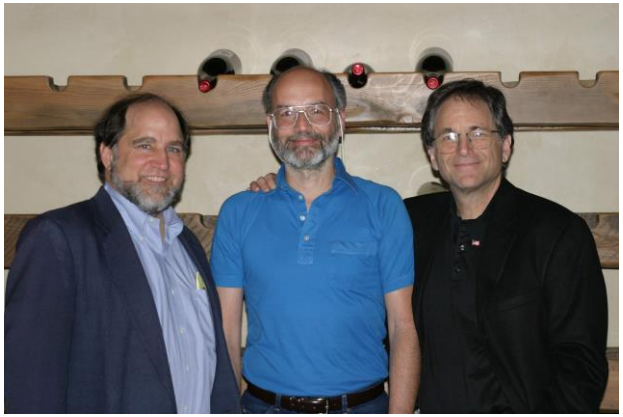
破密分析-從古典密碼學到現代密碼學

資訊安全人才培育計畫 | Hacking Weekend | MyFirstCTF Training

現代密碼 快速入門篇



Ron Rivest, Adi Shamir, and Leonard Adleman

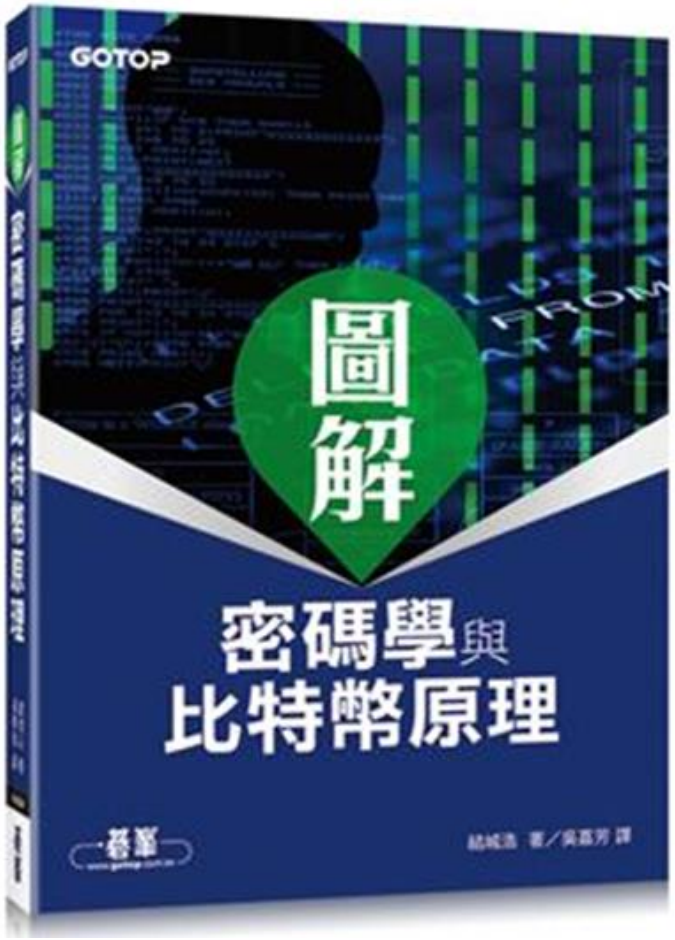
Security Mentors

臺灣好厲駭 讓你更厲害

資安實務導師培訓計畫

現代密碼

本次課程將提到**紅色部分**[推薦你閱讀此書]



第I部 密碼
第1章 進入密碼的世界
第2章 密碼的歷史
第3章 對稱式密碼（共用金鑰密碼）
第4章 區塊加密的模式
第5章 公開金鑰密碼系統
第6章 混合型密碼系統

第II部 認證
第7章 單向雜湊函數(HASH)
第8章 訊息認證碼(MAC)
第9章 數位簽章
第10章 憑證

第III部 金鑰、亂數、應用技術
第11章 金鑰
第12章 亂數
第13章 PGP
第14章 SSL/TLS
第15章 密碼技術與現實社會

Modern Cipher@openssl
本課程將主要以openssl來示範現代密碼的加解密
OpenSSLhttps://zh.wikipedia.org/wiki/OpenSSL

沒有登入 對話 貢獻 建立帳號 登入

條目 討論 台灣正體 汉 漢 閱讀 編輯 檢視歷史 搜尋維基百科

維基百科
自由的百科全書

首頁
分類索引
特色內容
新聞動態
近期變更
隨機條目

說明

說明
維基社群
方針與指引
互助客棧
知識問答
字詞轉換
IRC即時聊天
聯絡我們
關於維基百科

中文維基百科條目協作計劃專頁已建立，歡迎報名參與！ [關閉]

OpenSSL [編輯]

維基百科，自由的百科全書

在電腦網路上，**OpenSSL**是一個開放原始碼的軟體函式庫套件，應用程式可以使用這個套件來進行安全通訊，避免竊聽，同時確認另一端連線者的身分。這個套件廣泛被應用在網際網路的網頁伺服器上。

其主要函式庫是以C語言所寫成，實作了基本的加密功能，實作了SSL與TLS協定。OpenSSL可以運行在絕大多數類Unix作業系統上（包括Solaris，Linux，Mac OS X與各種版本的開放原始碼BSD作業系統），OpenVMS與Microsoft Windows。它也提供了一個移植版本，可以在IBM i（OS/400）上運作。

雖然此軟體是開放原始碼的，但其授權書條款與GPL有衝突之處，故GPL軟體使用OpenSSL時（如Wget）必須對OpenSSL給予例外。

目錄 [隱藏]

OpenSSL

OpenSSL
Cryptography and SSL/TLS Toolkit

開發者	OpenSSL專案
穩定版本	1.1.0f（2017年5月25日，4個月前[1]） [±]
作業系統	跨平台
類型	安全性加密函式庫
授權條款	OpenSSL授權條款（類Apache授權）
網站	www.openssl.org
原始碼庫	github.com/openssl/openssl

Modern Cipher@openssl
本課程將主要以openssl來示範現代密碼的加解密

https://www.openssl.org/ https://github.com/openssl/openssl

OpenSSL is a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols.

It is also a general-purpose cryptography library.

OpenSSL
Cryptography and SSL/TLS Toolkit

```
OVERVIEW
--------

The OpenSSL toolkit includes:

libssl (with platform specific naming):
    Provides the client and server-side implementations for SSLv3 and TLS.

libcrypto (with platform specific naming):
    Provides general cryptographic and X.509 support needed by SSL/TLS but
    not logically part of it.

openssl:
    A command line tool that can be used for:
       Creation of key parameters
       Creation of X.509 certificates, CSRs and CRLs
       Calculation of message digests
       Encryption and decryption
       SSL/TLS client and server tests
       Handling of S/MIME signed or encrypted mail
       And more...
```

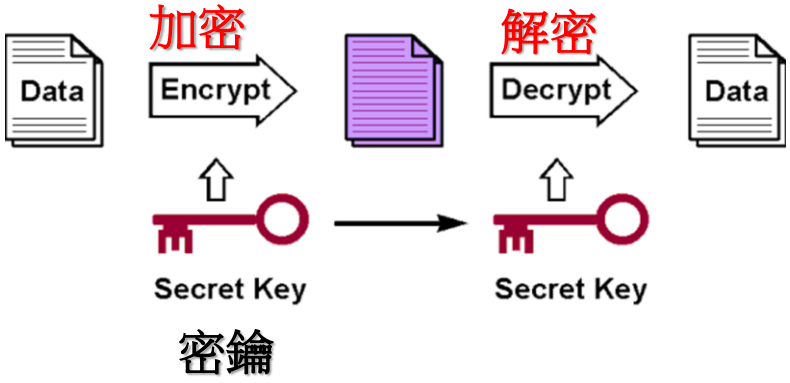
現代密碼學:第一種分類

根據type of key的分類

https://en.wikipedia.org/wiki/Cipher

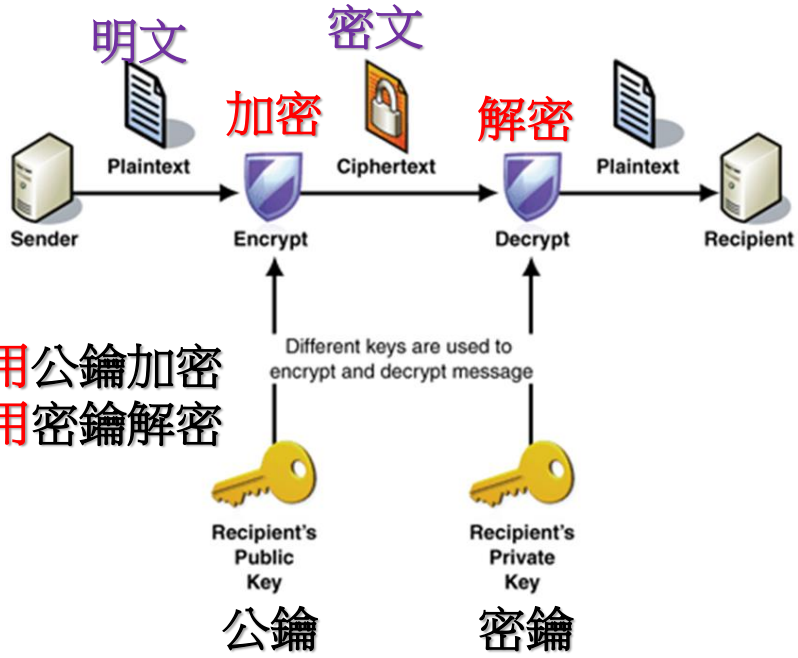
https://en.wikipedia.org/wiki/Outline_of_cryptography

對稱式密碼
symmetric key algorithms
Private-key cryptography



加密與解密
都是使用同一把key

非對稱式密碼
asymmetric key algorithms
Public-key cryptography



加密::使用公鑰加密
解密::使用密鑰解密

stream ciphers
Advanced Encryption Standard (Rijndael)

RSA – factoring(質因數分解)
El Gamal – discrete logarithm
Elliptic curve cryptography –
(discrete logarithm variant)

1

對稱式密碼

symmetric key algorithms

Private-key cryptography

Xor cipher

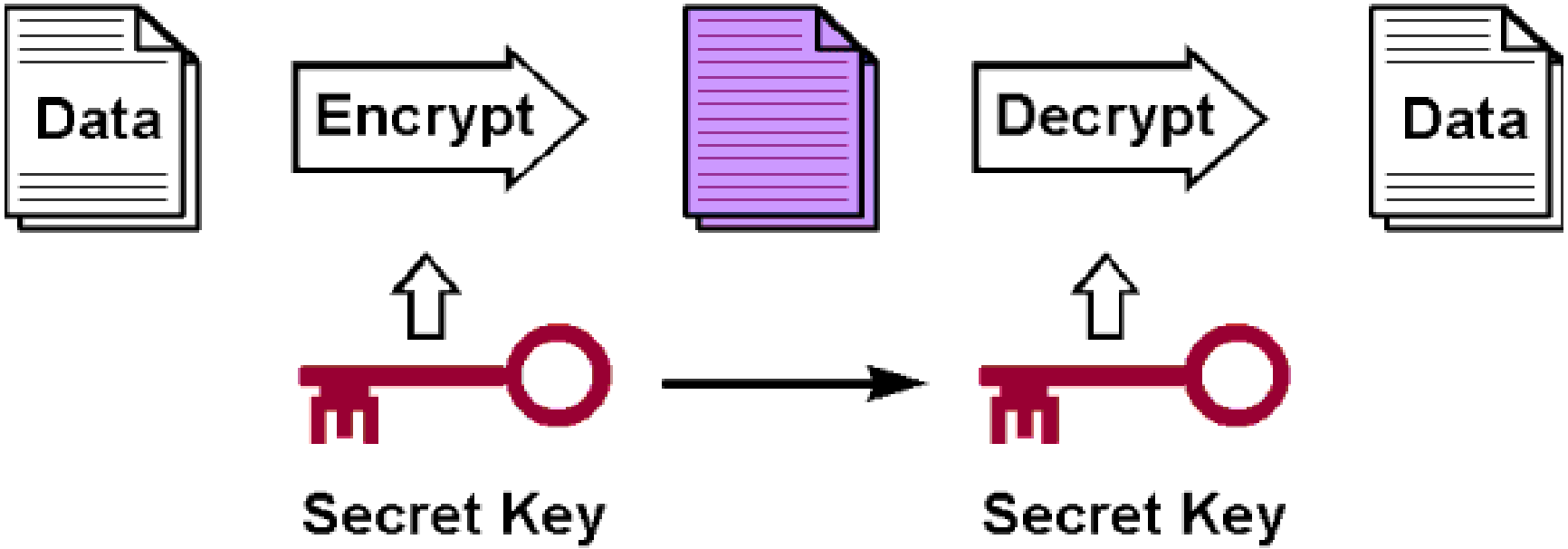
DES

Triple DES

AES

.....

對稱式密碼
symmetric key algorithms
Private-key cryptography



使用XOR 邏輯運算建構簡易的密碼(容易被破)

Boolean exclusive-OR operation

A	B	A AND B	A OR B	A ⊕ B	(A ⊕ B)⊕ B	(A ⊕ B)⊕ A
0	0	0	0	0		
0	1	0	1	1		
1	0	0	1	1		
1	1	1	1	0		

危險的對稱式密碼::XOR ciphers

NOT Instruction:
0➔1
1➔0

python實作

bin(0b1111 ^ 0b1111)

a=**0b**01010111011010010110101101101001
k=**0b**11110011111100111111001111110011
bin(a^k)

a = 3 b = 4

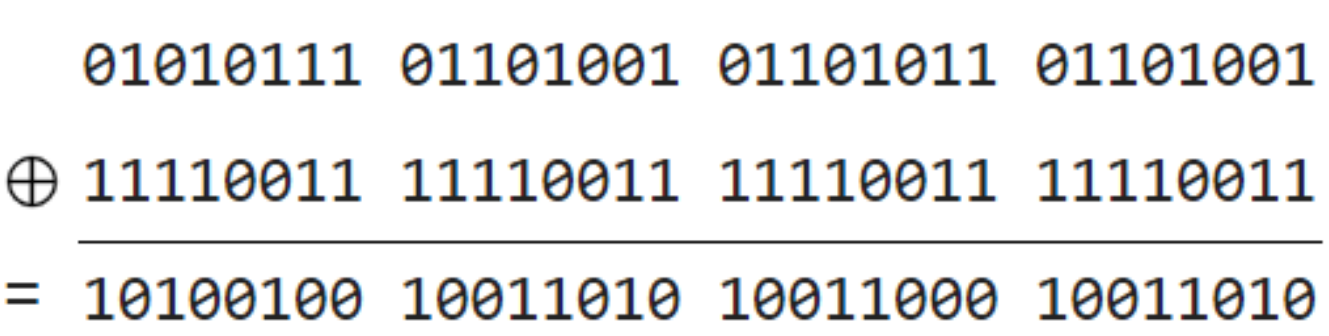
作業:
使用XOR運算時做兩整數互換

a = 4 b = 3

危險的對稱式密碼:: XOR cipher

https://en.wikipedia.org/wiki/XOR_cipher

"wiki" ➔ (01010111 01101001 01101011 01101001 in 8-bit ASCII)



```
  01010111 01101001 01101011 01101001
⊕ 11110011 11110011 11110011 11110011
-----------------------------------
= 10100100 10011010 10011000 10011010
```



```
  ⊕ 11110011 11110011 11110011 11110011
  = 10100100 10011010 10011000 10011010
```

And conversely, for decryption:

```
    10100100 10011010 10011000 10011010
  ⊕ 11110011 11110011 11110011 11110011
  = 01010111 01101001 01101011 01101001
```

The XOR operator is extremely common as a component in more complex ciphers. By itself, using a constant repeating key, a simple XOR cipher can trivially be broken using frequency analysis. If the content of any message can be guessed or otherwise known then the key can be revealed. Its primary merit is that it is simple to implement, and that the XOR operation is computationally inexpensive. A simple XOR (i.e. using the same key for xor operation on the whole data) cipher is therefore sometimes used for hiding information in cases where no particular security is required.

python實作

作業解答

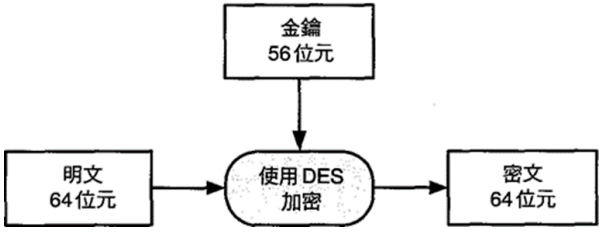
```
>>> a ^= b
>>> b ^= a
>>> a ^= b
```

資料加密標準
DES::Data Encryption Standard

https://zh.wikipedia.org/wiki/資料加密標準

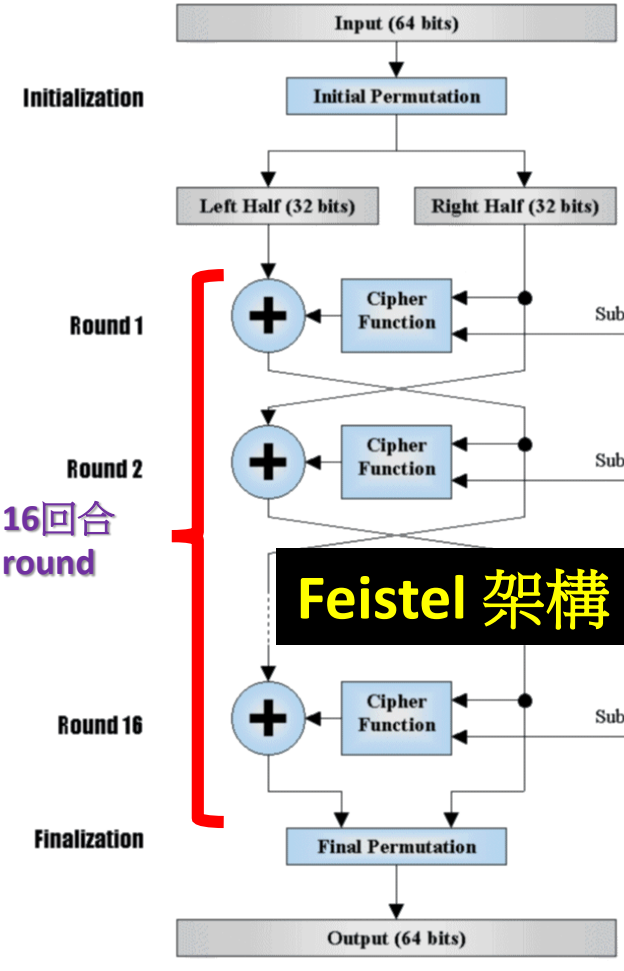
DES對稱演算法的主要架構:

- DES是一種**對稱密鑰加密塊密碼(Block cipher)**演算法
- 1976年被**美國聯邦政府國家標準局(NIST)**確定為聯邦資料處理標準（FIPS），隨後在國際上廣泛流傳。



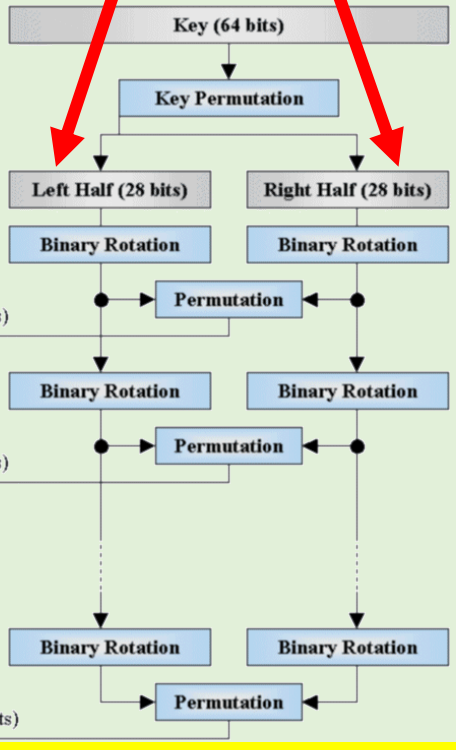
同學可以了解基本架構,更多細節可上大學再學習更多,可以先學習使用openssl來實作加解密!

64-bits為區塊為單位進行加緊密



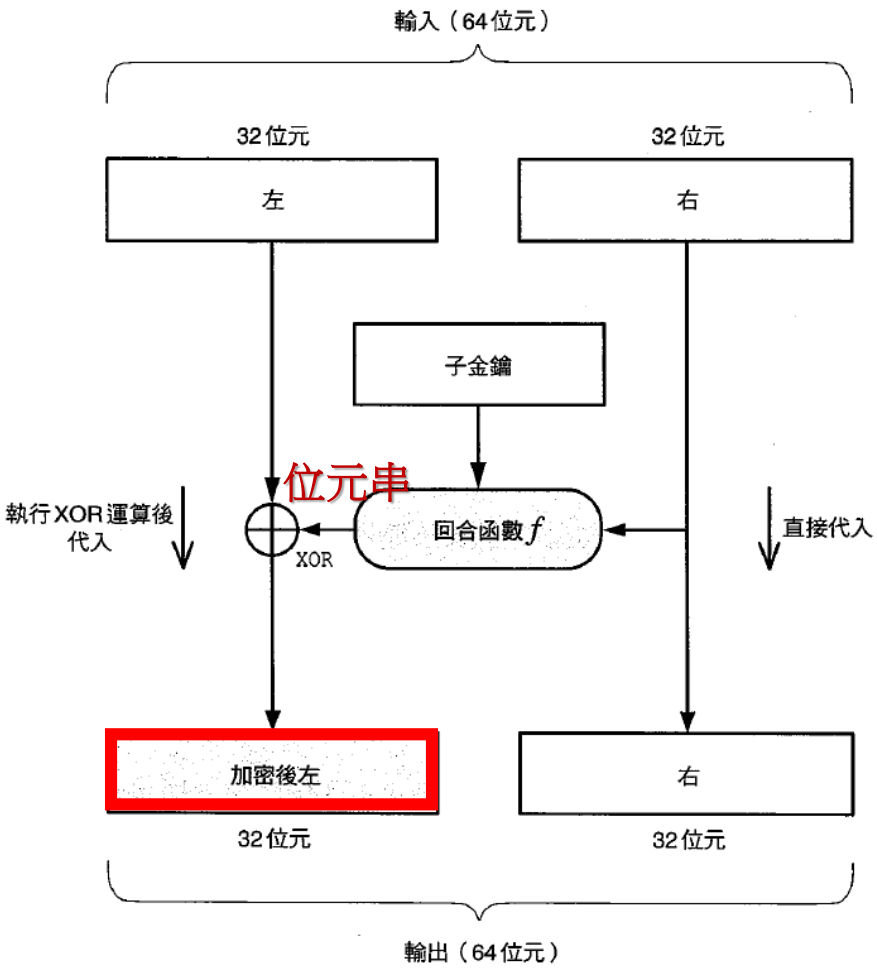
金鑰(keys)

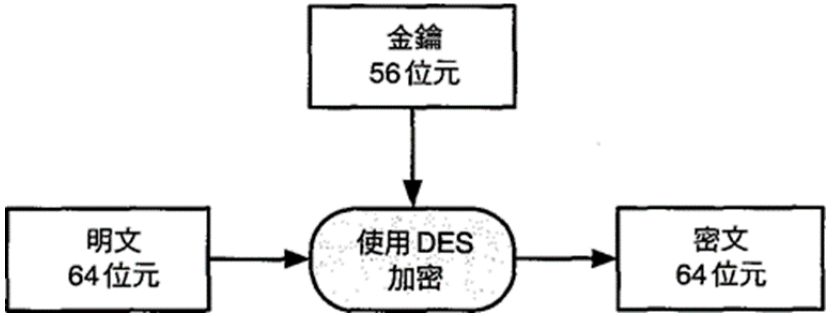
使用56位金鑰(keys)



金鑰的實際位元長度是 56 位元。但是每 7 位元就有 1位元的錯誤檢核資料，所以DES 的金鑰長度是 64 位元。

資料加密標準
DES::Data Encryption Standard



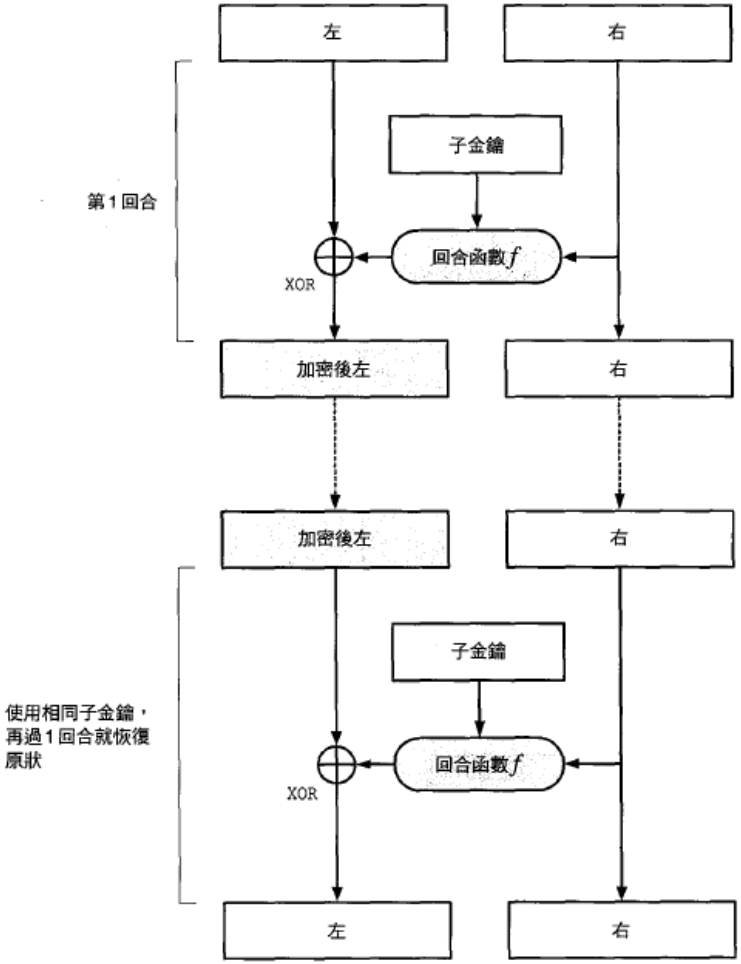


- 一回合的輸入分成左右兩邊。
- 右邊直接輸出成「 右」。
- 將右帶入回合函數 f
- **回合函數f**使用右及子金鑰， 計算出隨機位元串。
- 將得到的位元串與左進行 XOR 運算後，得到的結果成為加密後左。

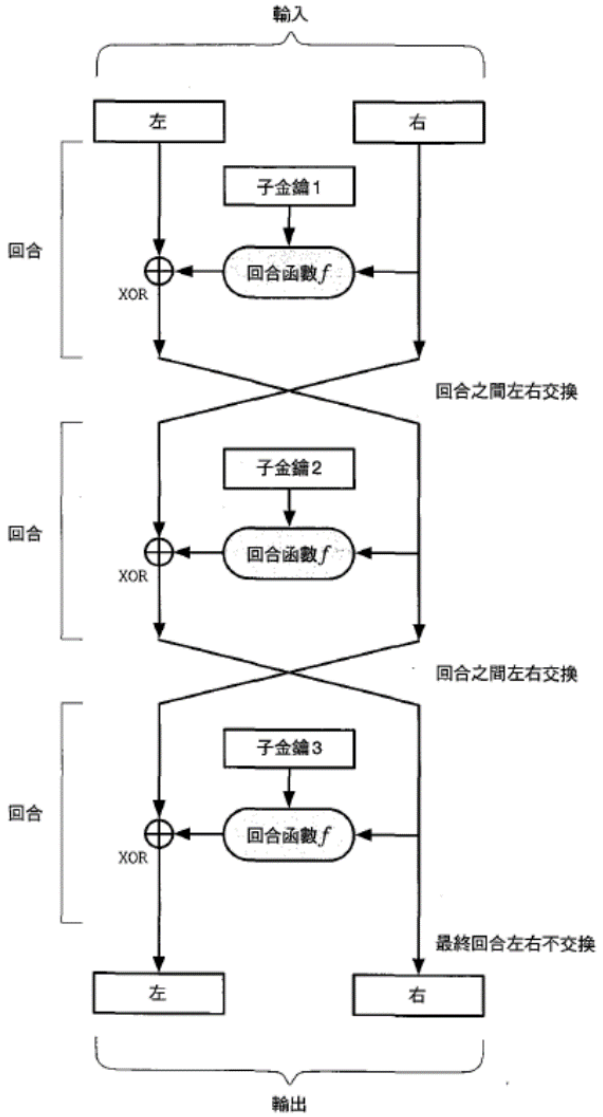
資料加密標準
DES::Data Encryption Standard

Feistel 架構

把 1回合的 Feistel 網路輸出，再次放 入相同子金鑰的 Feistel 網路中,不論回合函數 f是哪種函數， 都會正確恢復原狀



VS



三重DES(Triple-DES)

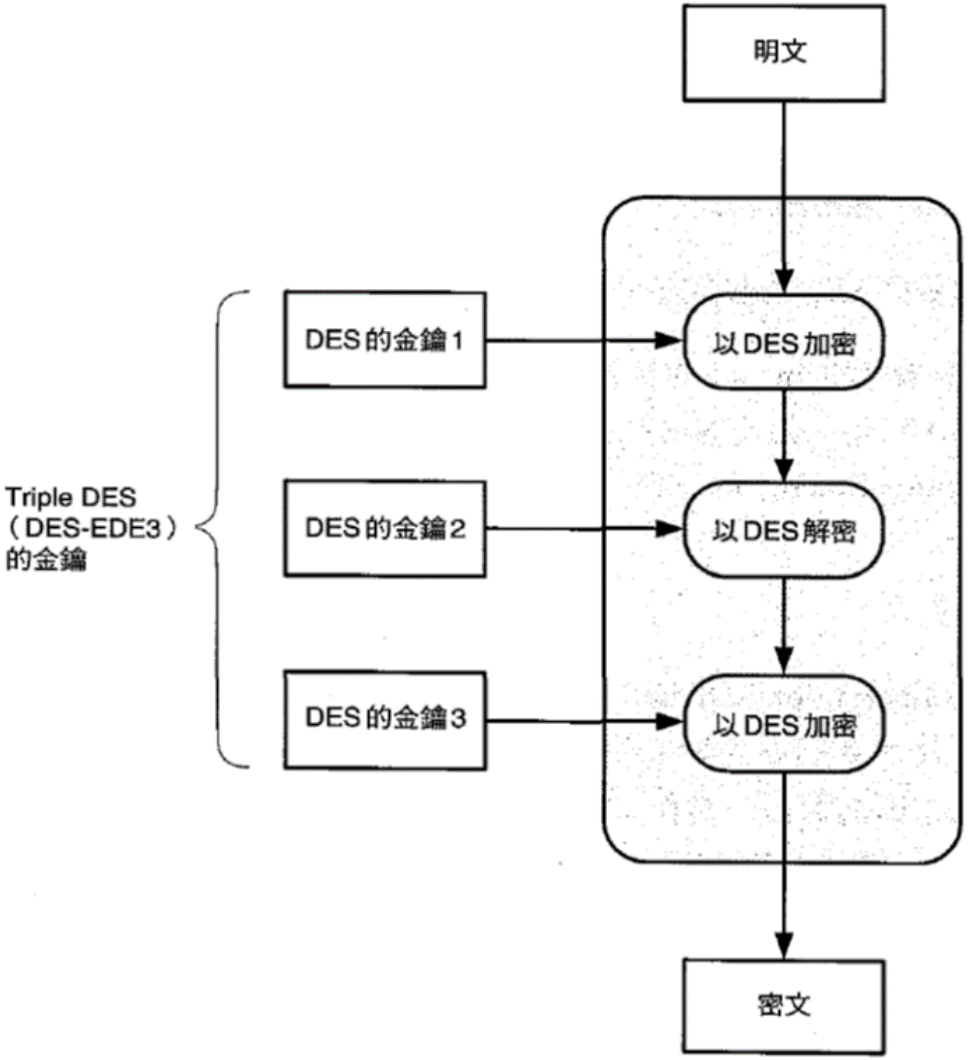
https://zh.wikipedia.org/wiki/三重資料加密演算法

三重資料加密演算法
|Triple Data Encryption Algorithm
|TDEA Triple DEA|3DES|Triple DES
是一種對稱密鑰加密塊密碼

對每個資料塊應用三次資料加密標準（**DES**）演算法。

由於電腦運算能力的增強，原版**DES**密碼的金鑰長度變得容易被暴力破解

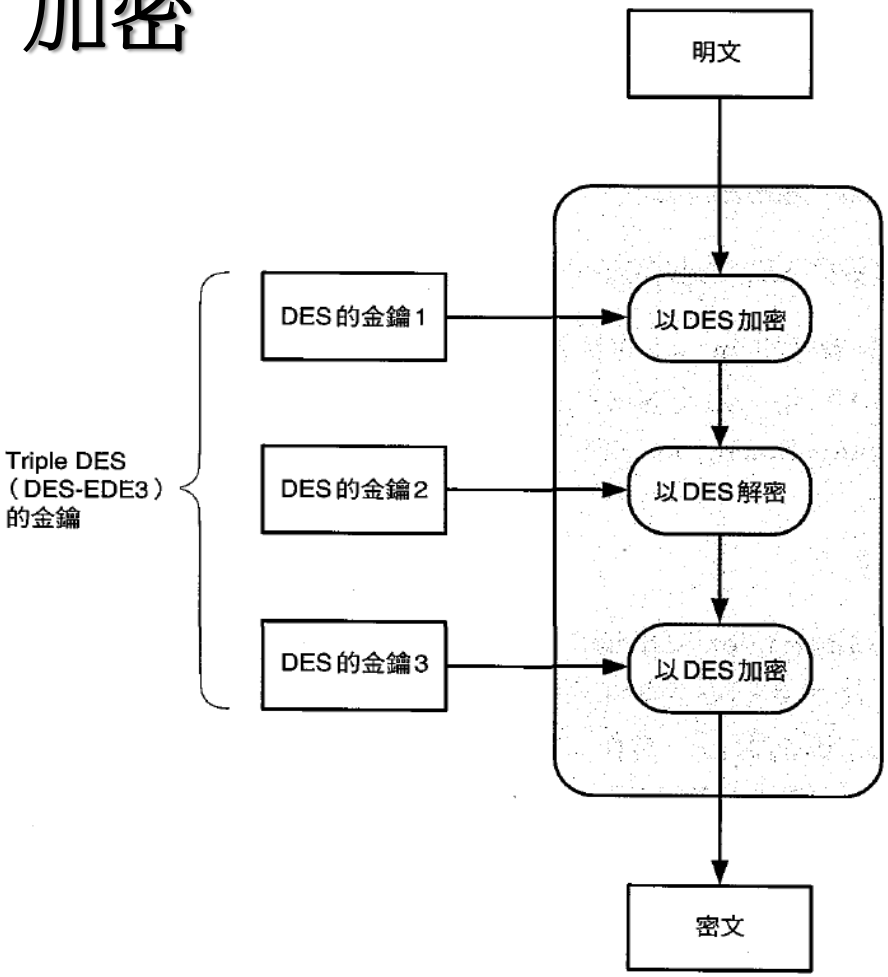
3DES是設計用來提供一種相對簡單的方法，通過增加**DES**的金鑰長度來避免類似的攻擊，而不是設計一種全新的塊密碼演算法。



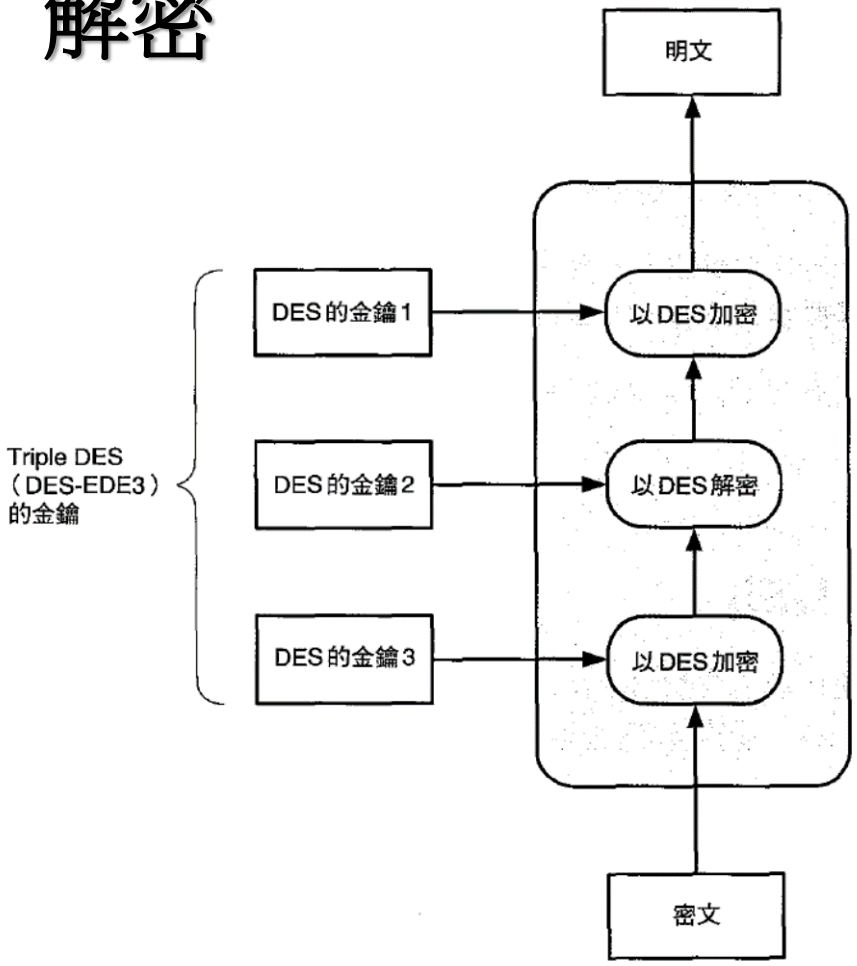
三重DES(Triple-DES)

https://zh.wikipedia.org/wiki/三重資料加密演算法

加密

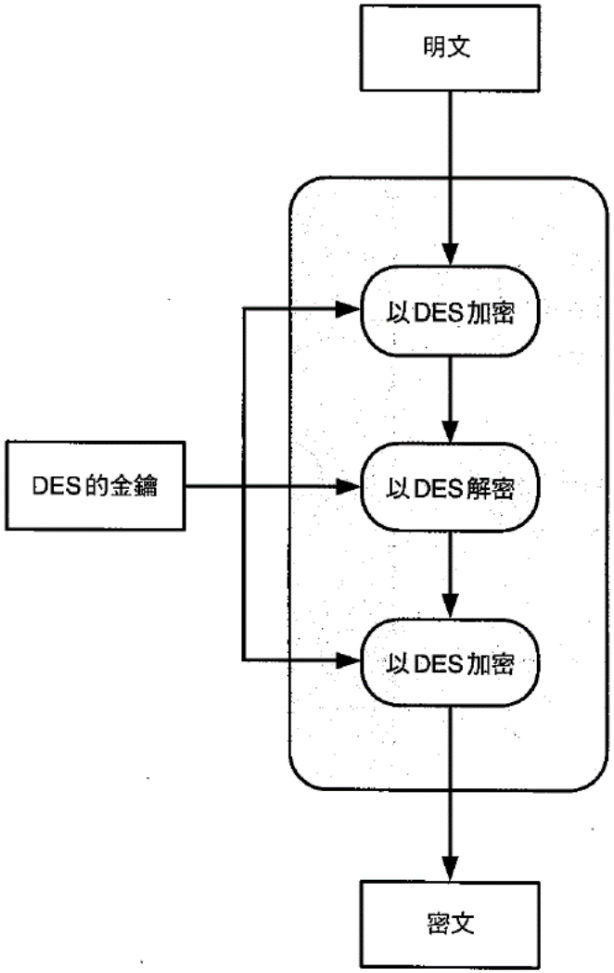


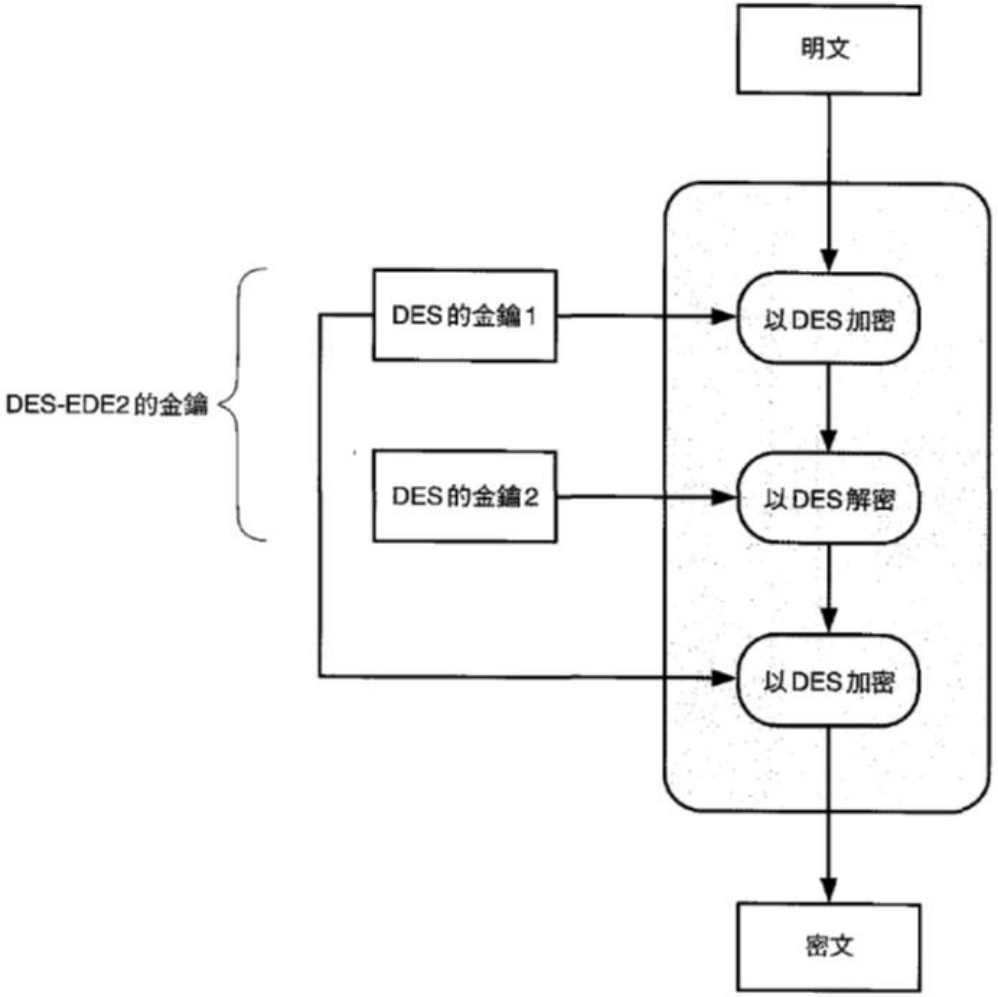
解密



三重DES(Triple-DES)

https://zh.wikipedia.org/wiki/三重資料加密演算法





DES/3DES加解密using **openssl**

列出 OpenSSL 提供的對稱式加解密演算法	openssl enc –h 注意輸出顯示中的Cipher Types
使用 DES 加密	openssl **des** -in file -out file.des 執行後，OpenSSL 會提示使用者由鍵盤上輸入加密之密碼，如下: enter des-cbc encryption password: 需要注意的是，為了安全性，此時不管鍵盤輸入什麼，畫面上都不會出現任何字元，否則若旁人經過時，可能會故意或不經意的記下你的密碼。直到輸入完成後，按下鍵盤上的 "Enter" 鍵即可。 OpenSSL 會再一次要求使用者輸入一次相同的密碼 加密的檔案將以 file.des 的名稱存在於磁碟中。
使用 DES 解密	openssl **des -d** -in file.des -out file 執行後，OpenSSL 會提示使用者由鍵盤上輸入加密之密碼，如下: enter des-cbc encryption password: 需要注意的是，為了安全性，此時不管鍵盤輸入什麼，畫面上都不會出現任何字元，否則若旁人經過時，可能會故意或不經意的記下你的密碼。直到輸入完成後，按下鍵盤上的 "Enter" 鍵即可。 此時若使用者輸入了正確的密碼，就會成功將 file.des 解密之檔案，以 file 的檔案名稱存在於磁碟上。

	使用 OpenSSL 的 Triple DES 加解密
使用 Triple DES加密	openssl **des3** -in file -out file.des3
使用 Triple DES 解密	openssl **des3 -d** -in file.des3 -out file

https://www.openfoundry.org/tech-column/8609

2

非對稱式密碼

asymmetric key algorithms

Public-key cryptography

RSA

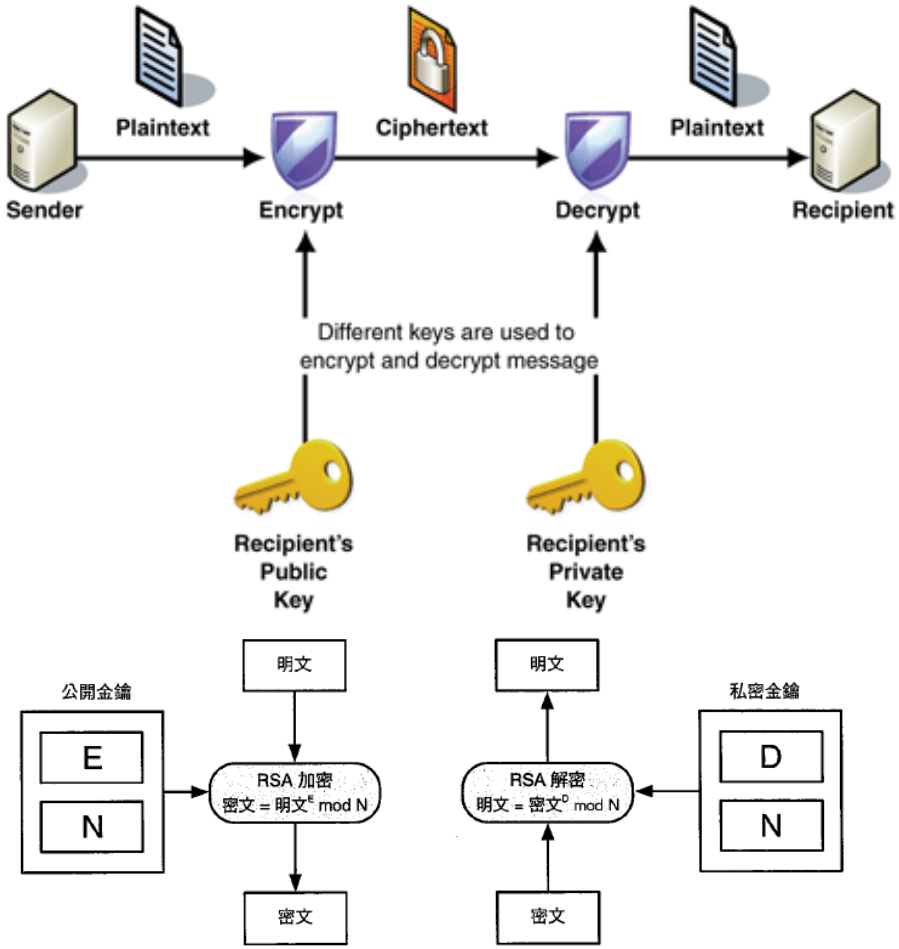
ElGamal

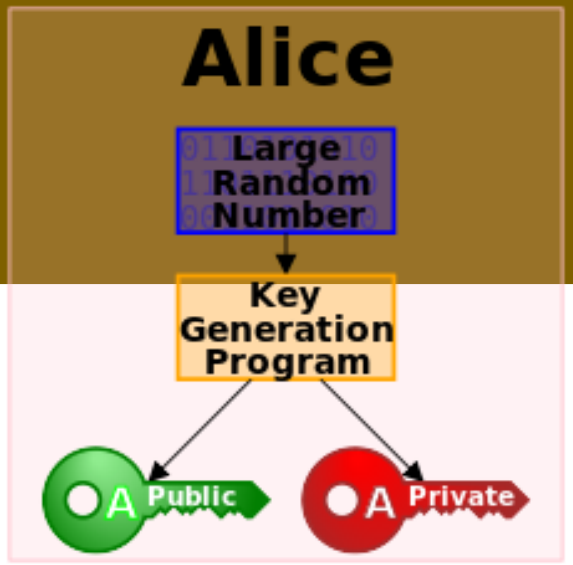
elliptic curve techniques

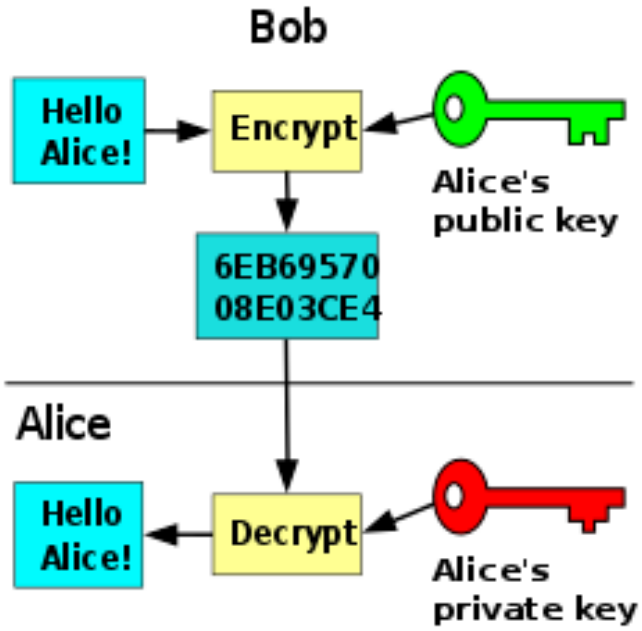
https://en.wikipedia.org/wiki/Public-key_cryptography

非對稱式密碼
asymmetric key algorithms
Public-key cryptography

https://en.wikipedia.org/wiki/Public-key_cryptography







非對稱式密碼
asymmetric key algorithms
Public-key cryptography

Encryption

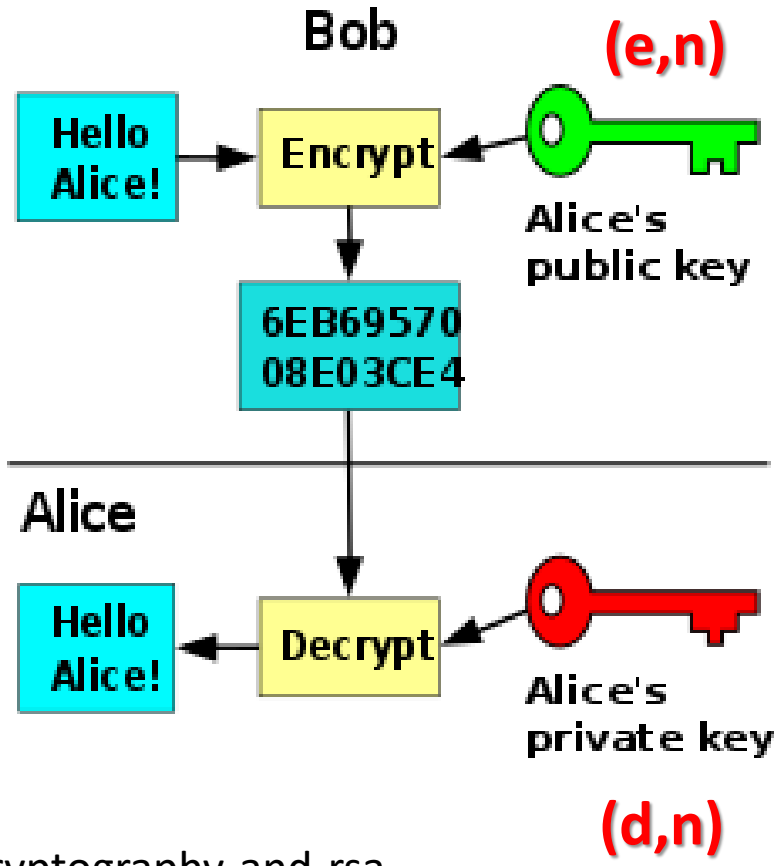
$$cipher = (message)^e \bmod n$$

Decryption

$$message = (cipher)^d \bmod n$$

x mod y means the remainder of x divided by y

RSA加密演算法- 维基百科



https://www.slideshare.net/shafaan/public-key-cryptography-and-rsa

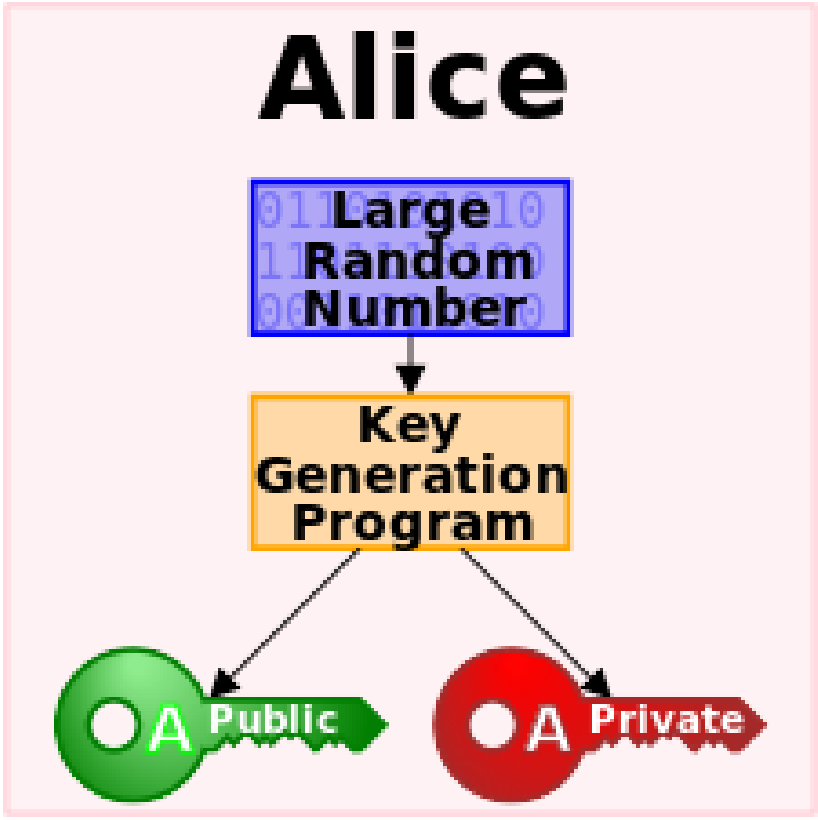
非對稱式密碼
asymmetric key algorithms
Public-key cryptography

Key Generation

1. Generate two large prime numbers, p and q
2. Let $n = pq$
3. Let $m=\phi(n) = (p-1)(q-1)$
4. Choose a small number e, co prime to m, with GCD $(\phi(n), e)= 1$; $1<e<\phi(n)$
5. Find d, such that $de \bmod \phi(n) = 1$

Publish e and n as the public key.
Keep d and m as the secret key.

RSA加密演算法- 维基百科



https://www.slideshare.net/shafaan/public-key-cryptography-and-rsa

非對稱式密碼
asymmetric key algorithms
Public-key cryptography

Key generation(產生key pair)
(1)Select primes p=17, q=11
(2)Compute n=pq=187
(3)Compute ϕ(n)=(p-1)(q-1)=160
(4)Select e=7➔GCD(7,160)=1
(5)Compute d: d=23 ➔7*23 mod 160=1
(use the extended Euclid's algorithm)

公鑰pub={e,n}={7,187}
私鑰pri={d,n}={23,187}
==========
明文M=88
加密(Encrypt): 88^7mod 187
88^7mod 187 = 11(密文C)
==========
解密Decrypt C=11: 11^23mod 187
M=11^23 mod 187=88

Key Generation

1. Generate two large prime numbers, p and q
2. Let $n = pq$
3. Let $m=\phi(n) = (p-1)(q-1)$
4. Choose a small number e, co prime to m, with GCD $(\phi(n), e)= 1$; $1<e<\phi(n)$
5. Find d, such that $de \bmod \phi(n) = 1$

Publish e and n as the public key.
Keep d and m as the secret key.

Encryption

$cipher = (message)^e \bmod n$

Decryption

$message = (cipher)^d \bmod n$

https://www.slideshare.net/shafaan/public-key-cryptography-and-rsa

不同的非對稱式密碼各有其安全性基礎.....
RSA非對稱式密碼的安全::質因數分解[15=5*3]

12301866845301177551304949583849627207728535695953347921973224
52151726400507263657518745202199786469389956474942774063845925
19255732630345373154826850791702612214291346167042921431160222
12404792747377940806653514195974595856902143413

=

33478071698956898786044169
84821269081770479498371376
85689124313889828837938780
02287614711652531743087737
814467999489

×

36746043666799590428244633
79962795263227915816434308
76426760322838157396665112
79233373417143396810270092
798736308917

http://www.ruanyifeng.com/blog/2013/07/rsa_algorithm_part_two.html

RSA加解密using **openssl**

產生私鑰

openssl **genrsa** -out private.pem

預設會產生長度為 **512 bit** 的私鑰

openssl **genrsa** -out private.pem **1024**

產生 **1024 bit** 長度的私鑰

- 愈長的私鑰被破解的機率愈低
- 但使用加密與解密的時間也會愈長

使用 **RSA** 私鑰產生相對應的公鑰	openssl **rsa -in private.pem** -out public.pem -outform PEM –pubout **-out" 參數指定產生的公鑰檔案名稱** **"-outform" 參數指定公鑰的輸出格式** **"-pubout" 參數結尾** **執行後，OpenSSL 會產生 public.pem 的檔案在磁碟中**
使用公鑰加密檔案	openssl **rsautl -encrypt -inkey public.pem** -pubin **-in file -out file.rsa** **"-inkey" 參數指定密鑰檔案，"-pubin" 參數將公鑰產生於加密檔案中，** **"-in" 參數指定欲加密的檔案，以及 "-out" 參數指定加密後的檔案名稱** **執行後，OpenSSL 會產生 file.rsa 的檔案在磁碟中。** **RSA 非對稱式加解密演算法因為先天的限制，無法加密過大的檔案**
使用 私鑰解密檔案	openssl rsautl -decrypt -inkey private.pem -in file.rsa -out file

https://www.openfoundry.org/tech-column/8615