Crypto

OAlienO

2018/12/14

Table of Contents

Group

Definition

- A group $(G, \cdot)$ is a set G with a binary operation $\cdot$ such that
- 1. $*$ is associative : $(a * b) * c = a * (b * c)$
- 2. $\exists\ e \in G$ such that $\forall\ x \in G : e \cdot x = x \cdot e = x$
- 3. $\forall\ a \in G : \exists\ a^{-1} \in G$ such that $a \cdot a^{-1} = a^{-1} \cdot a = e$

Cyclic Subgroup

- Let G be a group and $a \in G$
- $H = \{a^n : n \in \mathbb{Z}\}$ is a cyclic subgroup generated by a, denoted $\langle a \rangle$
- a group is called cyclic if $\exists\ a \in G$ sucht that $G = \langle a \rangle$, the element a is called a generator for G

Diffie Hellman Key Exchange

- Generate a large prime p, and a generator α of $\mathbb{Z}_p^*$
- Alice select a random integer $a : 1 \leq a < p - 1$
- Bob select a random integer $b : 1 \leq b < p - 1$
- Alice and Bob exchange g^a and g^b
- Alice and Bob share g^{ab} secret

Discrete Logarithm Problem

- Diffie Hellman Key Exchange 的安全性來自 Discrete Logarithm Problem
- 更一般性的 Diffie Hellman Key Exchange 就是將 $\mathbb{Z}_p^*$ 換成任意的 Finite Cyclic Group G，例如 Elliptic Curve

Discrete Logarithm Problem (DLP)

- A generator α of a finite cyclic group G
- An element $\beta \in G$
- Find x such that $\alpha^x = \beta$

Public Key Encryption 簡介

- 公鑰加密系統會有**公鑰** e 和**私鑰** d 分別做加密和解密
- 在安全的公鑰加密系統中，不可能從**公鑰** e 計算出**私鑰** d

RSA 簡介

- 1997 年由 Ron Rivest, Adi Shamir, Leonard Adleman 提出的非對稱式加密演算法
- 廣泛應用於
  - https 加密連線
  - ssh 公鑰認證
  - WannaCry

RSA 產生密鑰

```python
def genkeys():
    e = 65537
    while True:
        p, q = getPrime(512), getPrime(512)
        n, phi = p * q, (p - 1) * (q - 1)
        if GCD(e, phi) == 1:
            d = inverse(e, phi)
            return (n, e), (n, d)
```

RSA 加解密

```python
def enc(m, public):
    n, e = public
    return pow(m, e, n)

def dec(c, private):
    n, d = private
    return pow(c, d, n)
```

費馬小定理 (Fermat's little theorem)

條件

a 是正整數, p 是質數, gcd(a, p) $= 1$

費馬小定理

$$a^{p-1} \equiv 1 \pmod{p}$$

歐拉函數 (Euler's totient function)

定義

$$\begin{aligned}\varphi(n) &= |\{1 \le x \le n \mid gcd(x, n) = 1\}| \\ &= n \prod_{p|n} \left(1 - \frac{1}{p}\right)\end{aligned}$$

範例

$$\begin{aligned}\varphi(6) &= |\{1, 5\}| = 2 \\ \varphi(24) &= |\{1, 5, 7, 11, 13, 17, 19, 23\}| = 8 \\ \varphi(pq) &= (p - 1)(q - 1)\end{aligned}$$

RSA 正確性

目標

驗證 $m^{ed} \equiv m \pmod{n}$

拆解成小問題

分別驗證

1. $m^{ed} \equiv m \pmod{p}$
2. $m^{ed} \equiv m \pmod{q}$

再用中國剩餘定理拼起來得證 $m^{ed} \equiv m \pmod{n}$

RSA 正確性

Lemma

$$ed \equiv 1 \pmod{\varphi(n)}$$

$$ed = k\varphi(n) + 1 \text{ for some k}$$

$$= k(p-1)(q-1) + 1$$

RSA 正確性

驗證 $m^{ed} \equiv m \pmod{p}$

if gcd(m, p) = 1

$$\rightarrow m^{ed} = m^{k(p-1)(q-1)+1} = (m^{(p-1)})^{k'} m \equiv m \pmod{p}$$

if gcd(m, p) = p

$$\rightarrow m^{ed} \equiv 0 \equiv m \pmod{p}$$

驗證 $m^{ed} \equiv m \pmod{q}$

By the same argument

RSA Problem (RSAP)

RSA Problem

Given composite integer n = pq where p, q are primes
And ciphertext $c \in \{0, 1, \cdots, n-1\}$
Find m such that $m^e \equiv c \pmod{n}$

RSA Problem (RSAP)

Remark

- The security of RSA public-key encryption depends on the intractability of RSA problem
- Actually, the RSA problem is that of finding e^{th} roots modulo a composite integer n with unknown factorization
- It is widely believed that the RSA problem and the **integer factorization problem** are computationally equivalent

Relation to integer factorization

factor n $\longrightarrow$ obtain private key

如果我們可以分解 n
就可以順著原本的步驟產生私鑰，進而解密密文

Relation to integer factorization

Lemma 1

Given a prime p

$$x^2 \equiv 1 \pmod{p} \Rightarrow x \equiv \pm 1 \pmod{p}$$

Proof

$$x^2 \equiv 1 \pmod{p}$$
$$\Rightarrow (x-1)(x+1) \equiv 0 \pmod{p}$$
$$\Rightarrow (x-1) \equiv 0 \pmod{p} \text{ or } (x+1) \equiv 0 \pmod{p}$$
$$\Rightarrow x \equiv \pm 1 \pmod{p}$$

Relation to integer factorization

Lemma 2

Given primes p, q and a composite number $n = pq$

nontrivial square root of 1 modulo n $\Rightarrow$ factor n

Proof

$x^2 \equiv 1 \pmod{n} \Rightarrow x \equiv \pm 1 \pmod{p}$ and $x \equiv \pm 1 \pmod{q}$

x has four possible solutions modulo n

$x \equiv 1 \pmod{p}$ and $x \equiv 1 \pmod{q} \Rightarrow x \equiv 1 \pmod{n}$

$x \equiv -1 \pmod{p}$ and $x \equiv -1 \pmod{q} \Rightarrow x \equiv -1 \pmod{n}$

$x \equiv 1 \pmod{p}$ and $x \equiv -1 \pmod{q} \Rightarrow x \equiv y \pmod{n}$

$x \equiv -1 \pmod{p}$ and $x \equiv 1 \pmod{q} \Rightarrow x \equiv -y \pmod{n}$

$x \equiv \pm y \Rightarrow 1 < gcd(x-1, n) = p$ or $q < n$

Relation to integer factorization

obtain private key → factor n

$\exists k, t, r : ed - 1 = k\varphi(n) = 2^t r$

$\forall g \in \mathbb{Z}_n^* : g^{2^t r} = g^{k\varphi(n)} \equiv 1 \pmod{n}$

$\exists x_0, \cdots, x_i \neq 1 : g^r, g^{2^1 r}, \cdots, g^{2^t r} = x_0, \cdots, x_i, 1, \cdots, 1 \pmod{n}$

$x_i \neq -1 \Rightarrow 1 < gcd(x_i - 1, n) < n \Rightarrow$ factor n

repeatedly select different g until $x_i \neq -1$

Homomorphic Property

- Homomorphic Property 這個性質就是對密文作運算再解密，跟解密完再做運算的結果是一樣的
- RSA has multiplicative homomorphic property
- $E(m_1)E(m_2) = m_1^e m_2^e = (m_1 m_2)^e = E(m_1 m_2)$
- Leads to chosen-ciphertext attack

Factoring Tools

- http://www.factordb.com/index.php
- https://github.com/DarkenCode/yafu

How to pick large primes p, q

- $|p - q|$ 太小 $\rightarrow p \approx q \approx \sqrt{n}$
- 建議 p, q 要是 strong primes ... 嗎?
- random primes are no less secure than strong primes [1]

Strong Primes

- p - 1 has a large prime factor, denoted r (Pollard's [2])
- p + 1 has a large prime factor (Williams[3])
- r - 1 has a large prime factor (Cycling Attack)

Pollard's p - 1 Algorithm

假設

- 正整數 a, 合數 n, 質數 p
- gcd(a, p) = 1 且 p | n

Pollard's p - 1 Algorithm

$$a^{p-1} \equiv 1 \pmod{p}$$
$$a^{k(p-1)} \equiv 1 \pmod{p}$$
$$a^{k(p-1)} - 1 \equiv 0 \pmod{p} \text{ for some } k$$
$$p \mid gcd(a^{k(p-1)} - 1, n)$$

Pollard's p - 1 Algorithm

Pollard's p - 1 Algorithm (cont.)

測試 $gcd(2^{1}-1,n), gcd(2^{1\times2}-1,n), gcd(2^{1\times2\times3}-1,n), \cdots$
只要 $p-1 \mid 1\times2\times\cdots,\ \ gcd(2^{1\times2\times\cdots}-1,n)>1$

Pollard's p - 1 Algorithm

```python
def pollard(n):
    a = 2
    b = 2
    while True:
        a = pow(a, b, n)
        d = gcd(a - 1, n)
        if 1 < d < n: return d
        b += 1
```

How to choose public exponent e

- e 太小 $\rightarrow$ direct eth root, broadcast attack
- e 太大 $\rightarrow$ 加密很慢
- 常見的 e 會選 $2^x + 1$ 這種形式的數，例如 $2^{16} + 1 = 65537$，這樣在做 Square and Multiply 時只需要 16 + 1 次運算

Square and Multiply

```python
def SquareAndMultiply(x, y):
    if y == 0: return 1
    k = fastpower(x, y // 2) ** 2
    return k * x if y % 2 else k
```

Direct eth Root

- 滿足 $m < n^{\frac{1}{e}} \rightarrow m^e < n$
- 直接取 eth root 就可以還原 m

Broadcast Attack

- 用 e 個不同的 n 加密 m，中國剩餘定理可以直接解回 m
- 以 e = 3 為例

$$
\begin{aligned}
m^3 &\equiv c_1 \pmod{n_1} \\
m^3 &\equiv c_2 \pmod{n_2} \\
m^3 &\equiv c_3 \pmod{n_3} \\
\text{Use CRT, } m^3 &\equiv c \pmod{n_1 n_2 n_3} \\
m^3 &< n_1 n_2 n_3 \rightarrow m^3 = c \rightarrow \text{direct eth root}
\end{aligned}
$$

How to choose private exponent d

- d 是從 e 算出來的，所以實際上我們不是在選 d
- d 太小 $\rightarrow$ Wiener's attack, Boneh-Durfee's attack

Wiener's Attack

Wiener's Attack

條件: $d < \frac{1}{3}n^{\frac{1}{4}}$
結果: 分解 n

Continued Fraction

- $$\frac{13}{17} = 0 + \cfrac{1}{1 + \frac{1}{3+\frac{1}{4}}}$$
- $\frac{13}{17}$ 的 continued fraction expansion 是 $[0, 1, 3, 4]$

Continued Fraction

- $\frac{13}{17}$ 的 convergents of the continued fraction expansion 是

$$
\begin{aligned}
c_0 &= 0 = \frac{0}{1} \\
c_1 &= 0 + \frac{1}{1} = \frac{1}{1} \\
c_2 &= 0 + \frac{1}{1 + \frac{1}{3}} = \frac{3}{4} \\
c_3 &= 0 + \frac{1}{1 + \frac{1}{3 + \frac{1}{4}}} = \frac{13}{17}
\end{aligned}
$$

Legendre's theorem in Diophantine approximations

Legendre's theorem in Diophantine approximations

給定 $\alpha \in \mathbb{R}, \frac{a}{b} \in \mathbb{Q}$，並且滿足 $\left|\alpha - \frac{a}{b}\right| < \frac{1}{2b^2}$
那麼 $\frac{a}{b}$ 會是 α 的 convergent of the continued fraction expansion

Wiener's Attack

- $ed = k\varphi(n) + 1$
- $d < \frac{1}{3}n^{\frac{1}{4}} \rightarrow \left|\frac{e}{n} - \frac{k}{d}\right| < \frac{1}{2d^2}$
- $\frac{k}{d}$ 會是 $\frac{e}{n}$ 的 convergents of the continued fraction expansion
- 遍歷所有 $\frac{e}{n}$ 的 convergents of the continued fraction expansion 其中一個會是 $\frac{k}{d}$

確認正確的 $\frac{k}{d}$

- $\varphi(n) = \frac{ed-1}{k}$
- $\varphi(n) = (p-1)(q-1) = n - p - \frac{n}{p} + 1$
- $p^2 + p(\varphi(n) - n - 1) + n = 0$
- 求解一元二次方程式可得 p，驗證 p 是否為 n 的因子即可

時間複雜度

- 計算 continued fraction expansion 時，其實是做輾轉相除法
- 解一元二次方程式只需要 $O(1)$
- Wiener Attack： $O(log(e))$

Proof - Wiener's Attack

Lemma 1

如果 $p \approx q \approx \sqrt{n}$

$$n - \varphi(n) < 3\sqrt{n} \tag{1}$$

Proof

$$\begin{aligned} n - \varphi(n) &= n - (p-1)(q-1) && (2) \\ &= n - pq + p + q - 1 && (3) \\ &= p + q - 1 && (4) \\ &< 3\sqrt{n} && (5) \end{aligned}$$

Proof - Wiener's Attack

Lemma 2

如果 $d < \frac{1}{3}n^{\frac{1}{4}}$

$$k < \frac{1}{3}n^{\frac{1}{4}} \tag{6}$$

Proof

$$k\varphi(n) = ed - 1 < ed < \varphi(n)d \tag{7}$$

$$k < d < \frac{1}{3}n^{\frac{1}{4}} \tag{8}$$

Proof - Wiener's Attack

Lemma 3

如果 $d < \frac{1}{3}n^{\frac{1}{4}}$

$$\frac{1}{2d} > \frac{1}{n^{\frac{1}{4}}} \tag{9}$$

Proof

$$d < \frac{1}{3}n^{\frac{1}{4}} \tag{10}$$

$$2d < 3d < n^{\frac{1}{4}} \tag{11}$$

$$\frac{1}{2d} > \frac{1}{n^{\frac{1}{4}}} \tag{12}$$

Proof - Wiener's Attack

如果 $d < \frac{1}{3}n^{\frac{1}{4}}$

$$\left|\frac{e}{n} - \frac{k}{d}\right| = \left|\frac{ed - nk}{nd}\right| \tag{13}$$

$$= \left|\frac{1 + k\varphi(n) - nk}{nd}\right| \tag{14}$$

$$= \frac{k(n - \varphi(n)) - 1}{nd} < \frac{3k\sqrt{n} - 1}{nd} < \frac{3k\sqrt{n}}{nd} \tag{15}$$

$$< \frac{1}{n^{\frac{1}{4}}d} < \frac{1}{2d^2} \tag{16}$$

Common Modulus Attack

- 相同 m, n 以及互質的 e_1, e_2 加密出兩個密文 c_1, c_2
- Bézout's lemma gives us $e_1 s_1 + e_2 s_2 = gcd(e_1, e_2) = 1$
- $c_1^{s_1} c_2^{s_2} \equiv m^{e_1 s_1} m^{e_2 s_2} = m^{e_1 s_1 + e_2 s_2} = m \pmod{n}$

LSB Oracle Attack

情境

給 server 密文 c，得到解密後明文的最後一個 bit 稱作 r

LSB Oracle Attack - 方法一

要解密的資料

解密 $2^e c$ 成 $2m$

Oracle

$$m \in [0, \frac{n}{2}) \rightarrow 2m \bmod n \bmod 2 = 2m \bmod 2 = 0$$

$$m \in [\frac{n}{2}, n) \rightarrow 2m \bmod n \bmod 2 = 2m - n \bmod 2 = 1$$

根據最後一個 bit 是 0 或 1 就可以知道 m 在 $\frac{n}{2}$ 之前或之後

LSB Oracle Attack - 方法一

要解密的資料

解密 $4^e c$ 成 $4m$

Oracle

如果$m \in [0, \frac{n}{2})$

$m \in [0, \frac{n}{4}) \to 4m \bmod n \bmod 2 = 4m \bmod 2 = 0$

$m \in [\frac{n}{4}, \frac{2n}{4}) \to 4m \bmod n \bmod 2 = 4m - n \bmod 2 = 1$

根據最後一個 bit 是 0 或 1 就可以知道 m 在 $\frac{n}{4}$ 之前或之後

LSB Oracle Attack - 方法一

要解密的資料

解密 $4^e c$ 成 $4m$

Oracle

如果$m \in [\frac{n}{2}, n)$

$$m \in [\frac{2n}{4}, \frac{3n}{4}) \rightarrow 4m \bmod n \bmod 2 = 4m - 2n \bmod 2 = 0$$

$$m \in [\frac{3n}{4}, n) \rightarrow 4m \bmod n \bmod 2 = 4m - 3n \bmod 2 = 1$$

根據最後一個 bit 是 0 或 1 就可以知道 m 在 $\frac{3n}{4}$ 之前或之後

LSB Oracle Attack - 方法二

定義

$$\forall\ i : 0 \le x_i \le 1$$

$$m = y_0 = \sum_{i=0}^{k-1} 2^i x_i$$

$$y_i = \sum_{j=i}^{k-1} 2^{j-i} x_j$$

LSB Oracle Attack - 方法二

要解密的資料

解密 c 成 m

Oracle

$$m \equiv x_0 + 2y_1 \pmod{n}$$
$$r \equiv m \bmod 2 \equiv x_0 \pmod{n}$$
$$x_0 \equiv r \pmod{n}$$

LSB Oracle Attack - 方法二

要解密的資料

解密 $(2^{-1})^e c$ 成 $2^{-1}m$

Oracle

$$2^{-1}m \equiv 2^{-1}x_0 + x_1 + 2y_2 \pmod{n}$$

$$r \equiv 2^{-1}m \text{ mod } 2 \equiv (2^{-1}x_0 + x_1) \text{ mod } 2 \pmod{n}$$

$$x_1 \equiv (r - 2^{-1}x_0) \text{ mod } 2 \pmod{n}$$

LSB Oracle Attack - CTF

- Google CTF QUALS 2018 - PERFECT-SECRECY
- TokyoWesterns CTF 4th 2018 - mixed-cipher
- HITCON CTF 2018 - Lost-Key

Bleichenbacher's Attack

- When server decrypt a message, it check whether first two bytes is 02
- It gives us an oracle to test whether the decrypted message has 02 as its first two bytes
- Using these information, we can efficiently decrypt any messages [4]

02	Random	00	M

Figure: PKCS #1

Coppersmith Method

- $f \in \mathbb{Z}[x]$ is a monic polynomial with degree d
- Coppersmith Method can efficiently find all roots $x_0 < n^{\frac{1}{d}-\epsilon}$ where $0 < \epsilon < \frac{1}{d}$
- 將 RSA 解密轉換成多項式 $f(x) = x^e - c$ 求根

Franklin-Reiter Related Message Attack

- 兩個明文滿足 $m_1 = f(m_2), f \in \mathbb{Z}_n[x]$
- $g_1(x) = f(x)^e - c_1 \in \mathbb{Z}_n[x]$
- $g_2(x) = x^e - c_2 \in \mathbb{Z}_n[x]$
- $x - m_2$ 會是 g_1, g_2 的公因數
- 對 g_1, g_2 做輾轉相除法可得 $x - m_2$

Franklin-Reiter Related Message Attack - CTF

- HITCON CTF QUALS 2014 - rsaha
- N1CTF 2018 - rsa_padding

ROCA (Return of Coppersmith's Attack)

- The Return of Coppersmith's A ttack: Practical Factorization of Widely Used RSA Moduli[5]
- CVE-2017-15361

Insecure Key Generation

- All primes p, q have the following form
- $p = kM + (65537^a \mod M)$
- $M = \prod_{i=0}^{n} P_i = 2 \times 3 \times 5 \times \cdots$ (n successive primes)
- known M, unknown k, a

Coppersmith Method

- Using coppersmith method to factor with high bits known only require $\frac{1}{4}\log_2 N$ bits of p

N (bits)	n	M (bits)
512	39	219.19
1024	71	474.92
2048	126	970.96
4096	225	1962.19

Table: Show that the generated modulus has low entropy

Factorization

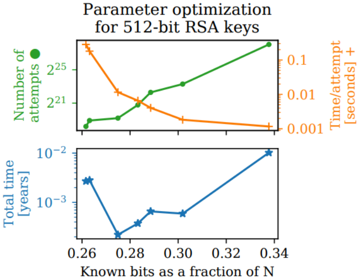
- $p = kM + (65537^a \mod M)$
- brute force a
- use coppersmith method to solve k

Complexity

- x $=$ order of 65537 in $\mathbb{Z}_M^*$
- y $=$ coppersmith method complexity
- complexity $= x \times y$

Optimization

- We can reduce x, but with less information, y will increase



Rabin 簡介

- Rabin 是 RSA 在 e = 2 的特例
- e = 2 的時候，$gcd(e, \varphi(n)) = 2$，沒辦法求 modular inverse
- 在 p 跟 q 下分別求模開方根再用中國剩餘定理組起來

Rabin 加解密

```python
def enc(m, public):
    n = public
    return pow(m, 2, n)

def dec(c, private):
    p, q = private
    mp = modular_sqrt(c, p)
    mq = modular_sqrt(c, q)
    return [crt([mp, mq], [p, q]),
            crt([-mp, mq], [p, q]),
            crt([mp, -mq], [p, q]),
            crt([-mp, -mq], [p, q])]
```

模開方根

- 在合數下求模開方根跟分解合數一樣困難，在質數下求模開方根可以用 Tonelli-Shanks algorithm [6]
- m 是 c 在質數 p 下模開方根，-m 也會是
- 在質數 $p \equiv 3 \pmod 4$ 有特殊解，$\sqrt{c} \equiv c^{\frac{1}{4}(p+1)} \pmod p$

Rabin - CTF

- HITCON CTF Quals 2015 - Rsabin

ElGamal

- 基本上就是用 Diffie-Hellman Key Exchange 交換 α^{ab}
- 然後加密就用 $alpha^{ab}$ 做乘法，解密就用 α^{ab} 的 inverse 做乘法
- ElGamal 的安全性和 Diffie Hellman Key Exchange 一樣都是基於 Discrete Logarithm Problem
- 更一般性的 ElGamal 就是將 $\mathbb{Z}_p^*$ 換成任意的 Finite Cyclic Group G，例如 Elliptic Curve

ElGamal 產生公私鑰

- Generate a large prime p, and a generator α of $\mathbb{Z}_p^*$
- Select a random integer $a : 1 \leq a < p - 1$
- Public Key is (p, α, α^a), Private Key is a

ElGamal 加密

- Select a random integer $b : 1 \leq b < p - 1$
- Ciphertext $c = (\alpha^b, m\alpha^{ab})$

- R. L. Rivest and R. D. Silverman, "Arestrong'primes needed for rsa?," in *The 1997 RSA Laboratories Seminar Series, Seminars Proceedings*, 1997.
- J. M. Pollard, "Theorems on factorization and primality testing," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 76, pp. 521–528, Cambridge University Press, 1974.
- H. C. Williams, "A p + 1 method of factoring," *Mathematics of Computation*, vol. 39, no. 159, pp. 225–234, 1982.
- D. Bleichenbacher, "Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs# 1," in *Annual International Cryptology Conference*, pp. 1–12, Springer, 1998.
- M. Nemec, M. Sýs, P. Svenda, D. Klinec, and V. Matyas, "The return of coppersmith's attack: Practical factorization of

widely used rsa moduli.," in *ACM Conference on Computer and Communications Security* (B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, eds.), pp. 1631–1648, ACM, 2017.

"Computing modular square roots in python."