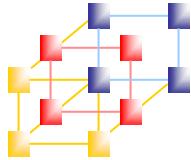
- 1998年8月20日NIST舉行第一屆AES會議，宣佈15個獲AES初選之演算法：CAST-256、CRYPTON、DEAL、DFC、E2、FROG、HPC、LOKI97、MAGENTA、MARS、RC6、RIJNDAEL、SAFER+、SERPENT、TWOFLISH。
- 1999年3月22日第二屆AES會議中針對此15個初選AES之安全性、效率及相容性提出分析報告。
- 1999年8月20日公佈MARS、RC6、Rijndael、Serpent及Twofish進入第二回合決選。
- 2000年4月13日舉行第三屆AES會議，對五個決選AES演算法再進行分析。
- 2000年10月2日宣佈Rijndael（發音為Rain Doll）獲選為AES之演算法。
→ 是目前最主流的對稱式(一把Key)加密系統



對稱加解密系統的使用方法

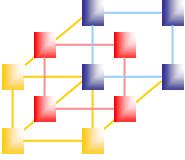


→But 內容超過區塊長度怎麼辦？！
→But 金鑰如何傳遞？！



區塊加密的特性

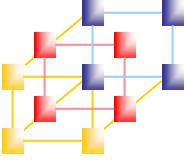
- 把整個鑰加密的文件切切切
 - 每56Bits跑一次
- 要怎麼切？金鑰怎麼用？



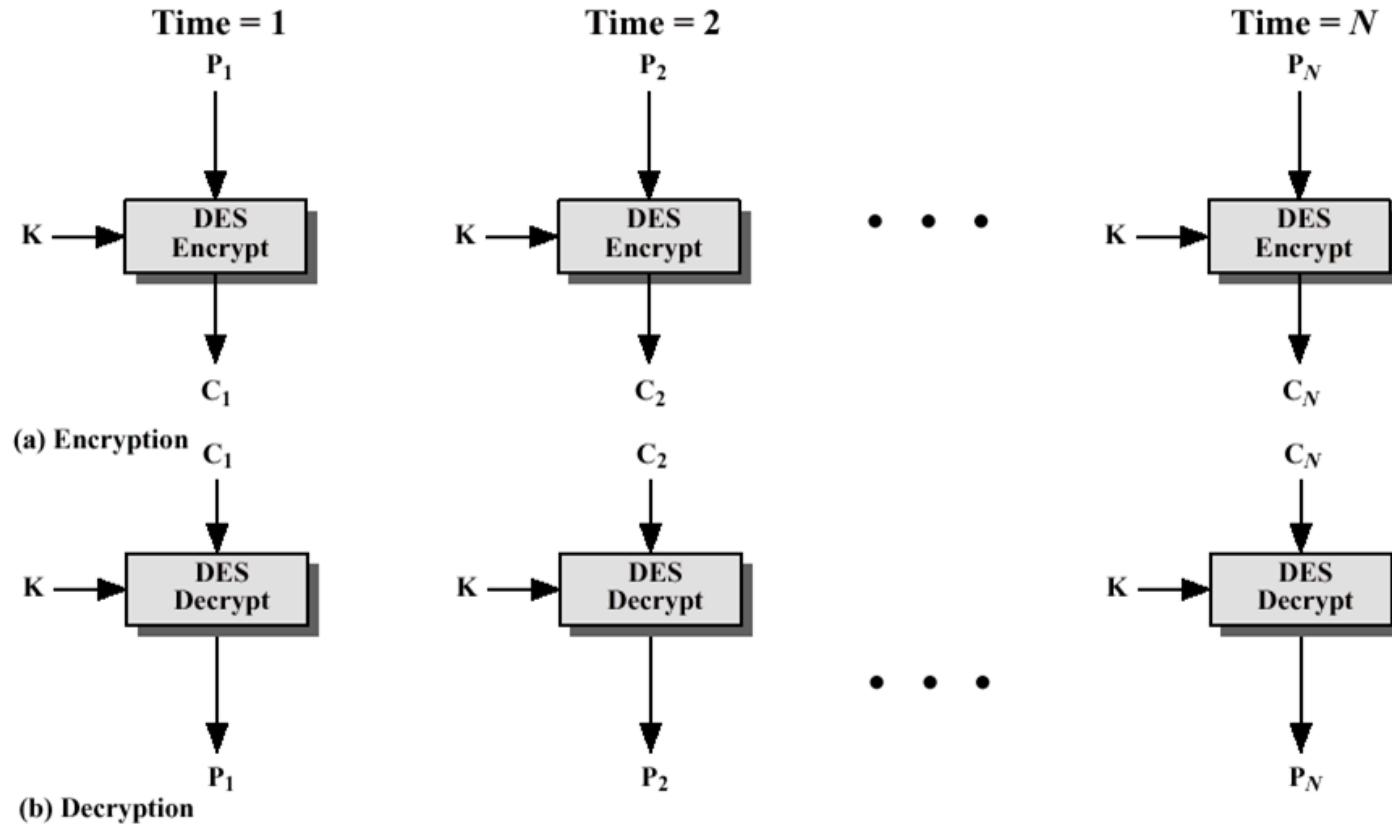
各種加密模式



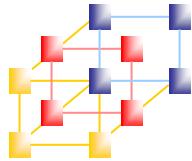
- ECB Mode (Electronic Code Book Mode)
- CBC Mode (Cipher Block Chaining Mode)
- CFB Mode (Cipher Feedback Mode)
- OFB Mode (Output Feedback Mode)
- CTR Mode (Counter Mode)



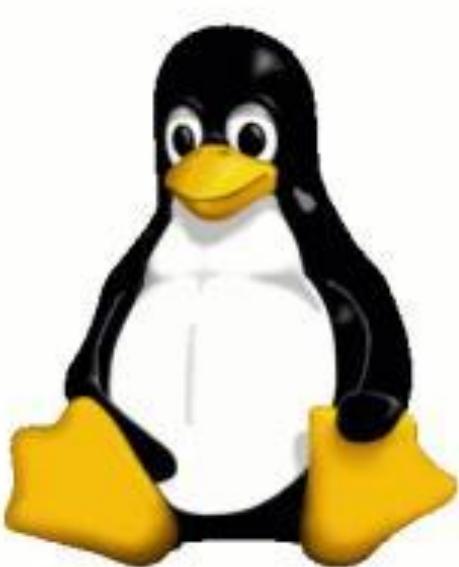
ECB Mode



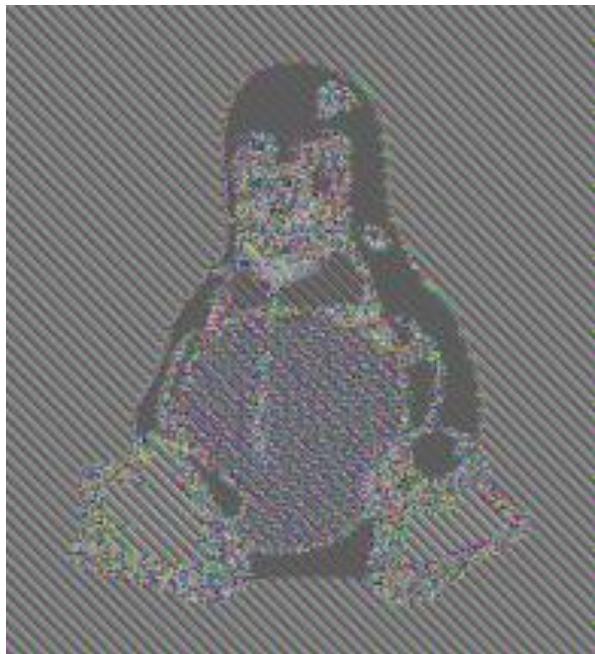
特性：可以平行處理，加快加解密速度，任一區塊的錯誤不會影響其他區域。但是相同明文會產生相同密文，不適合具有結構性的訊息或是較長的訊息。



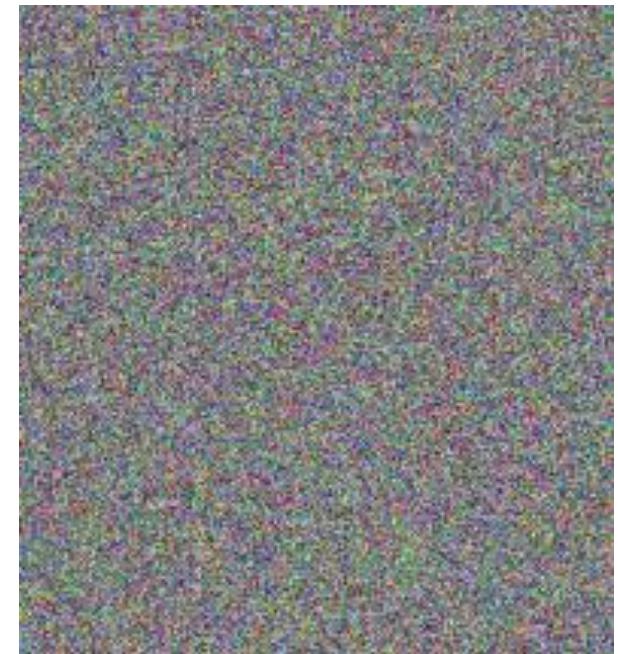
ECB Mode



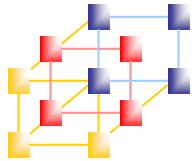
原圖



ECB Mode

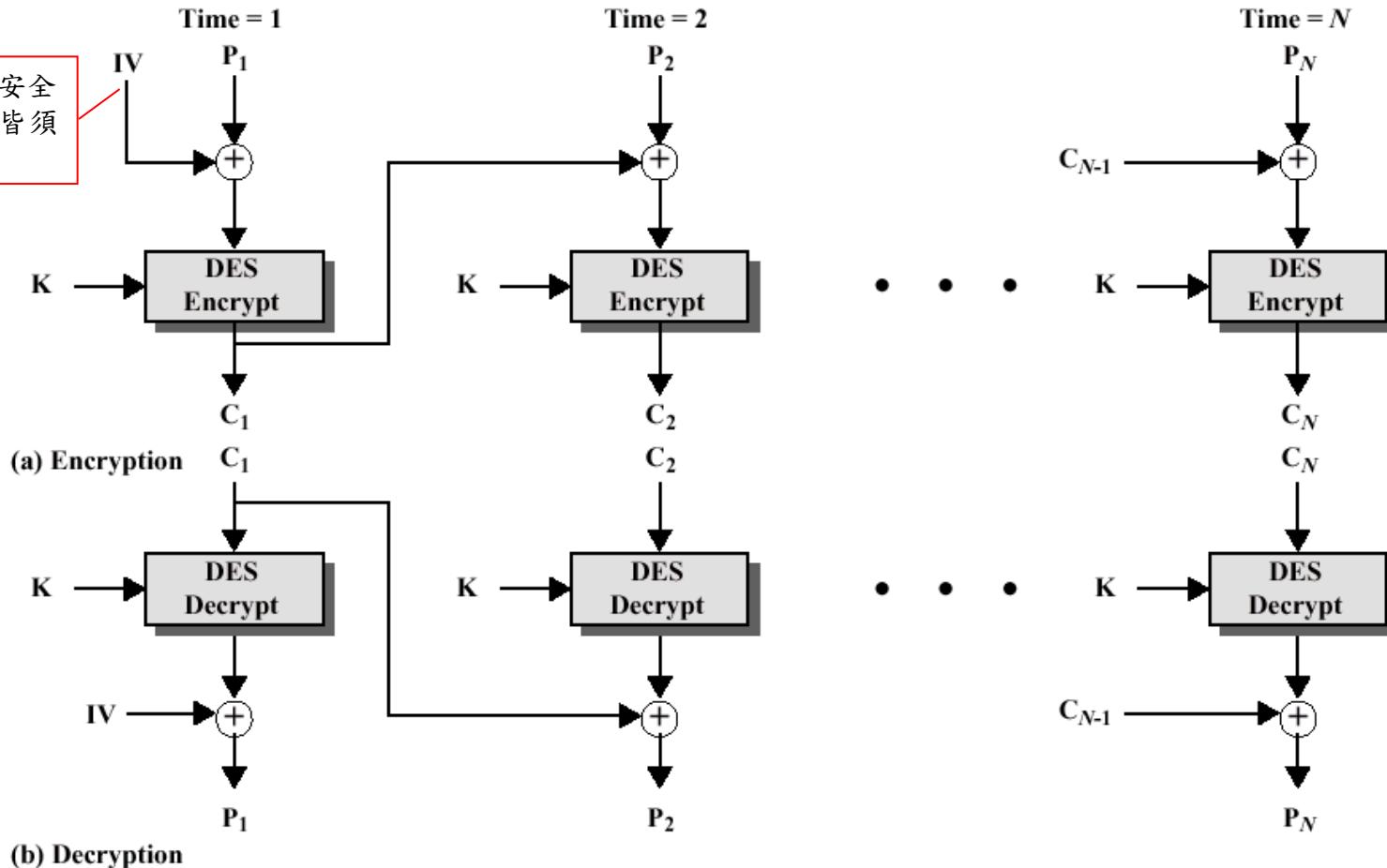


Other Mode



CBC Mode

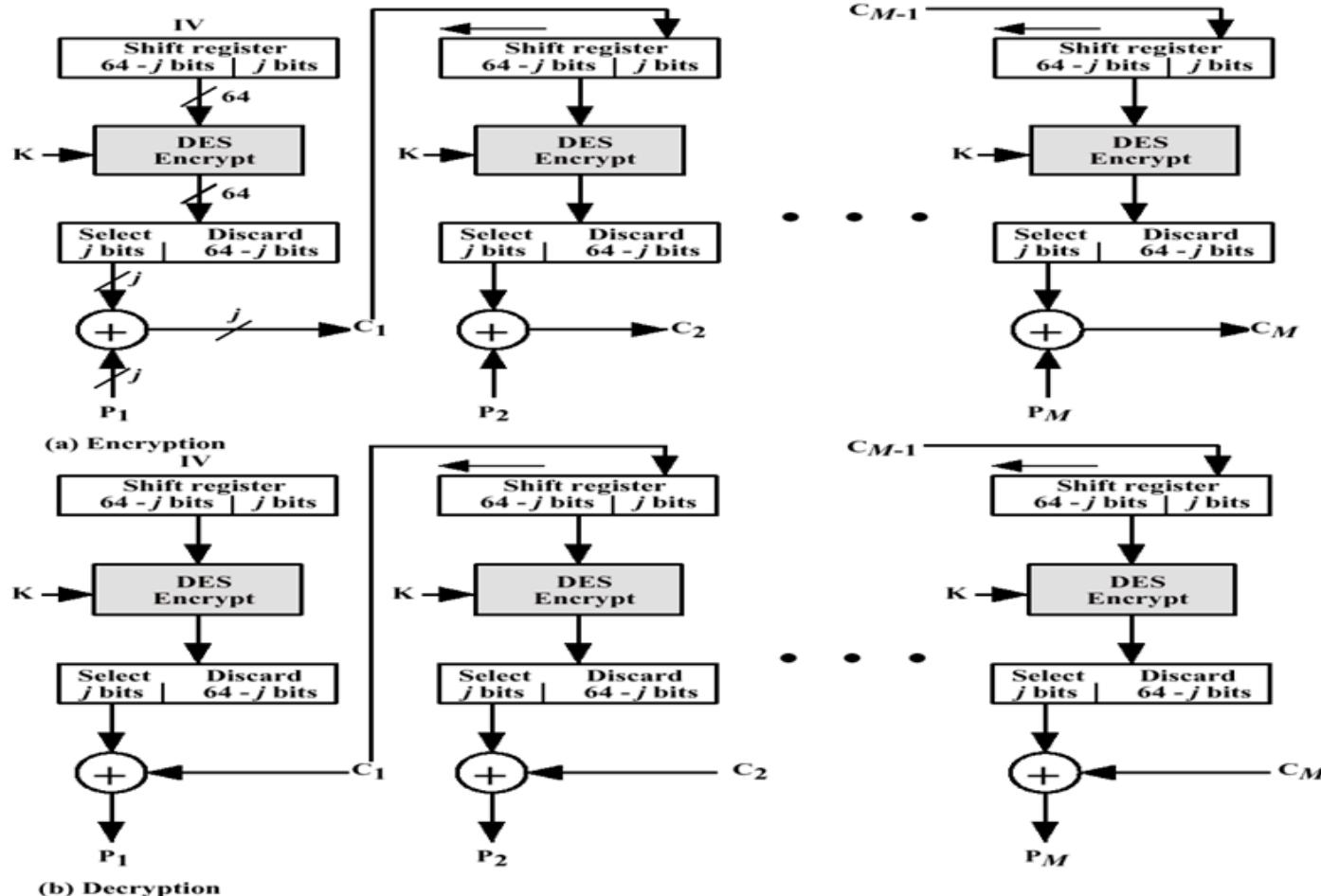
若要達到較高安全性，IV 與 Key 皆須被保護。



特性：加入初始向量，不同初始向量可讓相同明文產生相異密文。
解密時密文 C 錯誤，不會錯誤擴散。明文必須填充至區塊的整數倍。



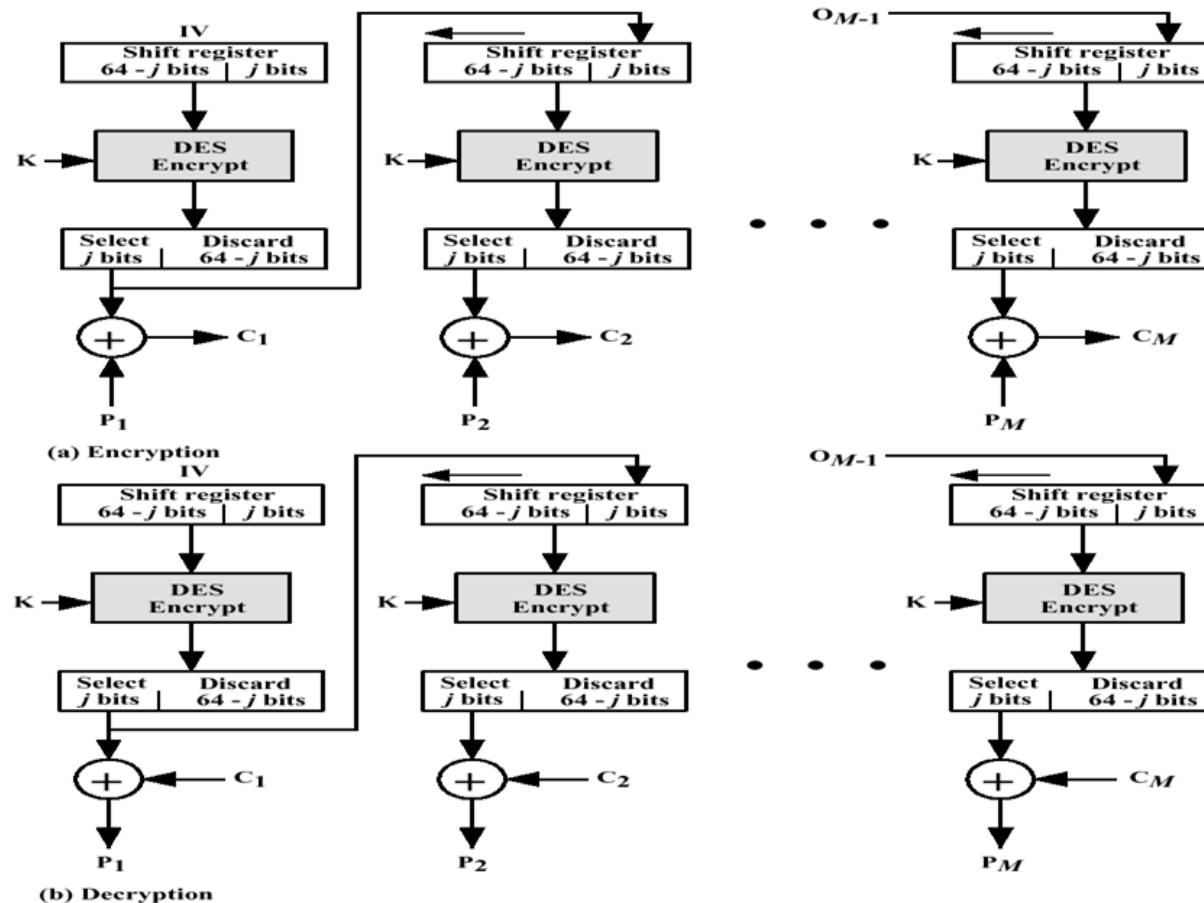
CFB Mode



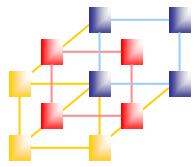
特性：加密區塊長度可變動，解密時需收到密文 C_i 才能進行區塊 P_{i+1} 的DES運算，若區塊 C_i 遺失，僅會造成 P_i 與 P_{i+1} 區塊無法解密，後續封包可自我同步。



OFB Mode



特色：可變動加密區塊長度，解密時不會錯誤擴散。解密時可以預先輸入 IV完成所有區塊的DES運算，收到密文後僅需進行 \oplus 運算即可(稱為 *pre-computed*)。若區塊 C_i 遺失，會造成後續區塊均無法解密。



CTR Mode

Counter Counter + 1



Counter + N - 1



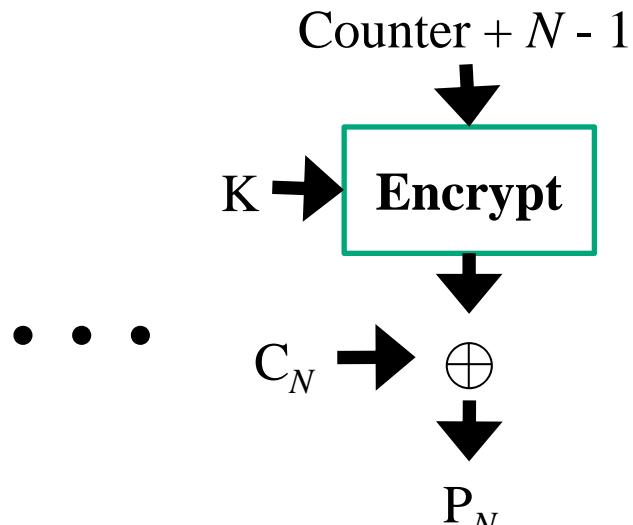
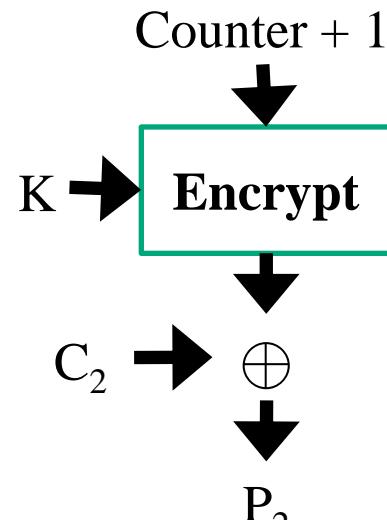
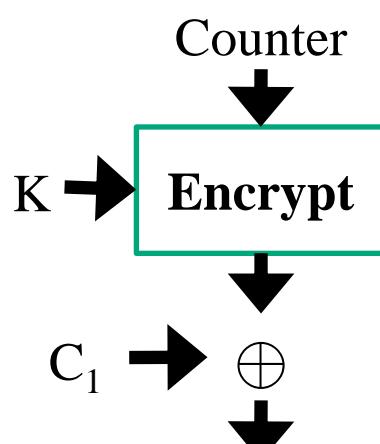
$$P_1 \rightarrow \oplus \downarrow C_1$$

$$P_2 \rightarrow \oplus \downarrow C_2$$

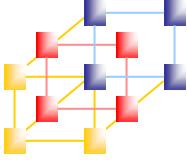
• • •

$$P_N \rightarrow \oplus \downarrow C_N$$

(a) CTR Mode Encryption

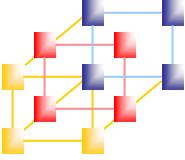


(b) CTR Mode Decryption *Information and Network Security*



CTR Mode的優點

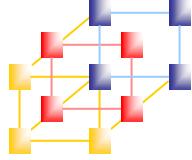
- 硬體效能、軟體效能、前置處理、簡單
- 硬體效能
 - CTR Mode可以同時對多個明文或密文區段加密(或解密)。
 - 以往的串接模式演算法必須完成一個區段的計算，才能在開始下一個區段，使得演算法的加解密運算無法達到最大的生產力。
- 軟體效能
 - 因為CTR Mode的同步處理特性，擁有同步處理功能的處理器便能發揮最大的效能。



CTR Mode的優點



- 前置處理
 - 底層的加密演算法並不需要明文或密文輸入。因此，假如有足夠的記憶體及完善的安全措施，我們就可以透過前置處理來預先算出加密方塊的輸出；這樣一來，當明文或密文輸入就緒時，剩餘的計算就是一系列的XOR。這樣可以大幅提升生產力。
- 簡單
 - 不像ECB和CBC模式，CTR模式只需要時做加密演算法，而不需要解密演算法。此外，我們也無須擔心解密程序的key產生問題。



OTP Token

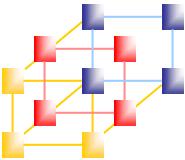


- Time Based Token

<https://www.youtube.com/watch?v=XDEGfaOJ10M>

- Challenge Response Based Token

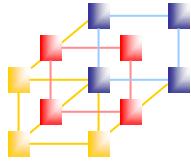
<https://www.youtube.com/watch?v=Sh2Iha88agE>



Quiz 4

- Q：
 - 請問您認為什麼這些Token都沒有很普遍，不是人人都拿？
 - 請問這些OTP Token有什麼缺點？
(請不要寫會忘記帶這項)

- 請將答案寫在紙條上，還有下一題，不要急。
- ((請記得寫上班級姓名學號



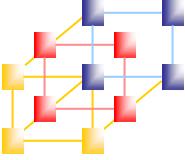
密碼系統的分類

- 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



公開金鑰密碼系統

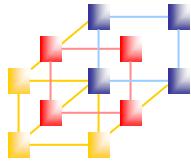


公開金鑰密碼系統(Public-Key Cryptosystems)
又稱雙金鑰密碼系統(Two-Key Cryptosystems)
也稱非對稱密碼系統(Asymmetric Cryptosystems)

優點：沒有金鑰管理的問題
高安全性
有數位簽章功能

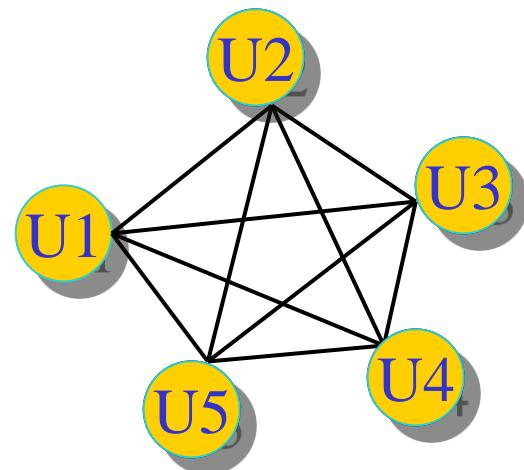
缺點：加解密速度慢

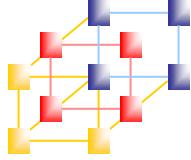
著名之公開密碼系統：
 RSA 密碼系統
 ElGamal 密碼系統



公開金鑰基本概念

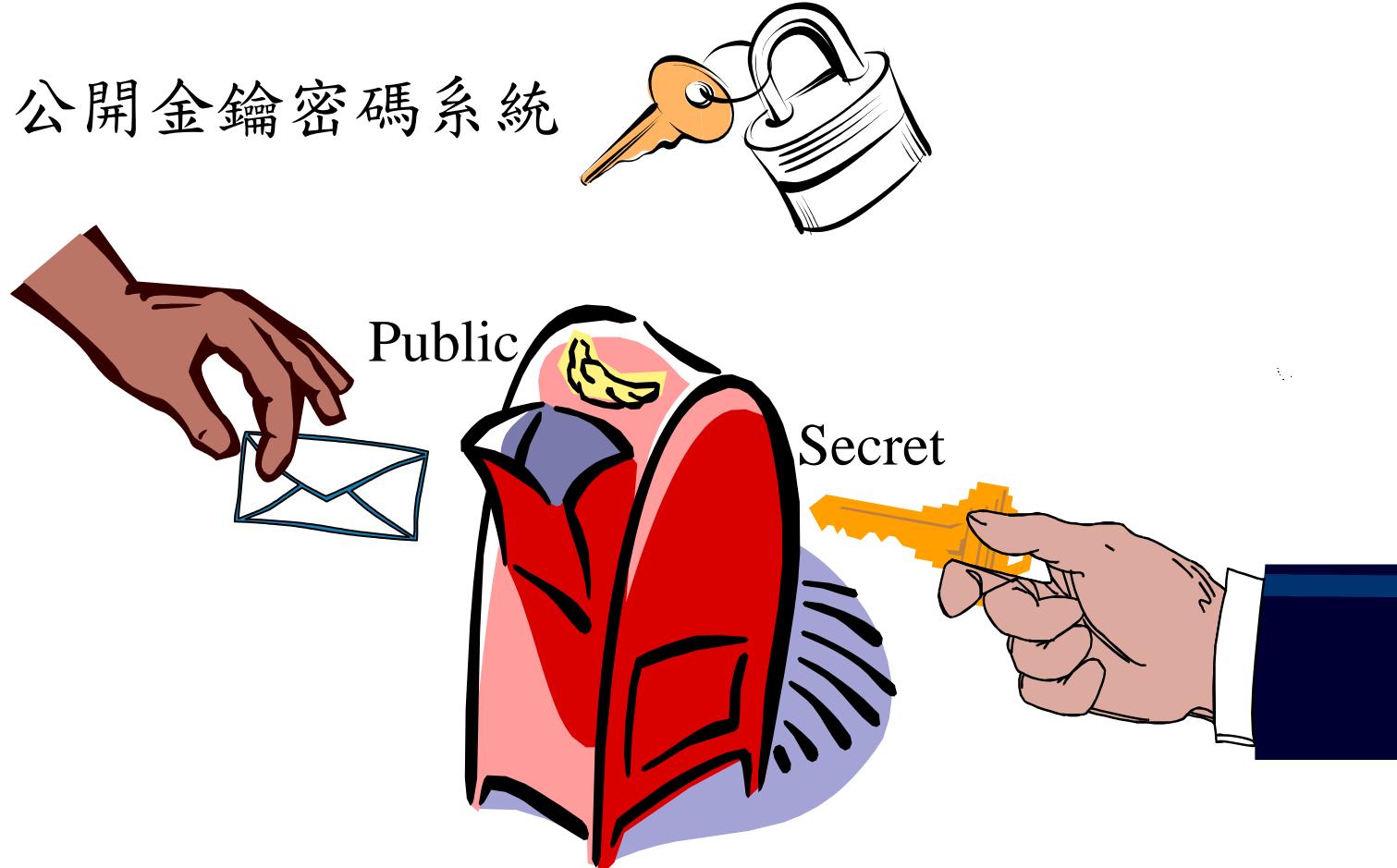
- 對稱式密碼系統有金鑰的管理問題
 - 例如要與N個人做秘密通訊，那麼就必須握有N把秘密金鑰
- 為了改善對稱式密碼系統問題，於是便有公開金鑰密碼系統(Public-Key Cryptosystems)的產生

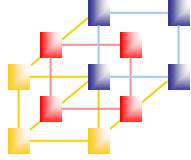




你家門口有信箱嗎？

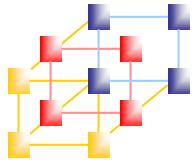
- 公開金鑰密碼系統



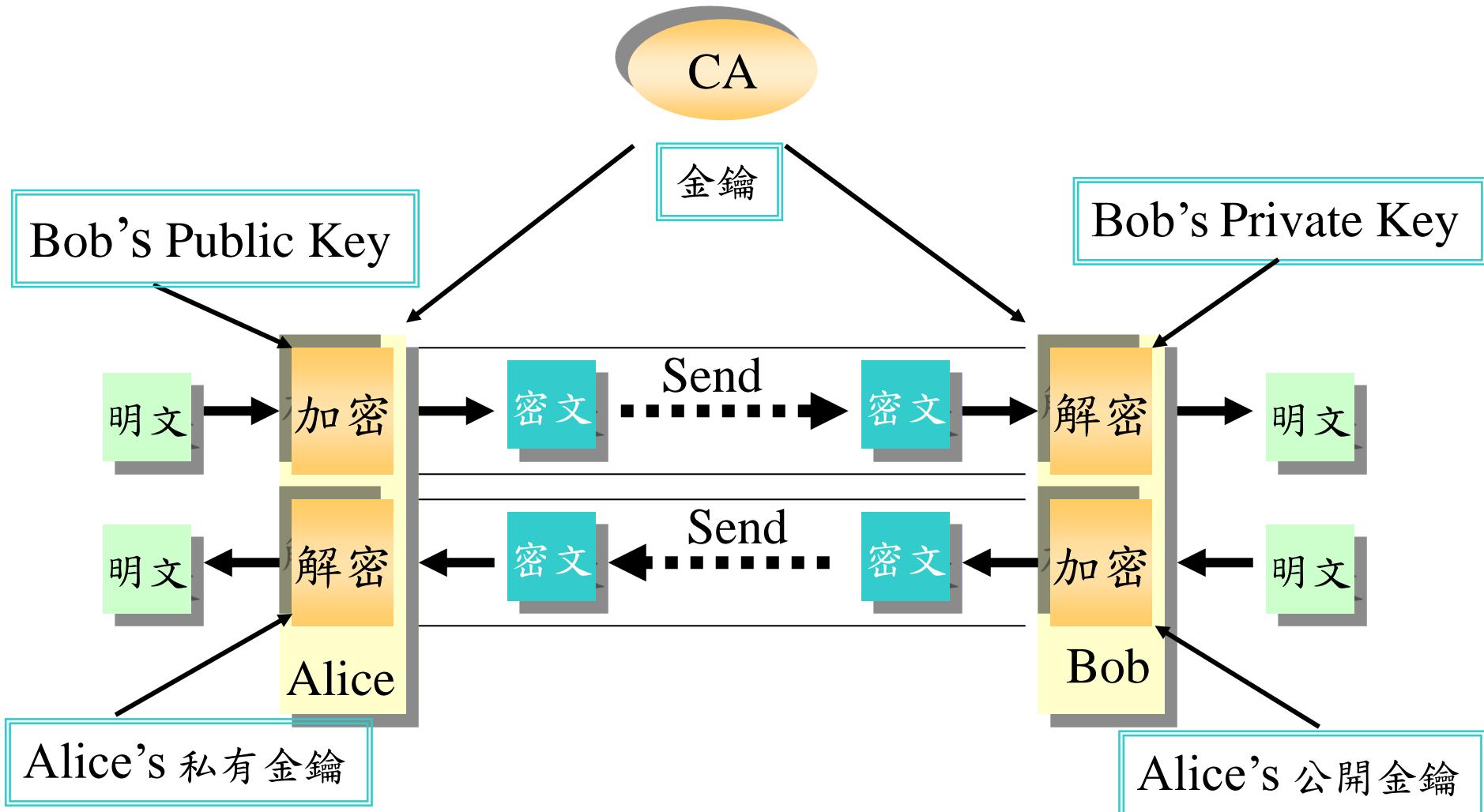


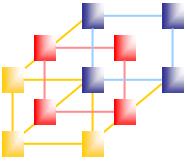
公開金鑰密碼系統

- 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



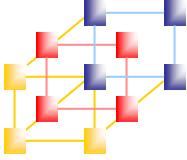
公開金鑰加密系統





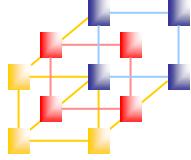
RSA 加密法

- 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。



有關非對稱式加解密方法

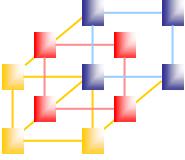
- 通常都基於數學難題
- RSA
 - 基於大數因數分解的難題 (兩個大質數相乘後難以回推)
 - 透過不斷的加長金鑰來強化安全性
(512→1024→2048)
 - RSA Factoring Challenge (<https://ppt.cc/fPsNQx>)
- 其他常見的公開金鑰系統
 - ElGamal (離散對數難題)
 - ECC (橢圓曲線加密) – 更新、安全性更高



RSA 演算法

- 張三選 2 個大質數 p 和 q (至少 100 位數)，令 $N = p \cdot q$
- 再計算 $\varphi(N) = (p-1)(q-1)$ ，並選一個與 $\varphi(N)$ 互質數 e
 - $\varphi(N)$ 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 $e \cdot d \bmod \varphi(N) = 1$
- 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 $M = C^d \bmod N$

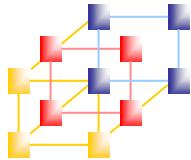
- RSA 之安全性取決於 **質因數分解之困難度**
- 要將很大的 N 因數分解成 P 跟 Q 之相乘，是很困難的



RSA 演算法- 例子

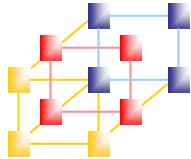
- 張三選 $p = 3$, $q = 11$
此時 $N = p \cdot q = 3 \times 11 = 33$
- 張三選出一個與 $(p-1) \times (q-1) = (3-1)(11-1) = 20$ 互質數 $e = 3$
- $(e, N) = (3, 33)$ 即為張三的公開金鑰
- 張三選一個數 $d = 7$ 當作解密金鑰，
滿足 $e \cdot d \equiv 1 \pmod{20}$ ($7 \times 3 \equiv 1 \pmod{20}$)

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



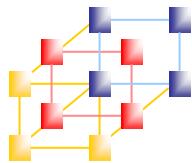
公開金鑰加密系統之特性 (1/2)

1. $D(d, E(e, M)) = M$ ，可還原性
2. d 和 e 很容易求得
3. 若公開(e, n)，別人很難從(e, n)求得 d ，即只有自己知道如何解密(以 e 加密)
4. $E(e, D(d, M)) = M$



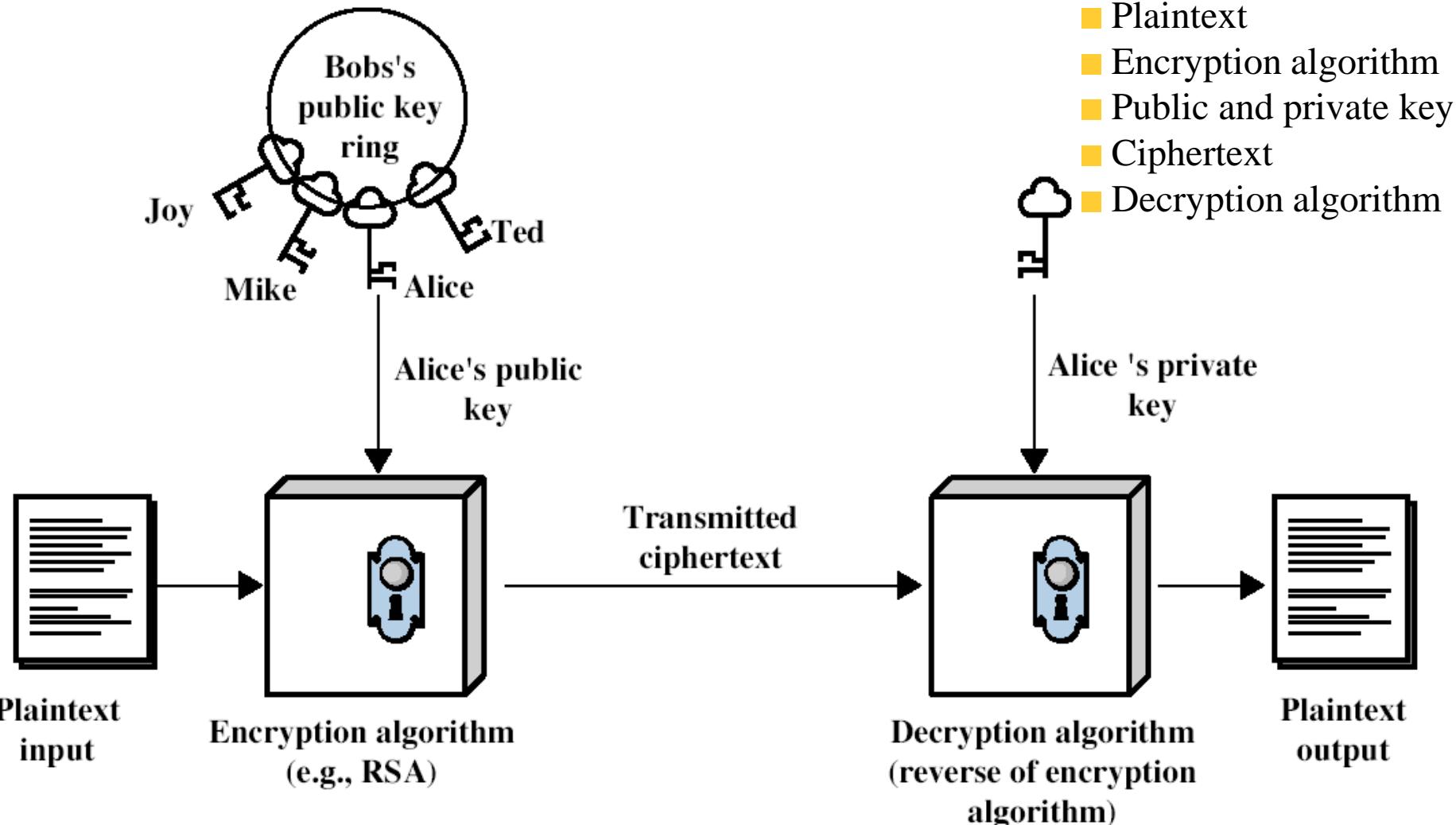
公開金鑰加密系統之特性 (2/2)

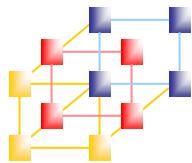
- 滿足1~3項稱之為trap-door one-way function
 - “one-way”因易加密而不易解密
 - “trap-door”若知一些特別資訊即可解密
- 滿足1~4項稱之為trap-door one-way permutation
- 1~3項為public-key cryptosystems之要求
- 若同時滿足第4項要求，則該保密法可用來製作數位簽章。



Public-Key Cryptography

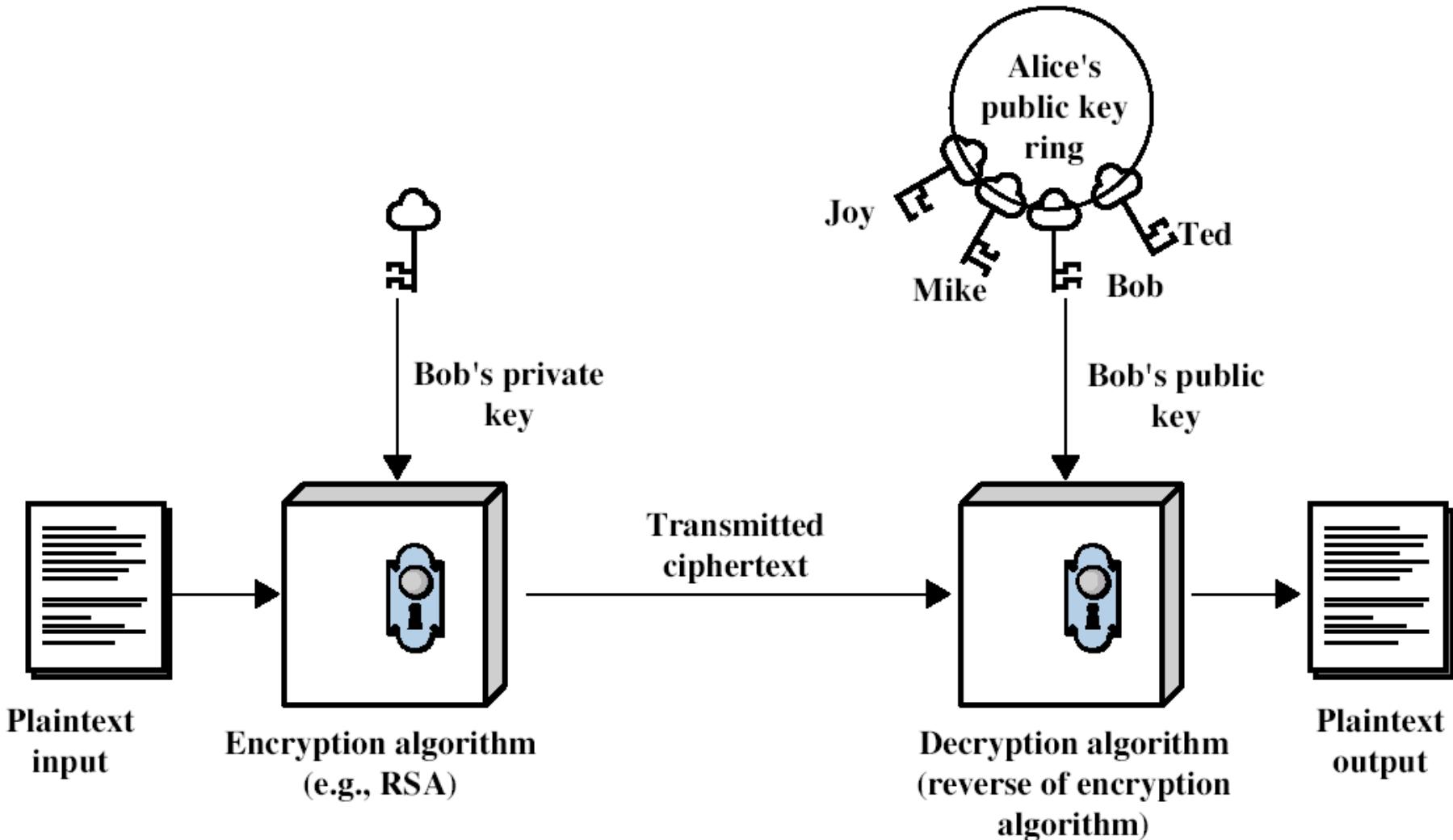
-- Encryption

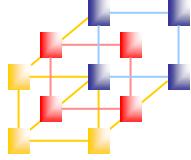




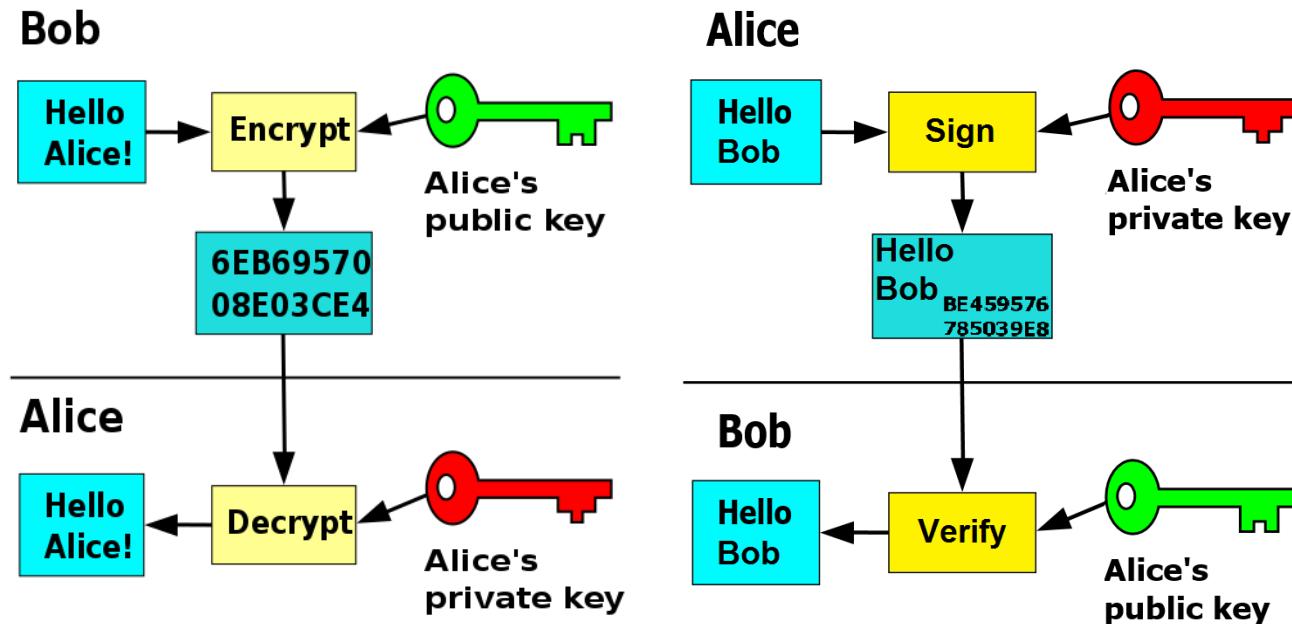
Public-Key Cryptography

-- Authentication

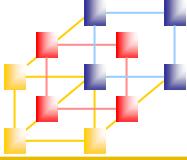




公開金鑰加解密系統使用方法

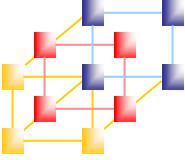


→But 這公鑰真的是你的嗎？



對稱式加密 v.s. 非對稱式加密

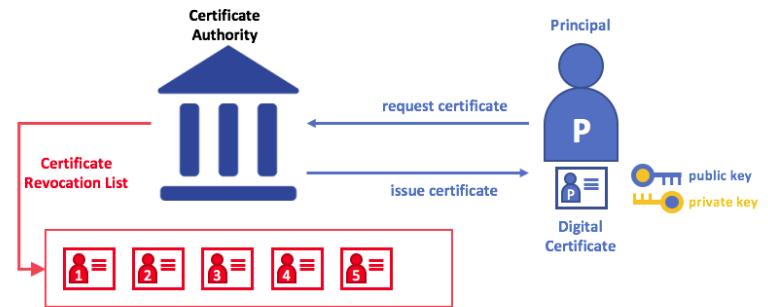
	對稱式加密	非對稱式加密
常見代表	DES、3DES、AES、Blowfish、Twofish	RSA、ElGamal、ECC
特色	加解密使用同一把金鑰	有兩把金鑰，分別進行加解密
開發原理	替換和位移	各種數學難題
優勢	快，非常快	沒有金鑰交換問題，一組Key打天下
金鑰長度	基於方法設計而有所不同	越長越安全，但越慢
缺點	須設法轉交金鑰給對方，Key滿天飛	須設法證明公鑰代表身分之真確性
常見用途	加密較大資料	加密少數資料、簽章
提供功能	機密性	機密性、完整性、不可否認性
好朋友	非對稱式加密	非對稱式加密(加密)、Hash(簽章)



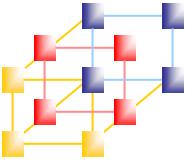
公鑰也只是一串亂碼

■ 怎麼辦？

- 公開金鑰的信任問題
- 每個使用者都要有Key Pairs才能通訊，造成不便
- 數位憑證 (證明金鑰的身分) → 誰來發、誰來簽
- PKI (Public key infrastructure, 公開金鑰基礎建設)
 - 自然人憑證、健保卡...
 - HTTPS、IMAPS、Balala“S”... (可以驗憑證的)

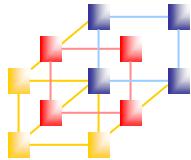


Source: <https://hyperledger-fabric.readthedocs.io>
Information and Network Security



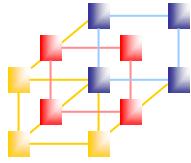
雜湊函數

- 在網路上公開的傳送文件訊息(Document or Message)很容易遭到駭客攔截竄改、新增、或刪除等攻擊。
 - 需對文件訊息作**完整性(Integrity)**驗證。
- 該文件訊息是否確實為某人所送過來的文件訊息，而非由他人假冒。
 - 需驗證訊息的**來源是否正確**。
- 這兩項功能可藉由訊息鑑別碼(Message Authentication Code，MAC) 的輔助來達成



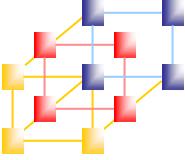
單向雜湊函數

- 單向雜湊函數二個主要功能
 - 將文件訊息打散及重組，使其不能再還原為原始文件訊息。
 - 將任意長度的文件訊息壓縮成固定長度的訊息摘要(Message Digest, MD)。
- 數學式子
 - $MD = H(M)$ $H(\cdot)$: 一單向雜湊函數
 M : 表一任意長度文件訊息
 - Ex: $E(M) = M^2 \bmod 1024$ ，不管文件 M 多大，經由 $E(\cdot)$ 計算的結果都是一個 10 bits 的數



單向雜湊函數特性

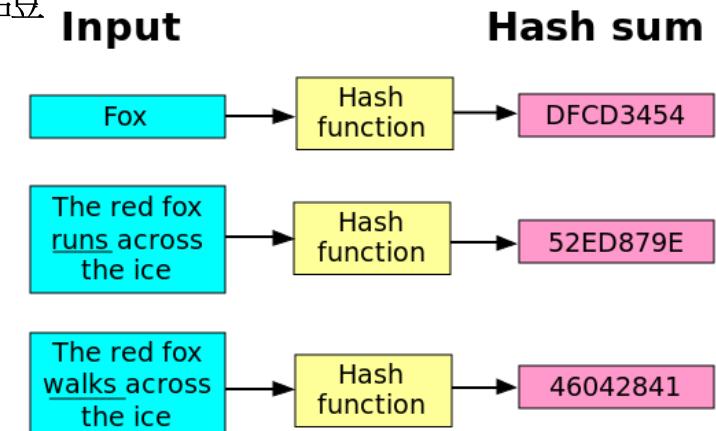
- 單向雜湊函數三種特性
 - 紿定文件訊息M，可很容易算出其對應的訊息摘要MD。
 - 紿定一訊息摘要MD，很難從MD去找到一個文件訊息M'，使 $H(M') = MD$ 。
 - 紉定一文件訊息M，很難再找到另一文件訊息M'，使 $H(M) = H(M')$ 。
 - 避免碰撞的情況發生 (Collision-resistance)



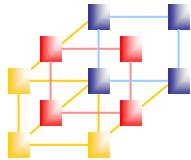
有關Hash Function

■ 單向雜湊函數

- 概念：指紋碼 
- 特色：無論輸入什麼東西、多大的東西，都可以跑出相同長度但很難重複的一串代碼(也可以稱為特徵碼)
- 改任意字元就會整串全部改變
- 常見的Hash Function：MD5、SHA1、SHA256...
- 用法：簽章標的、ISO檔完整驗證
- 安全問題：碰撞、資料庫
- Google: MD5 Crack online

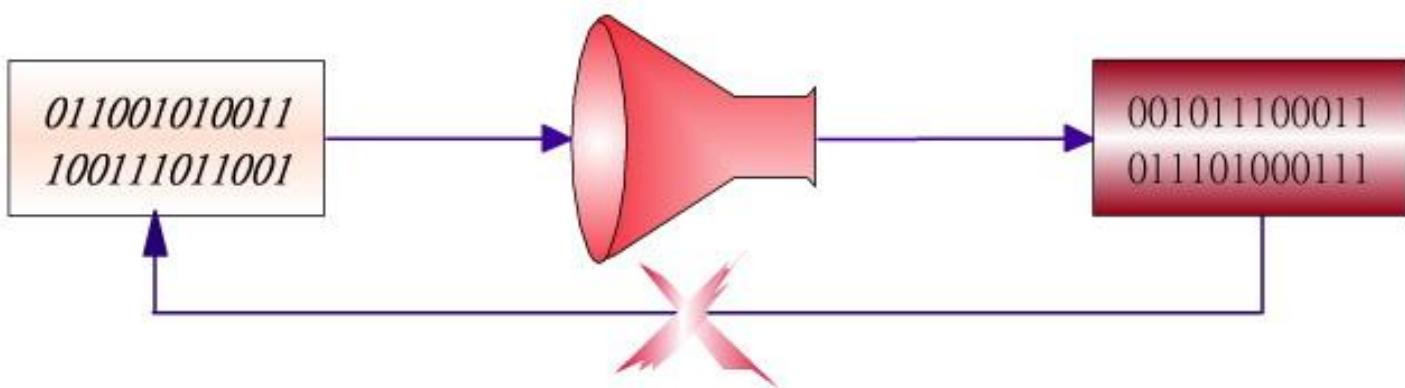


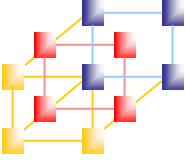
Source: Wikipedia



MD5

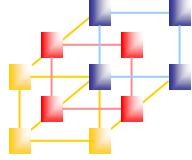
- MD5 (RFC1321)是由 MIT 工學院教授 Ron Rivest 所發展的單向雜湊函數演算法
 - 還有 MD2(RFC1319)、MD4(1320)
- MD5 可將任意長度的文件訊息壓縮成 128-bit (16 bytes) 訊息。





Secure Hash Algorithm (SHA-1)

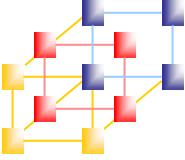
- SHA 由美國國家標準局 (National Institute of Standard and Technology , NIST) 所提出
- SHA-1 可將任意長度的文件訊息壓縮成 160-bit (20 bytes) 訊息
- 2017 年，Google 已經確認被廣泛使用的 SHA-1 雜湊演算法可發生碰撞 (collision) ，同時也提醒所有使用此演算法的系統盡早改用更安全的 SHA-3 或 SHA-256 演算法，以避免未來可能發生的資安問題



下載ISO時為何要看HASH

- Ex.

http://ftp.twaren.net/Linux/CentOS/7/isos/x86_64/



不一樣的Hash Function

- 所有Hash都是改任意字元就會整串全部改變？
 - Fuzzy Hashing
 - Ssdeep
 - 差異比對
 - 案例

Demo: Fuzzy Hash Comparator

Input

Hash 1: 48:ybr3MfxNiYfUFx2h3Jzx4

Hash 2: 48:yr7eiwpWzQndlqzn7BX€

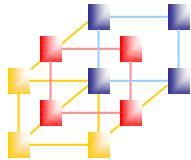
Reset

Run

Output

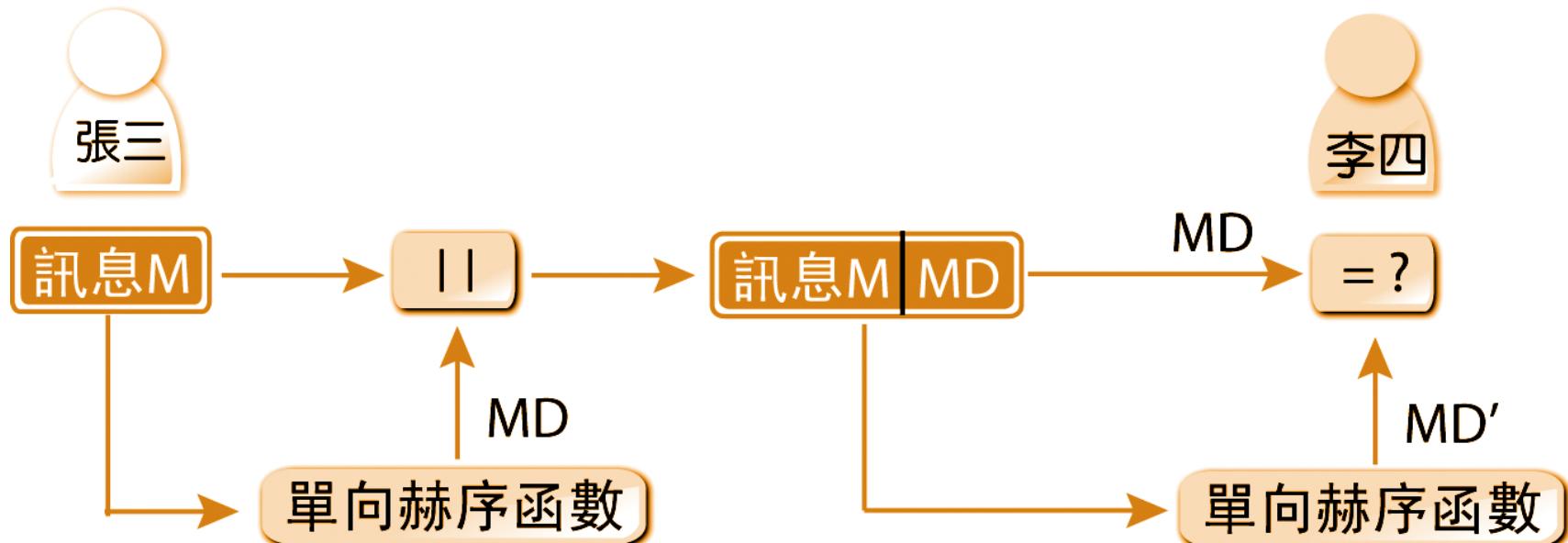
48:ybr3MfxNiYfUFx2h3Jzx4YhedM3v71yalwQ/2fBjZE:k3MfxNQ2h3Jzx8elOVZE,"dystcs.txt"
and

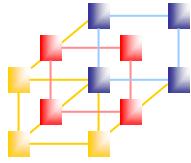
48:yr7eiwpWzQndlqzn7BX61HzvUS3v71yalwQ/2fBjZE:w7ypWcqHDF10VZE,"mhps.txt"
matched with score 50.



文件訊息完整性驗證 (1/2)

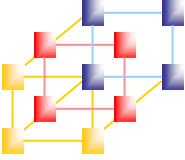
- 以單向赫序函數做文件訊息的完整驗證，其驗證方式如下：





文件訊息完整性驗證 (2/2)

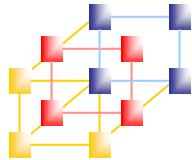
- 首先張三先將要傳送給李四的文件訊息M，經單向赫序函數運算後得到一訊息摘要MD，再將文件訊息M與訊息摘要MD一起送給李四
- 李四收到後先將文件訊息M用同樣的單向赫序函數運算，假設得到一訊息摘要MD'，李四再比較MD'是否與收到的MD相同，若相同，則李四則可確認此文件之完整性。



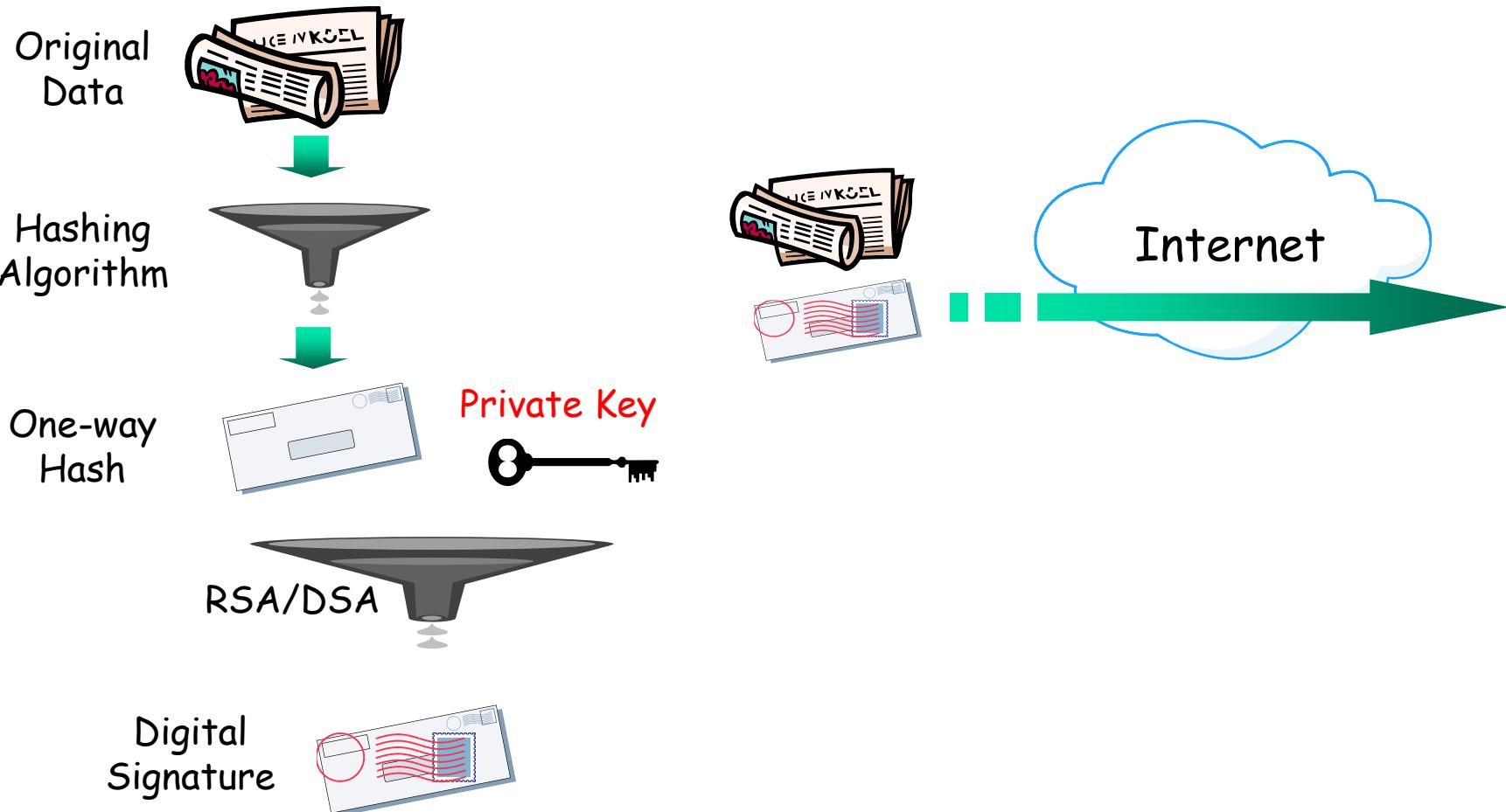
Digital Signature 數位簽章

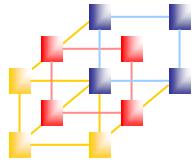


- 一般數位簽章具有下列功能：
 - 確認性(Authentication)
 - 可確認文件來源的合法性，而非經他人偽造。
 - 完整性(Integrity)
 - 可確保文件內容不會被新增或刪除。
 - 不可否認性(Non-repudiation)
 - 簽章者事後無法否認曾簽署過此文件。
- Very often digital signatures are used with hash functions, hash of a message is signed, instead of the message.

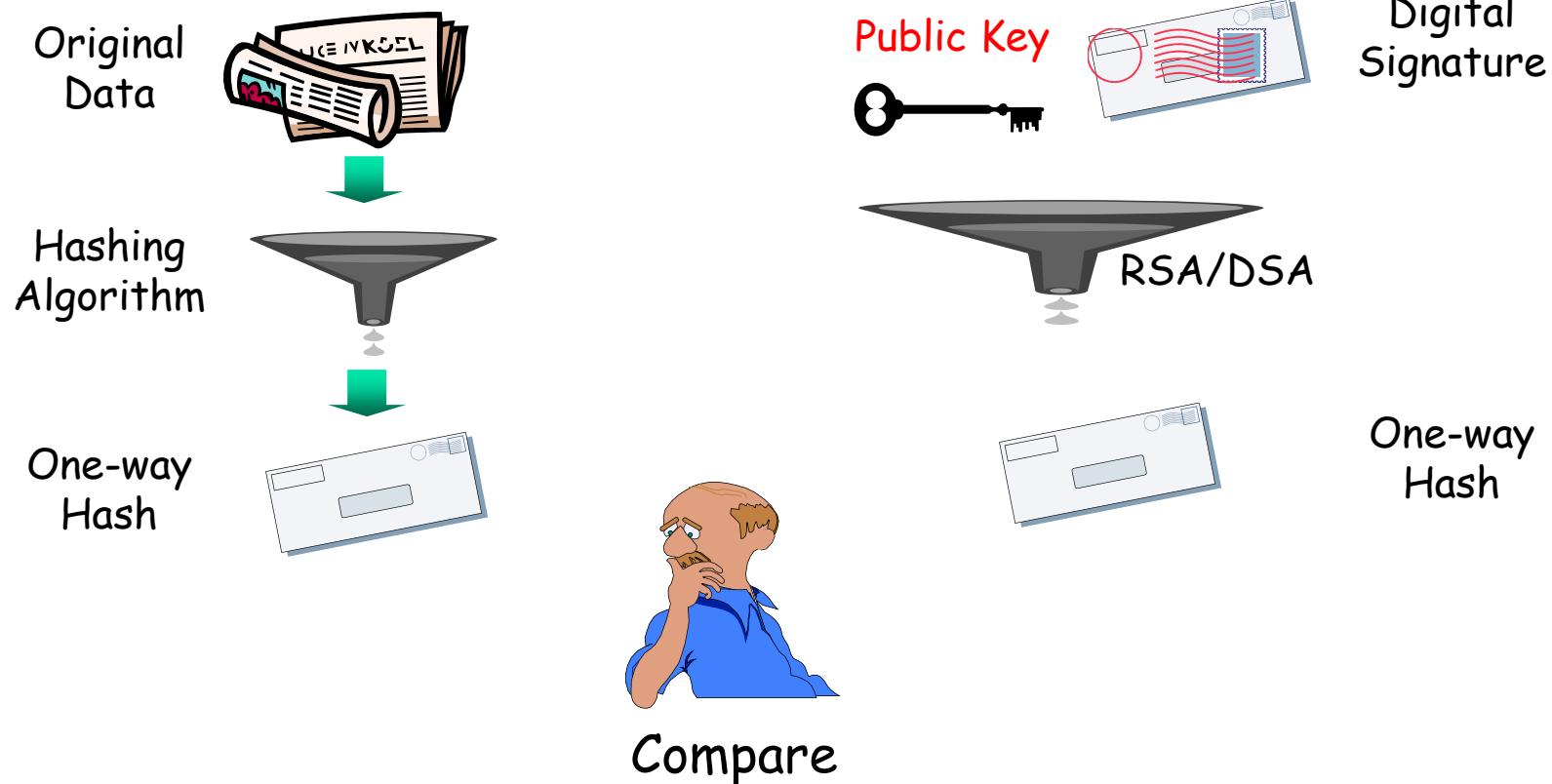


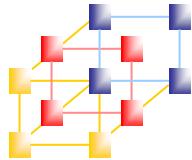
Digital Signature -- Sender



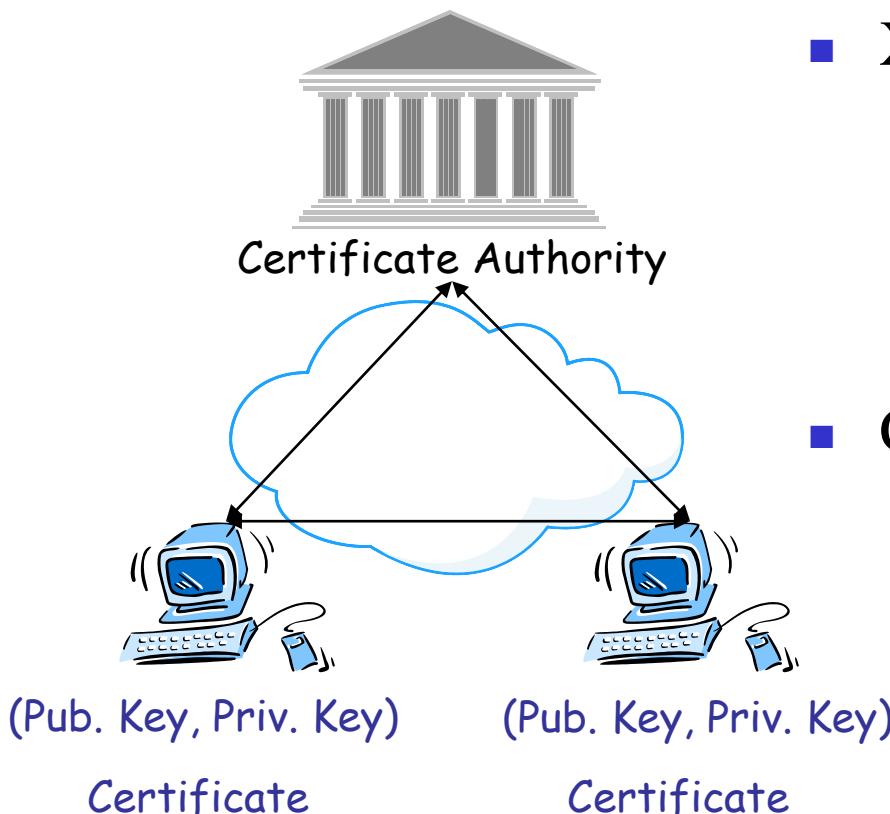


Digital Signature -- Receiver

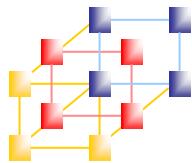




Public-Key Infrastructure



- 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

