

# Cryptography 0x02 - 在家輕鬆學 RSA

oalieno

2020/06/04

# Table of Contents

## 1 RSA Public Key Cryptosystem

- What happens if you pick wrong primes  $p, q$
- What happens if you pick wrong  $e$
- What happens if you pick wrong  $d$
- What happens if you reuse parameters
- Chosen Ciphertext Attack
- Coppersmith Method
- Related Message Attack

## 2 RSA Digital Signature

- Signature Forgery

# RSA

# RSA 產生金鑰

- 選兩個質數  $p, q$  和一個整數  $e$
- 計算
  - $n := pq$
  - $\varphi(n) = (p-1)(q-1)$
  - $d := e^{-1} \bmod \varphi(n)$
- 公鑰是  $(n, e)$  私鑰是  $(n, d)$

## 碎碎念

- $n, e$  要滿足  $\gcd(e, \varphi(n)) = 1$ ，沒有滿足就重選
- 不然  $e$  在模  $\varphi(n)$  下沒有模反元素，無法解密

# RSA 加解密

- 明文  $m$  密文  $c$
- 加密  $c = m^e \bmod n$
- 解密  $m = c^d \bmod n$

# 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

推論一下

$$\begin{aligned}d &\equiv e^{-1} \pmod{\varphi(n)} \\ \Rightarrow ed &\equiv 1 \pmod{\varphi(n)} \\ \Rightarrow ed &= k\varphi(n) + 1 \text{ for some } k \\ &= k(p-1)(q-1) + 1\end{aligned}$$

# RSA 正確性

驗證  $m^{ed} \equiv m \pmod{p}$

if  $\gcd(m, p) = 1$

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

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 正確性

## 組起來

- 另  $x = m^{ed}$ ，現在已知

$$x \equiv m \pmod{p}$$

$$x \equiv m \pmod{q}$$

- 想求  $x$  模  $n$  會是多少
- 中國剩餘定理告訴我們在模  $n = pq$  下存在唯一解
- $x$  模  $n$  等於  $m$  是一個解，那就只能是他了
- 所以  $m^{ed} \equiv m \pmod{n}$ ，得證

# Relation to integer factorization

factor  $n \rightarrow$  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

obtain private key  $\rightarrow$  factor  $n$

- 如果我們有一個很有效率的演算法  $f$  能找到模  $n$  下的開方根，那我們就能分解  $n$

# Relation to integer factorization

## 利用 $f$ 分解 $n$

- 選一個  $x$ ，計算  $y \equiv x^2 \pmod{n}$
- 利用那個演算法  $f$  找出  $y$  在模  $n$  下的開方根  $z$
- $y$  的模開方根會有四個解，有  $\frac{1}{2}$  的機率  $z \not\equiv \pm x$
- 如果  $z \not\equiv \pm x$ 
  - $z^2 \equiv x^2 \pmod{n} \Rightarrow (z+x)(z-x) \equiv 0 \pmod{n}$
  - $1 < \gcd(n, z+x) < n$  或  $1 < \gcd(n, z-x) < n$  會成立
  - 那就成功分解  $n$
- 如果  $z \equiv \pm x$ ，就再選一次  $x$

# Relation to integer factorization

## 有效率的演算法 $f$ 找模 $n$ 下得開方根

- 選一個  $g$ ，計算  $g^{ed-1} \equiv 1 \pmod{n}$
- $ed - 1 = k\varphi(n) = 2^t r$
- 這樣  $g^{2^t r} \equiv (g^{2^{t-1} r})^2 \pmod{n}$  就會是一組模開方根
- $g^{2^t r} \equiv 1 \equiv (\pm 1)^2 \pmod{n}$ ， $\pm 1$  已經是一組模開方根的解
- 如果  $g^{2^{t-1} r} \not\equiv \pm 1$ ，就可以用剛剛講的方法分解  $n$
- $g^{2^t r}, g^{2^{t-1} r}, g^{2^{t-2} r}, \dots, g^r$  都找不到就再選一次  $g$

# Factoring Tools

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

# 同態 ( Homomorphic )

## 同態的定義

- $f(x * y) = f(x) * f(y)$
- $*$  可以是任意的一種運算元

## RSA 的乘法

- RSA 的乘法有 homomorphic 的特性
- $E(m_1)E(m_2) = m_1^e m_2^e \bmod n = (m_1 m_2)^e \bmod n = E(m_1 m_2)$
- Leads to chosen ciphertext attack

# RSA

What happens if you pick wrong primes  $p, q$



# How to pick large primes $p, q$

- $|p - q|$  太小  $\rightarrow$  fermat factorization
- $p - 1$  的最大質因數很小  $\rightarrow$  Pollard's  $p - 1$  Algorithm
- $p + 1$  的最大質因數很小  $\rightarrow$  Williams's  $p + 1$  Algorithm
- $r - 1$  的最大質因數很小  $\rightarrow$  Cycling Attack
  - $r$  是  $p - 1$  的最大質因數

# Strong Primes

- 能抵檔針對質因數小的攻擊的質數我們叫他 Strong Primes
- 那建議  $p, q$  一定要選 Strong Primes ... 嗎?
- 其實隨機產生不會比 Strong Primes 還差 [1]

## Strong Primes

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

# Pollard's $p - 1$ Algorithm

- $p - 1$  是一個  $B$ -smooth 的數，也就是他最大的質因數是  $B$
- 如果  $B$  很小，就可以有效的分解  $n$

$$\begin{aligned}p - 1 &| 1 \times 2 \times \cdots B \\ \Rightarrow 2^{1 \times 2 \times \cdots B} &= 2^{k(p-1)} \equiv 1 \pmod{p} \\ \Rightarrow \gcd(2^{1 \times 2 \times \cdots B} - 1, n) &> 1\end{aligned}$$

# Pollard's $p - 1$ Algorithm

```
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
```

What happens if you pick wrong e

What happens if you pick wrong e

# 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  $2^x + 1$  這種形式的質數，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

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

# Direct eth Root

- 如果  $m, e$  都很小，使得  $m < n^{\frac{1}{e}} \Rightarrow m^e < n$
- 直接在整數下取 eth root 就可以還原  $m$
- 所以我們需要做 random padding



# Broadcast Attack

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

$$\begin{cases} m^3 \equiv c_1 \pmod{n_1} \\ m^3 \equiv c_2 \pmod{n_2} \\ m^3 \equiv c_3 \pmod{n_3} \end{cases}$$

Use CRT to get  $c$ ,  $m^3 \equiv c \pmod{n_1 n_2 n_3}$

$m^3 < n_1 n_2 n_3 \rightarrow m^3 = c \rightarrow$  direct eth root

# RSA

What happens if you pick wrong  $d$

- 基本上  $d$  也不是我們選的，是從  $e$  算出來的
- $d$  太小  $\rightarrow$  Wiener's attack, Boneh-Durfee's attack,  $\dots$

# 歷年來攻擊小 d 的演進

Bound for d	Assumed Interval for $\gamma$	Year	Citation
$d < \frac{1}{3} N^{\frac{1}{4}}$	No $\gamma$	1990	[4]
$d < \frac{1}{8} N^{\frac{3}{4}-\gamma}$	$0.25 \leq \gamma < 0.5$	2002	[5]
$d < N^{\frac{1-\gamma}{2}}$	$0.25 \leq \gamma < 0.5$	2008	[6]
$d < N^{\frac{3}{4}-\gamma}$	$0.25 \leq \gamma < 0.5$	2009	[7]
$d < \frac{\sqrt{6\sqrt{2}}}{6} N^{\frac{1}{4}}$	No $\gamma$	2013	[8]
$d < \frac{1}{2} N^{\frac{1}{4}}$	No $\gamma$	2015	[9]
$d < \frac{\sqrt{3}}{\sqrt{2}} N^{\frac{3}{4}-\gamma}$	$0.25 \leq \gamma < 0.5$	2019	[10]

**Table:** Comparison of the bounds on d for RSA modulus  $N = pq$  [10]

# Wiener Attack

## Wiener Attack

- $ed = k\varphi(n) + 1$

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

$$\Rightarrow \left| \frac{e}{n} - \frac{k}{d} \right| < \frac{1}{2d^2}$$

$\Rightarrow \frac{k}{d}$  會在  $\frac{e}{n}$  的收斂連分數裡面

- 遍歷所有  $\frac{e}{n}$  的收斂連分數，其中一個會是  $\frac{k}{d}$

# Wiener Attack

## Wiener Attack (cont.)

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

# Wiener Attack - 什麼是收斂連分數

- $\frac{13}{17} = 0 + \frac{1}{1 + \frac{1}{3 + \frac{1}{4}}}$

- $\frac{13}{17}$  的收斂連分數是  $[c_0, c_1, c_2, c_3]$

$$c_0 = 0 = \frac{0}{1}$$

$$c_1 = 0 + \frac{1}{1} = \frac{1}{1}$$

$$c_2 = 0 + \frac{1}{1 + \frac{1}{3}} = \frac{3}{4}$$

$$c_3 = 0 + \frac{1}{1 + \frac{1}{3 + \frac{1}{4}}} = \frac{13}{17}$$

# Wiener Attack - 證明

假設  $d < \frac{1}{3}n^{\frac{1}{4}}$

$$\begin{aligned}\left| \frac{e}{n} - \frac{k}{d} \right| &= \left| \frac{ed - nk}{nd} \right| \\&= \left| \frac{1 + k\varphi(n) - nk}{nd} \right| \\&= \frac{k(n - \varphi(n)) - 1}{nd} < \frac{3k\sqrt{n} - 1}{nd} < \frac{3k\sqrt{n}}{nd} \\&< \frac{1}{n^{\frac{1}{4}}d} < \frac{1}{2d^2}\end{aligned}$$



# Wiener Attack - 證明

## 稍微解釋一下中間的代換

- 假設  $p \approx q \approx \sqrt{n}$

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

# Wiener Attack - 證明

稍微解釋一下中間的代換 (cont.)

$$k\varphi(n) = ed - 1 < ed < \varphi(n)d$$
$$\Rightarrow k < d < \frac{1}{3}n^{\frac{1}{4}}$$

# Wiener Attack - 證明

稍微解釋一下中間的代換 (cont.)

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

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

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

# Wiener Attack - 證明

## Legendre's theorem in Diophantine approximations

給定  $\alpha \in \mathbb{R}$ ,  $\frac{a}{b} \in \mathbb{Q}$ , 並且滿足  $|\alpha - \frac{a}{b}| < \frac{1}{2b^2}$   
那麼  $\frac{a}{b}$  會是  $\alpha$  的收斂連分數

## 根據這個定理

- 剛剛已經推得  $|\frac{e}{n} - \frac{k}{d}| < \frac{1}{2d^2}$
- 對應上述定理  $\alpha = \frac{e}{n}$ ,  $a = k$ ,  $b = d$
- 所以  $\frac{k}{d}$  會是  $\frac{e}{n}$  的收斂連分數，得證

## Wiener Attack - 時間複雜度

- 計算收斂連分數時，是在做輾轉相除法，需要  $O(\log(e))$
- 解一元二次方程式只需要  $O(1)$
- 時間複雜度:  $O(\log(e))$

# RSA

## What happens if you reuse parameters

# Common Factor Attack

## 情境

- 當兩把公鑰的 modulus  $n_1, n_2$  有共同的質因數

## 解法

- 使用  $\gcd(n_1, n_2)$  求出共同質因數

## Batch GCD

- 當你有很多公鑰想找裡面有沒有共同的質因數
- 參考這篇 [11]，他們有實作工具 <https://factorable.net/>

# Common Modulus Attack

## 情境

- 已知兩把公鑰  $(n, e_1)$ ,  $(n, e_2)$  有相同的 modulus  $n$
- $e_1, e_2$  滿足  $\gcd(e_1, e_2) = 1$
- 用這兩把公鑰加密明文  $m$  為  $c_1, c_2$
- 可以在不分解  $n$  的情況下找出  $m$

## 解法

- 根據貝祖定理  $e_1 s_1 + e_2 s_2 = \gcd(e_1, e_2) = 1$  有整數解  $s_1, s_2$
- 整數解  $s_1, s_2$  可由擴展歐幾里德演算法求出
- 計算  $c_1^{s_1} c_2^{s_2} \equiv m^{e_1 s_1 + e_2 s_2} \equiv m \pmod{n}$





# Chosen Ciphertext Attack

## 情境

- 有一個 oracle 給他密文可以得到明文
- 但是唯獨不能解某個特定的密文  $c$

## 解法

- 使用 oracle 解  $2^e c$  得  $2m$
- $2^{-1} \cdot 2m \equiv m \pmod{n}$

- Least Significant Bit Oracle Attack
- 有一個 oracle 給他密文可以得到明文的最低位那個 bit
- 類似 Chosen Ciphertext Attack 但是只能拿 1 bit

# LSB Oracle Attack - 解法一

Oracle

$$2^e c \xrightarrow{\text{oracle}} 2m$$

推論

- $$\begin{cases} \lfloor \lfloor 2m \rfloor_n \rfloor_2 = \lfloor 2m \rfloor_2 = 0, & \text{if } m \in [0, \frac{n}{2}) \\ \lfloor \lfloor 2m \rfloor_n \rfloor_2 = \lfloor (2m - n) \rfloor_2 = 1, & \text{if } m \in [\frac{n}{2}, n) \end{cases}$$
- 根據最低位那個 bit 是 0 或 1 推論  $m$  在  $\frac{n}{2}$  之前或之後

# LSB Oracle Attack - 解法一

Oracle

$$4^e c \xrightarrow{\text{oracle}} 4m$$

推論

- 如果  $m \in [0, \frac{n}{2})$
- $$\begin{cases} \lfloor \lfloor 4m \rfloor_n \rfloor_2 = \lfloor 4m \rfloor_2 = 0, & \text{if } m \in [0, \frac{n}{4}) \\ \lfloor \lfloor 4m \rfloor_n \rfloor_2 = \lfloor (4m - n) \rfloor_2 = 1, & \text{if } m \in [\frac{n}{4}, \frac{2n}{4}) \end{cases}$$
- 根據最低位那個 bit 是 0 或 1 推論  $m$  在  $\frac{n}{4}$  之前或之後
- 如果  $m \in [\frac{n}{2}, n)$ ，同理

- 每次可以縮小一半的範圍，需要  $O(\log(n))$  次 oracle
- 有點像是在二分搜尋

# LSB Oracle Attack - 解法二

Oracle

$$c \xrightarrow{\text{oracle}} m$$

推論

$y_1$

$x_0$

$$m = x_0 + 2y_1$$

$$r \equiv \lfloor x_0 + 2y_1 \rfloor_n \pmod{2}$$

$$\equiv x_0 \pmod{2}$$

$$\Rightarrow x_0 \equiv r \pmod{2}$$

# LSB Oracle Attack - 解法二

Oracle

$$(2^{-1})^e c \xrightarrow{\text{oracle}} 2^{-1} m$$

推論

$y_2$	$x_1$	$x_0$
-------	-------	-------

$$2^{-1} m = 2^{-1} x_0 + x_1 + 2y_2$$

$$r \equiv \lfloor 2^{-1} x_0 + x_1 + 2y_2 \rfloor_n \pmod{2}$$

$$\equiv \lfloor 2^{-1} x_0 \rfloor_n + x_1 \pmod{2}$$

$$\Rightarrow x_1 \equiv r - \lfloor 2^{-1} x_0 \rfloor_n \pmod{2}$$



# LSB Oracle Attack - 解法二

Oracle

$$(2^{-2})^e c \xrightarrow{\text{oracle}} 2^{-2} m$$

推論

$y_3$	$x_2$	$x_1$	$x_0$
-------	-------	-------	-------

$$2^{-2} m = 2^{-2} x_0 + 2^{-1} x_1 + x_2 + 2y_3$$

$$r \equiv \lfloor 2^{-2} x_0 + 2^{-1} x_1 + x_2 + 2y_3 \rfloor_n \pmod{2}$$

$$\equiv \lfloor 2^{-2} x_0 + 2^{-1} x_1 \rfloor_n + x_2 \pmod{2}$$

$$\Rightarrow x_2 \equiv r - \lfloor 2^{-2} x_0 + 2^{-1} x_1 \rfloor_n \pmod{2}$$

# LSB Oracle Attack - 解法二

## 碎碎念

- $x_i$  代表  $m$  的第  $i$  個 bit
- $y_i$  代表  $m$  整段從最高位的 bit 到最低位數來第  $i$  個 bit
- 每次可以推論一個 bit，需要  $O(\log(n))$  次 oracle

# LSB Oracle Attack - CTF

## CTF 考古題

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

## Bleichenbacher 1998 (BB98)

- 介紹一個 Real World 的例子 Bleichenbacher 1998 年的論文
- 也是一種 Chosen Ciphertext Attack，這次的 oracle 是給他密文可以得到明文的最高位那個 byte 是不是 0x0002
- 這個情境很像 Padding Oracle Attack，會告訴你給他的格式正不正確
- 這個格式是 PKCS#1 v1.5



Figure: PKCS #1 v1.5

# Bleichenbacher 1998

## Oracle

$s^e_C \xrightarrow{\text{oracle}} sm$  格式對不對

## 推論

00

02

...

 $\leq sm \% n <$ 

00

03

...

- 他檢查格式的方法是看前面兩個 bytes 是不是 0002
- 所以如果他符合格式的話，代表  $2B \leq sm \% n < 3B$

# Bleichenbacher 1998

## 推論 (cont.)

- $sm = kn + (sm \% n)$  for some unknown  $k$

$$2B \leq sm \% n < 3B$$

$$\Rightarrow 2B \leq sm - kn < 3B$$

$$\Rightarrow \frac{2B + kn}{s} \leq m < \frac{3B + kn}{s}$$

- 雖然我們不知道  $k$ ，但是我們可以考慮所有可能的  $k$

# Bleichenbacher 1998

## 推論 (cont.)

- 要考慮所有可能的  $k$ ，那就要看一下  $k$  的範圍

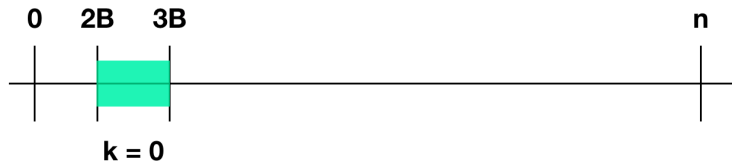
$$2B \leq sm - kn < 3B$$
$$\Rightarrow \frac{sm - 3B}{n} < k \leq \frac{sm - 2B}{n}$$

- 我們知道原本的明文會滿足  $0 \leq m < n$
- 所以就可以用舊的  $m$  的範圍推論  $k$  的範圍
- 再用  $k$  的範圍推論新的  $m$  的範圍

# Bleichenbacher 1998

假設  $s = 1$  格式正確

- $k$  只可能是 0
- 新的  $m$  的範圍  $2B \leq m < 3B$

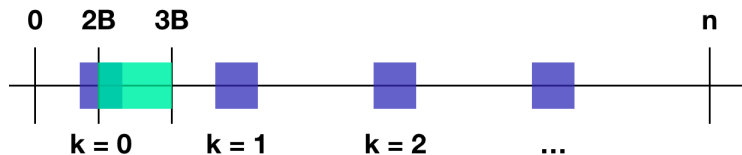




# Bleichenbacher 1998

假設  $s = 10$  格式正確

- 綠色是  $m$  舊的範圍，藍色這次 oracle 得到的新範圍
- 兩個交集的地方就是  $m$  新的可能的範圍



- 每次 oracle 會把這些範圍和舊的範圍交集起來
- $m$  的範圍就會越來越小，直到只剩一個

# Bleichenbacher 1998

## 後續影響

- 2016 : DROWN: Breaking TLS Using SSLv2 [12]
- 2018 : Return Of Bleichenbacher Oracle Threat (ROBOT) [13]
- 2019 : The 9 Lives of Bleichenbacher's CAT [14]
- 到近幾年還是會看到他的身影

## Coppersmith Method [15]

# Coppersmith Method - 問題敘述

## Given

- A monic polynomial  $f(x) = x^\delta + \dots$
- An integer  $N$  of unknown factorization
- An upper bound  $X$

## Goal

- Find all  $|x_0| \leq X$  satisfy  $f(x_0) \equiv 0 \pmod{b}$
- Where  $b$  is a divisor of  $N$ , and  $b \geq N^\beta$

## 碎碎念

- 這個演算法就是 find small modular root

# Coppersmith Method - 原理

## 換個問題解

- Let  $X_0 = \{x_0 \mid f(x_0) \equiv 0 \pmod{b}, |x_0| \leq X\}$
- Find a function  $g$  such that  $\forall x_0 \in X_0, g(x_0) = 0$

## 碎碎念

- 計算  $\text{mod } b$  下的根很難，就算根的範圍縮小了，還是不知道要怎麼做啊
- 如果我們能找到一個  $g(x)$ ，他在整數底下求出來的根和  $f(x)$  在  $\text{mod } b$  下求出來的根一樣就好，在整數底下求根我會做

# Coppersmith Method - 原理

## $g(x)$ 從哪裡來

- Pick two integers  $m, t$
- Construct a collection of polynomial from  $f(x)$

$$f_i(x) = \begin{cases} g_{i,j}(x) = x^j N^{m-i} f(x) & \text{for } 0 \leq i < m, 0 \leq j < \delta \\ h_i(x) = x^i f^m(x) & \text{for } 0 \leq i < t \end{cases}$$

- Satisfy  $\forall x_0 \in X_0, g_{i,j}(x_0) \equiv h_i(x_0) \equiv 0 \pmod{b^m}$

## 碎碎念

- 生成一堆滿足  $\text{mod } b^m = 0$  的變種  $f$

# Coppersmith Method - 原理

## 嘗試製造 $g(x)$

- Construct an integer linear combination  $g(x)$

$$g(x) = \sum_{i=0}^{n-1} a_i f_i(x), \text{ for } a_i \in \mathbb{Z}$$

Where  $n = m\delta + t$

## 碎碎念

- 把變種  $f$  看成 coefficient vector，考慮變種  $f$  生成的 lattice



# Coppersmith Method - 原理

如果  $|g(x_0)| < b^m$  的話

- 我們構造的  $f_i(x)$  都滿足  $f_i(x_0) \equiv 0 \pmod{b^m}$ ， $g(x)$  也滿足
- $\forall x_0 \in X_0$

$$\begin{cases} g(x_0) \equiv 0 \pmod{b^m} \\ |g(x_0)| < b^m \end{cases} \Rightarrow g(x_0) = 0$$

- 這樣我們找  $g(x)$  上所有根，裡面就會包含我們要的答案  $X_0$

# Coppersmith Method - 原理

## 再換個問題解

$$\blacksquare \forall |x_0| < X, \|g(xX)\| < \frac{b^m}{\sqrt{n}} \Rightarrow |g(x_0)| < b^m$$

## 碎碎念

- 這個小定理幫我們把問題轉成在 lattice 找 short vector
- 覺得很熟悉嗎，登愣，Shortest Vector Problem，那我們就用 LLL 演算法來做搞定了
- 這裡的  $\|g(xX)\|$  是指  $g(xX)$  的 coefficient vector 的 norm

# Coppersmith Method - 原理

## Proof of the previous statement

- Let  $c_i$  be the coefficients of  $g(x)$ , then  $\forall |x_0| < X$

$$\begin{aligned}|g(x_0)| &= \sum_{i=0}^n c_i x_0^i \leq \sum_{i=0}^n |c_i x_0^i| \\ &\leq \sum_{i=0}^n |c_i X^i| = \sqrt{(\sum_{i=0}^n |c_i X^i|)^2} \leq \sqrt{n \sum_{i=0}^n |c_i X^i|^2} \\ &= \sqrt{n} \|g(xX)\| < b^m\end{aligned}$$

## 碎碎念

- 中間用到了柯西不等式，忘記的同學們可以回去複習一下

# Coppersmith Method - 原理

## LLL 找到的 short vector 夠嗎？

- Let  $L$  be the coefficient matrix formed by  $f_0(x), f_1(x), \dots$ , and  $v$  be the vector found using LLL
- 現在要證明  $v$  滿足  $\|g(xX)\| < \frac{b^m}{\sqrt{n}}$
- 已知  $v$  會滿足  $\|v\| \leq 2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}}$
- 所以我們只要讓  $2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}} < \frac{b^m}{\sqrt{n}} < \frac{N^{\beta m}}{\sqrt{n}}$  就可以了

# Coppersmith Method - 原理

## Compute $\det(L)$

- Notice that the degree of  $f_i(x)$  is  $i$ , therefore the basis of  $L$  forms a lower triangular matrix
- So the  $\det(L)$  is simply the product of all entries on the diagonal

$$\det(L) = N^{\frac{1}{2}\delta m(m+1)} X^{\frac{1}{2}n(n-1)}$$

# Coppersmith Method - 原理

## 決定 $m, t, X$

- 我們其實還沒決定  $m, t, X$  要是多少
- 基本上我們想要  $m, t$  越小越好，這樣 lattice 會小一點
- $X$  則是越大越好，這樣我們能解的東西越多
- 而且最好可以消掉一些變數，比較好化簡
- $m = \left\lceil \frac{\beta^2}{\delta \varepsilon} \right\rceil, t = \left\lfloor \delta m \left( \frac{1}{\beta} - 1 \right) \right\rfloor, X = \left\lceil \frac{1}{2} N^{\frac{\beta^2}{\delta} - \varepsilon} \right\rceil$
- 這裡引入了一個新的常數  $\varepsilon$ ，滿足  $0 < \varepsilon \leq \frac{1}{7}\beta$ ，可以調參

# Coppersmith Method - 原理

## 最後的最後

- 最後就剩驗證  $2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}} < \frac{N^{\beta m}}{\sqrt{n}}$
- 把  $m, t, X, \det(L)$  代進去喇一喇就行了

# Coppersmith Method - 總結

## Given

- A monic polynomial  $f(x) = x^\delta + \dots$
- An integer  $N$  of unknown factorization
- A rational number  $\beta$

## Coppersmith Method

- Find all  $|x_0| \leq \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$  satisfy  $f(x_0) \equiv 0 \pmod{b}$
- Where  $b$  is a divisor of  $N$ , and  $b \geq N^\beta$



# Coppersmith Method - 總結

## Push the bound

- 我們可以很輕易的把  $X$  推廣到  $cN^{\frac{\beta^2}{\delta}}$  for some constant  $c$
- 選  $\varepsilon = \frac{1}{\log N}$  那  $X = \frac{1}{4} N^{\frac{\beta^2}{\delta}}$
- 接著把  $[-cN^{\frac{\beta^2}{\delta}}, cN^{\frac{\beta^2}{\delta}}]$  切成  $4c$  個區間
- 假設  $x_i$  是每個區間的中點，那把所有  $f(x - x_i)$  的根收集起來就是答案了
- 可以平行去跑所有區間

# Coppersmith Method - 總結

## Given

- A monic polynomial  $f(x) = x^\delta + \dots$
- An integer  $N$  of unknown factorization
- A rational number  $\beta$

## Coppersmith Method

- Find all  $|x_0| \leq cN^{\frac{\beta^2}{\delta}}$  satisfy  $f(x_0) \equiv 0 \pmod{b}$
- Where  $b$  is a divisor of  $N$ , and  $b \geq N^\beta$

# Coppersmith Method - 演算法步驟

## Step 1 - 選參數

- 選  $m = \left\lceil \frac{\beta^2}{\delta \epsilon} \right\rceil$ ,  $t = \left\lfloor \delta m \left( \frac{1}{\beta} - 1 \right) \right\rfloor$ , 計算  $X = \left\lceil N^{\frac{\beta^2}{\delta} - \epsilon} \right\rceil$

## Step 2 - 計算變種 f

- 計算

$$g_{i,j}(x) = x^j N^{m-i} f^j(x) \text{ for } 0 \leq i < m, 0 \leq j < \delta$$

$$h_i(x) = x^i f^m(x) \quad \text{for } 0 \leq i < t$$

# Coppersmith Method - 演算法步驟

## Step 3 - LLL

- $g_{i,j}(xX), h_i(xX)$  的 coefficient vectors 組成的 lattice basis  $B$
- 對  $B$  做 LLL algorithm

## Step 4 - 還原 $g$

- 假設  $v$  是 shortest vector in LLL reduced basis
- 那  $v$  是某個  $g(xX)$  的 coefficient vector, 從  $v$  還原  $g(x)$

# Coppersmith Method - 演算法步驟

## Step 5 - 找根

- 找出  $g(x)$  的所有根
- 檢查根  $x_0$  是否滿足  $\gcd(N, f(x_0)) \geq N^\beta$ ，沒有滿足的丟掉

# Stereotyped messages

- 已知大部分的訊息，比如某個固定格式
- 明文  $m = \tilde{m} + x_0$ ，已知  $\tilde{m}$ ，未知  $x_0$  滿足  $x \leq N^{\frac{1}{e}}$
- 密文  $c \equiv m^e \equiv (\tilde{m} + x_0)^e \pmod{N}$
- $x_0$  會是  $f(x) \equiv (\tilde{m} + x)^e - c \pmod{N}$  的小根

## Coppersmith Method

- `small_roots` 參數選擇
  - $\beta = 1, \delta = e$
  - $\epsilon$  調小，更大的  $X$ ，花更多時間（預設  $\epsilon = \frac{\beta}{8}$ ）
  - 能找到的範圍  $X < \frac{1}{2} n^{\frac{1}{e} - \epsilon}$

# Known High Bits of $p$

- $N = pq$ , 質數  $p = \tilde{p} + x_0$ , 已知  $\tilde{p}$ , 未知  $x_0$  滿足  $|x_0| < N^{\frac{1}{4}}$
- $x_0$  會是  $f(x) = \tilde{p} + x \pmod{p}$  的小根

## Coppersmith Method

- `small_roots` 參數選擇
  - $\beta = \frac{1}{2}, \delta = 1$
  - 洩漏高位的  $p$  不一定大於  $n^{\frac{1}{2}}$ ,  $\beta$  可調小一點, 但  $X$  會縮小
  - $\epsilon$  調小, 更大的  $X$ , 花更多時間 (預設  $\epsilon = \frac{\beta}{8}$ )
  - 能找到的範圍  $X < \frac{1}{2}n^{\frac{1}{4}-\epsilon}$

# Real World

## Real World Example

- 2013 : Factoring RSA Keys from Certified Smart Cards [16]
- 2017 : The Return of Coppersmith's Attack (ROCA) [17]



## Related Message Attack

# Franklin-Reiter Related Message Attack

## 情境

- 已知公鑰  $n, e$ ，加密兩個明文  $m_1, m_2$  為密文  $c_1, c_2$
- $m_1, m_2$  滿足  $m_2 = f(m_1)$  對某個多項式  $f$

# Franklin-Reiter Related Message Attack

## 解法

- 考慮下面兩個多項式

$$g_1(x, y) \equiv x^e - c_1 \pmod{n}$$

$$g_2(x, y) \equiv f(x)^e - c_2 \pmod{n}$$

- $m_2$  同時是  $g_1, g_2$  的根，也就是  $(x - m_1)$  可以整除  $g_1, g_2$
- $\gcd(g_1, g_2) = x - m_1$

# Coppersmith Short-Pad Attack

## 情境

- 已知公鑰  $n, e$ ，加密兩個明文  $m_1, m_2$  為密文  $c_1, c_2$
- $m_1 = 2^k M + r_1, m_2 = 2^k M + r_2$
- $r_1, r_2$  是 padding， $M$  是真正的明文

# Coppersmith Short-Pad Attack

## 解法

- 考慮下面兩個雙變量多項式

$$g_1(x, y) \equiv x^e - c_1 \pmod{n}$$

$$g_2(x, y) \equiv (x + y)^e - c_2 \pmod{n}$$

- 設  $\bar{y} = r_2 - r_1$
- $g_1(x, \bar{y}), g_2(x, \bar{y})$  有公因式  $x - m_1 \Leftrightarrow \text{res}(g_1(x, \bar{y}), g_2(x, \bar{y})) = 0$
- 計算  $g_1, g_2$  對  $x$  的 resultant  $h(y) = \text{res}_x(g_1, g_2)$
- 使用 Coppersmith 求  $h(y)$  的小根,  $\bar{y}$  會是其中一個答案
- $y = \bar{y}$  代回式子, 簡化為 Franklin-Reiter Related Message

# Coppersmith Short-Pad Attack

## Resultant

- Sylvester matrix 是由兩個單變量的多項式算出來的
- 兩個單變量多項式的 Resultant 就是他們的 Sylvester matrix 的 Determinant，而這個 Resultant 等於 0 若且唯若他們有 common root

# Coppersmith Short-Pad Attack

## Coppersmith Method

- $\text{res}_x(g1, g2)$  會是  $e^2$  次多項式
- `small_roots` 參數選擇
  - $\beta = 1, \delta = e^2$
  - $\epsilon$  調小，更大的  $X$ ，花更多時間 ( 預設  $\epsilon = \frac{\beta}{8}$  )
  - 能找到的範圍  $X < \frac{1}{2} n^{\frac{1}{2} - \epsilon}$

# RSA



# How RSA Digital Signature works

- RSA 也可以用來做數位簽章
- 明文  $m$
- 簽章  $s = m^d \bmod n$
- 驗證  $v = s^e \bmod n$  是否等於  $m$
- 因為  $m < n$ ，所以實務上會先做 hash 然後 padding






◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻ 90/92





# Random Signature Forgery

- RSA 乘法的同態性質讓我們能夠偽造隨機的簽章
- 給  $(m, s)$  可以偽造出  $(m^k, s^k)$  for some  $k$
- $s^k = (m^d)^k = (m^k)^d$  所以簽章是合法的
- 但沒辦法控制偽造出什麼明文
- 只對 textbook RSA 有用，有 hash 有 padding 就沒用了

# Bleichenbacher 2006 (BB06)

- 介紹一個 Real World 的例子 Bleichenbacher 2006 年的論文
- 針對 PKCS#1 v1.5 (RFC 2313) 格式的 Signature Forgery
- 後續有一系列的衍生的攻擊
  - 2016 : RSA Signature Forgery in python-rsa [18]
  - 2019 : A Decade After Bleichenbacher '06, RSA Signature Forgery Still Works [19]

-  R. L. Rivest and R. D. Silverman, “Are strong 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.
-  M. J. Wiener, “Cryptanalysis of short rsa secret exponents,” *IEEE Transactions on Information theory*, vol. 36, no. 3, pp. 553–558, 1990.
-  B. De Weger, “Cryptanalysis of rsa with small prime difference,” *Applicable Algebra in Engineering, Communication and Computing*, vol. 13, no. 1, pp. 17–28, 2002.

-  S. Maitra and S. Sarkar, “Revisiting wiener’s attack—new weak keys in rsa,” in *International Conference on Information Security*, pp. 228–243, Springer, 2008.
-  C.-Y. Chen, C.-C. Hsueh, and Y.-F. Lin, “A generalization of de weger’s method,” in *2009 Fifth International Conference on Information Assurance and Security*, vol. 1, pp. 344–347, IEEE, 2009.
-  A. Nitaj, “Diophantine and lattice cryptanalysis of the rsa cryptosystem,” in *Artificial Intelligence, Evolutionary Computing and Metaheuristics*, pp. 139–168, Springer, 2013.
-  M. Asbullah, *Cryptanalysis on the Modulus  $N = p_2q$  and Design of Rabin-like Cryptosystem Without Decryption Failure*.  
PhD thesis, PhD thesis, Universiti Putra Malaysia, 2015.



M. Kamel Ariffin, S. Abubakar, F. Yunus, and M. Asbullah, "New cryptanalytic attack on rsa modulus  $n = pq$  using small prime difference method," *Cryptography*, vol. 3, no. 1, p. 2, 2019.



N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining your ps and qs: Detection of widespread weak keys in network devices," in *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, pp. 205–220, 2012.



N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni, et al., "{DROWN}: Breaking {TLS} using sslv2," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 689–706, 2016.



H. Böck, J. Somorovsky, and C. Young, "Return of bleichenbacher's oracle threat ({ROBOT})," in *27th*

{USENIX} Security Symposium ({USENIX} Security 18), pp. 817–849, 2018.



E. Ronen, R. Gillham, D. Genkin, A. Shamir, D. Wong, and Y. Yarom, “The 9 lives of bleichenbacher’s cat: New cache attacks on tls implementations,” in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 435–452, IEEE, 2019.



A. May, “Using lll-reduction for solving rsa and factorization problems,” in *The LLL algorithm*, pp. 315–348, Springer, 2009.



D. J. Bernstein, Y.-A. Chang, C.-M. Cheng, L.-P. Chou, N. Heninger, T. Lange, and N. Van Someren, “Factoring rsa keys from certified smart cards: Coppersmith in the wild,” in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 341–360, Springer, 2013.



M. Nemec, M. Sys, P. Svenda, D. Klinec, and V. Matyas, “The return of coppersmith’s attack: Practical factorization of



widely used rsa moduli,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1631–1648, ACM, 2017.



F. Valsorda, “Bleichenbacher’06 signature forgery in python-rsa,” May 2016.



S. Y. Chau, “A decade after bleichenbacher’06, rsa signature forgery still works,”