

Cryptography 111

TOC

Today

- Homomorphic
- Elliptic Curve
- Signature
- Quantum
- LFSR
- PRNG

同態加密 | Homomorphic Encryption

同態 | Homomorphism

定義

- $f(x * y) = f(x) * f(y)$

Partial homomorphic encryption

- 部份運算 (e.g. RSA：乘法、次方)

Fully homomorphic encryption

- 任意運算 (e.g. Lattice-based)
- 

回顧RSA | Recall RSA

定義:

- $E(m) := m^e \bmod N$

乘法

- $E(m_1) * E(m_2) = m_1^e * m_2^e = (m_1 * m_2)^e = E(m_1 * m_2)$

次方

- $E(m_1)^{m_2} = (m_1^e)^{m_2} = (m_1^{m_2})^e = E(m_1^{m_2})$

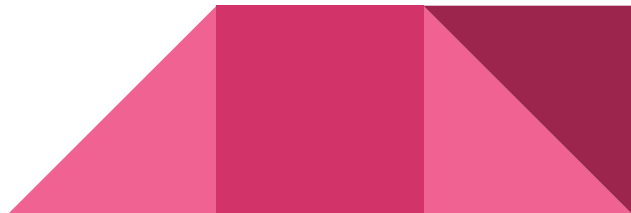
回顧ElGamal | Recall ElGamal

定義

- $E(m) := m s$

乘法

- $E(m_1) * m_2 = (m_1 s) m_2 = (m_1 m_2) s = E(m_1 * m_2)$




Paillier Cryptosystem

定義

- $E(m) := g^m * r^N \bmod N^2$
- $N = pq, r = \text{rand}(Z_{N^2}^*)$

特點

- 同態運算：加法，乘法
 - 密文不固定
 - 離散對數(?)
- 

離散對數 | Discrete Logarithm

定義

- $b^x = a$, 給定 a, b 找出 x

難度

- Subexponential (classical): Number Field Sieve
- Polynomial (quantum): Shor's algorithm



離散對數特例 | A Special Case of Discrete Logarithm

$b^x = a \pmod{N^2}$ ，給定 a, b 找出 $x \pmod{N}$

先考慮 $b = g = N + 1$ 的情況

- $b^x = (N + 1)^x = 1 + N^1 C_1^x + N^2 C_2^x + N^3 C_3^x \dots = 1 + Nx \pmod{N^2}$
- $\frac{a-1}{N} = (x \pmod{N})$ ，這裡的除法是整數除法

在 $b \neq N + 1$ 的情況，可以用換底公式來計算

- $\log_b a = \frac{\log_g a}{\log_g b} \pmod{N}$

Paillier Cryptosystem

金鑰生成

- 私鑰 p, q = primes of equivalent length, $\lambda = \phi(N) = (p - 1)(q - 1)$
- 公鑰 $N = pq$, $g = N + 1$

加密

- $r = \text{rand}(Z_{N^2}^*)$
- $E(m) := g^m r^N \bmod N^2$



Paillier Cryptosystem

解密

- $m = \log_g(E(m)^\lambda \bmod N^2) \lambda^{-1} \bmod N$

正確性

- $\phi(N^2) = p(p-1)q(q-1) = N\lambda$
- $E(m)^\lambda = g^{m\lambda} r^{N\lambda} = g^{m\lambda} \bmod N^2$
- $\log_g(E(m)^\lambda \bmod N^2) \lambda^{-1} = m\lambda \lambda^{-1} = m \bmod N$



Paillier Cryptosystem

同態運算

- $E(m_1) * E(m_2) = E(m_1 + m_2)$
- $E(m_1)^{m_2} = E(m_1 * m_2)$

Task

- Hackover 2018 – oblivious
 - HITCON 2018 – Lost Modulus
- 

離散對數 | Discrete Logarithm

暴力硬解 | Naïve Bruteforce

問題： $b^m = a$ ， $a, b \in G$ ， $\text{ord}(G) = n$ ，給定 a, b, n ，找出 m

算法

- 測試所有可能的 m

複雜度

- 平均 $\frac{n}{2}$ 次會找到，複雜度 $O(n)$

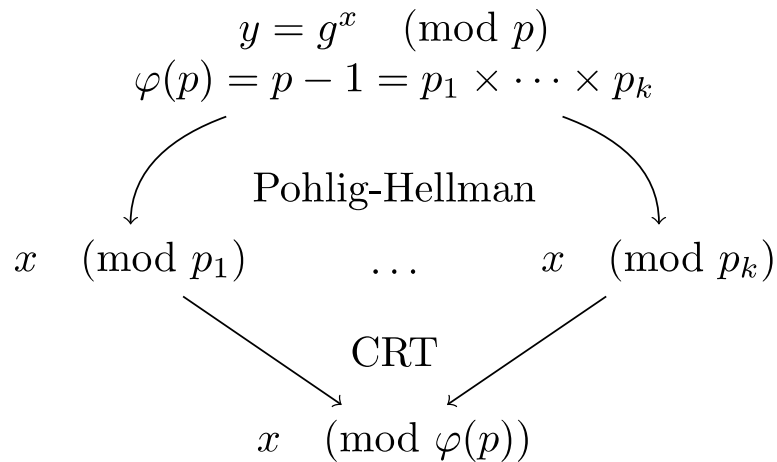


Pohlig-Hellman

在每個 subgroup 解離散對數後用 CRT 組回來

算法

- $\text{ord}(G) = n = p_0^{e_0} p_1^{e_1} p_2^{e_2} \dots$
- $b_i := b^{n/p_i^{e_i}}, a_i := a^{n/p_i^{e_i}}$
- 解 $\log_{b_i} a_i$ 獲得 $m_i = m \bmod p_i^{e_i}$
- 用 CRT 計算 m



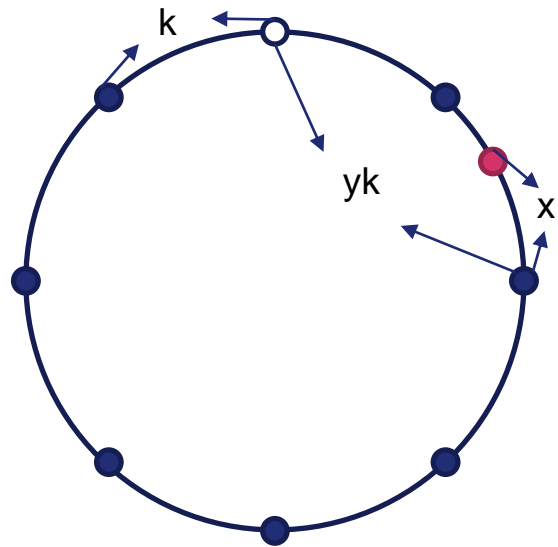
Baby-step Giant-step

算法

- $k := \lfloor \sqrt{n} \rfloor$
- 計算並儲存 b^{yk} , $0 < y \leq k + 1$
- 找到 x, y 使得 $a^x = b^{yk}$, $0 < x < k$
- $m = x^{-1}yk \bmod n$

複雜度

- time = $O(\sqrt{n})$ space = $O(\sqrt{n})$



Pollard's rho

算法

- 找到 $a^x b^y = a^u b^v$
- $mx + y = mu + v \pmod n$
- $m = (v - y) / (x - u) \pmod n$

複雜度

- time = $O(\sqrt{n})$ space = $O(1)$



Pseudo random sequence

定義

- $x_{i+1} = f(x_i)$
- 一遇到重複的就會開始循環

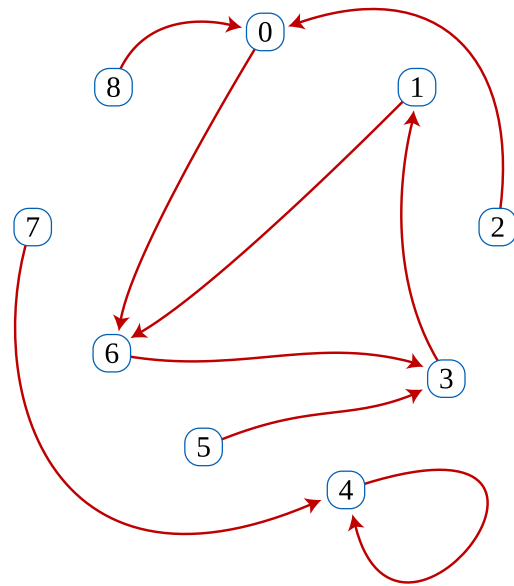
例子

- 從 $x_0 = 2$ 開始：2, 0, 6, 3, 1, 6, 3, 1, ...

複雜度

- time = $O(\sqrt{n})$, birthday paradox

x	$f(x)$
0	6
1	6
2	0
3	1
4	4
5	3
6	3
7	4
8	0



Pseudo random sequence

Pollard's rho random sequence

- $s_i := a^x b^y$
- $s_{i+1} = f(s_i) = \begin{cases} as_i & s \in S_0 \quad (x \leftarrow x + 1) \\ s_i^2 & s \in S_1 \quad (x \leftarrow x * 2, y \leftarrow y * 2) \\ bs_i & s \in S_2 \quad (y \leftarrow y + 1) \end{cases}$
- 找 $a^x b^y = s_i = s_{i+\lambda} = a^u b^v$



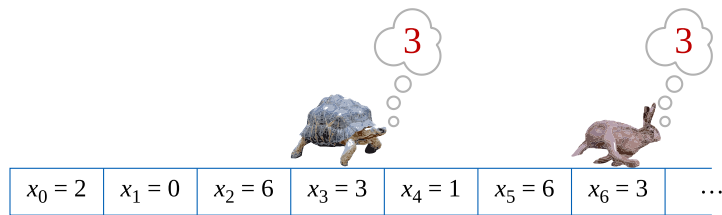
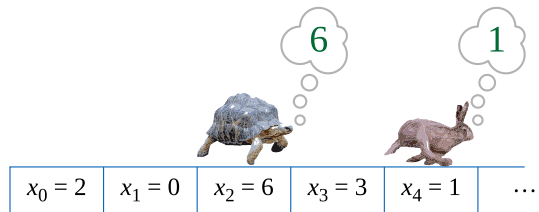
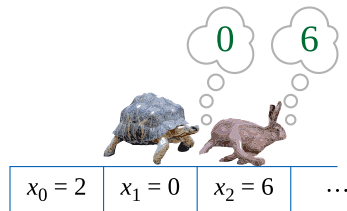
Floyd's Tortoise and Hare

算法

- 檢查 s_i, s_{2i} 是否相等
- $i \leftarrow i + 1$

複雜度

- $O(\lambda)$



Pollard's rho

算法

- 找到 $a^x b^y = a^u b^v$
- $mx + y = mu + v \pmod n$
- $m = (v - y) / (x - u) \pmod n$

複雜度

- time = $O(\sqrt{n})$ space = $O(1)$




簽章 | Signature

數字簽章算法 | DSA

參數生成

- $p = aq + 1$ ，其中 p, q 都是質數， a 是正整數
- $g = h^a \bmod p$ ，其中 h 是正整數 (通常是2)， $g \neq 1$
- 公開參數為 (p, q, g)

金鑰生成

- 私鑰：隨機生成在 0 到 q 之間的數字 x
 - 公鑰： $y = g^x$
- 

數字簽章算法 | DSA

簽名

- 隨機生成在 1 到 q 之間的正整數 k
- $r = (g^k \bmod p) \bmod q$
- $s = k^{-1}(H(m) + xr) \bmod q$



數字簽章算法 | DSA

驗證

- 檢查 $0 < r < q$ 以及 $0 < s < q$
- $w = s^{-1} \mod q$
- $u_1 = H(m) w \mod q$
- $u_2 = r w \mod q$
- $v = (g^{u_1} y^{u_2} \mod p) \mod q$
- 檢查 $v = r$

數字簽章算法 | DSA

正確性

- $w = s^{-1} = k (H(m) + xr)^{-1} \mod q$
- $v = (g^{H(m)w} y^{rw} \mod p) \mod q$
- $v = (g^{H(m)w} g^{xrw} \mod p) \mod q$
- $v = (g^{(H(m)+xr)w} \mod p) \mod q$
- $v = (g^k \mod p) \mod q$

Reuse k

重複使用 k (或是 k 之間有些關係) 會造成私鑰 x 能被還原出來

- $s_i = k^{-1} (H(m_i) + xr)$
- $s_1 - s_0 = k^{-1} (H(m_1) - H(m_0))$

解決方法

- $k = \text{HMAC}(m')$ ，其中 m' 從 $H(m)$ 經過一些運算後取得

現實案例

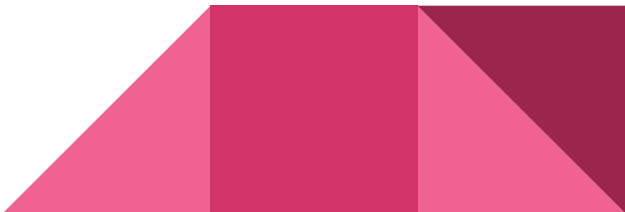
- Sony PS3 ECDSA key recover

Ed25519

設計上盡量避免了實做上容易犯錯的點

- $k = H(H_{b \dots 2b-1}(k), M)$

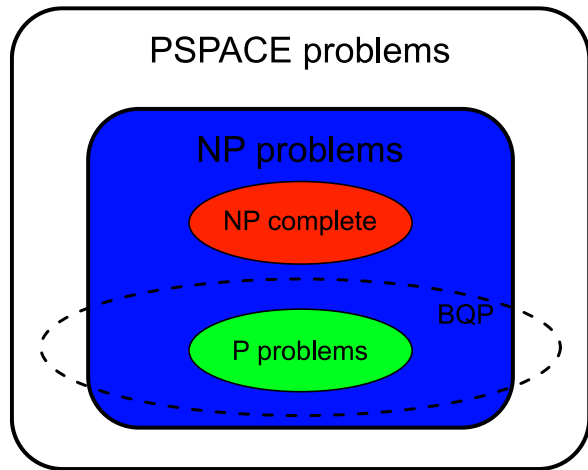
使用 Edwards Curve

- 運算速度較一般的橢圓曲線快速
 - Montgomery ladder 避免旁路攻擊
 - 同樣 security-level 簽章長度較短
- 

量子 | Quantum

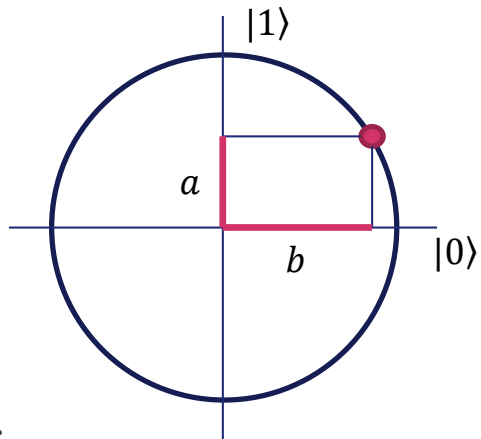
BQP

- Bounded-error Quantum Polynomial time
- 有至少 $\frac{1}{3}$ 的機率會輸出正確答案
- 傳統電路可以轉換成量子電路
 - BQP 包含 P
- BQP $\stackrel{?}{=}$ NP



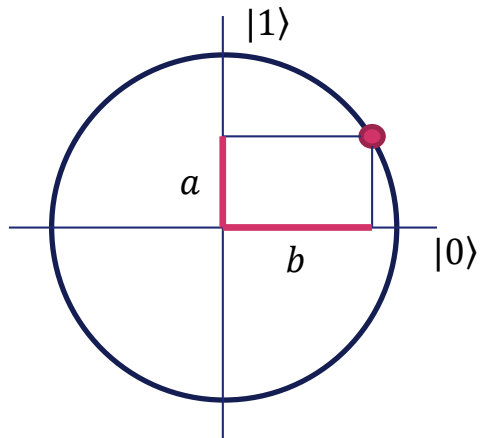
Basic Quantum

- 傳統上一個 bit 只會是 0 或是 1
- 而 qubit 可以是 0 和 1 中間的疊加態
 - 寫成 $a|0\rangle + b|1\rangle$ ，其中 $a^2 + b^2 = 1$
- 沒有辦法複製一顆量子
- 能夠做兩種操作：測量或乘上一個 Unitary Matrix
 - X: NOT, CX: XOR, CCX: AND ...



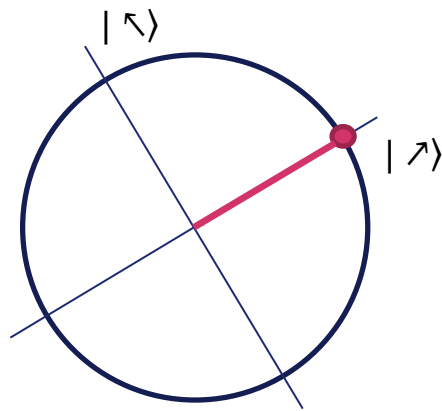
Measurement

- 沒有辦法直接測量出 a, b 是什麼
- 我們可以選一組基底做測量
 - 以 $|0\rangle$ 和 $|1\rangle$ 做測量的話，
 - 有 a^2 的機率塌陷到 $|1\rangle$
 - 有 b^2 的機率塌陷到 $|0\rangle$



Measurement

- 沒有辦法直接測量出 a, b 是什麼
- 我們可以選一組基底做測量
 - 以 $|\nearrow\rangle$ 和 $|\nwarrow\rangle$ 做測量的話，
 - 有 0 的機率塌陷到 $|\nwarrow\rangle$
 - 有 1 的機率塌陷到 $|\nearrow\rangle$



Entanglement

多於一顆但是互相獨立

- $(a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$

多於一顆且互相纏結

- $a|00\rangle + b|11\rangle$
- 當我們測量一顆得到 0，代表狀態塌陷到 $|00\rangle$ ，測量另一顆一定也是 0



Shor's algorithm

在 Polynomial time 做整數分解
算法

- 隨機選一個小於 N 的正整數 a
- 使用量子演算法找出 r 使得 $f(x) = a^x \bmod N$, $f(x) = f(x + r)$
- $a^r = 1 \bmod N$
- $a^r - 1 = (a^{r/2} - 1)(a^{r/2} + 1) = kN$
- $p = \gcd(N, (a^{r/2} - 1 \bmod N))$

Period finding

算法

- $Q = 2^q$, $N^2 < Q < 2N^2$
- 建造 x 為 0 到 Q 所有數字的疊加態 (所有 q 個 bit 都互相獨立且在 $|0\rangle + |1\rangle$ 的狀態)
- 計算 $f(x)$ ，並且測量 $f(x)$ 得到 y
- x 塌陷到所有 $a^x = y$ 的解 $(x_0, x_0 + r, x_0 + 2r \dots)$ 的疊加態
- 對 x 進行 QFT，得到 $(r, 2r, 3r \dots)$ 的疊加態
- 測量並計算與 Q 的連分數展開得到 r

Grover's algorithm

給定函數 $f(x) = \begin{cases} -1 & x = w \\ 1 & x \neq w \end{cases}$, $0 \leq x < N$, 找出 w 。

舉例來說

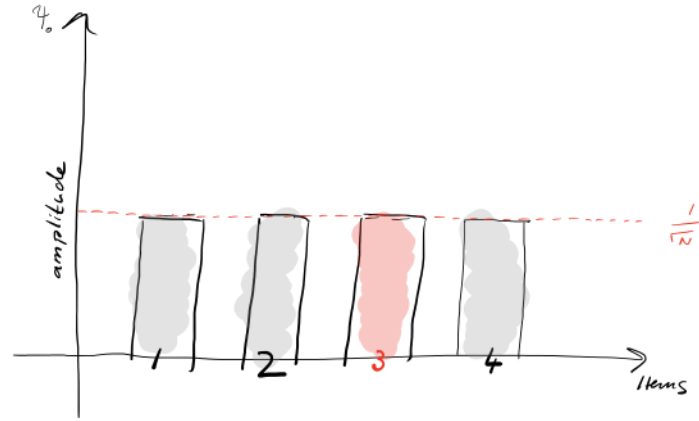
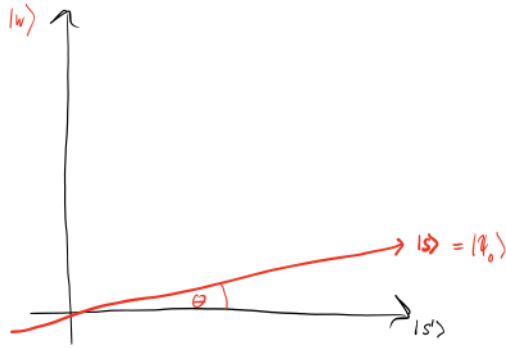
- $f(x) = \begin{cases} -1 & \text{SHA256}(x) = h \\ 1 & \text{SHA256}(x) \neq h \end{cases}$

概念：一直放大 $|w\rangle$ 的機率然後測量獲得 w

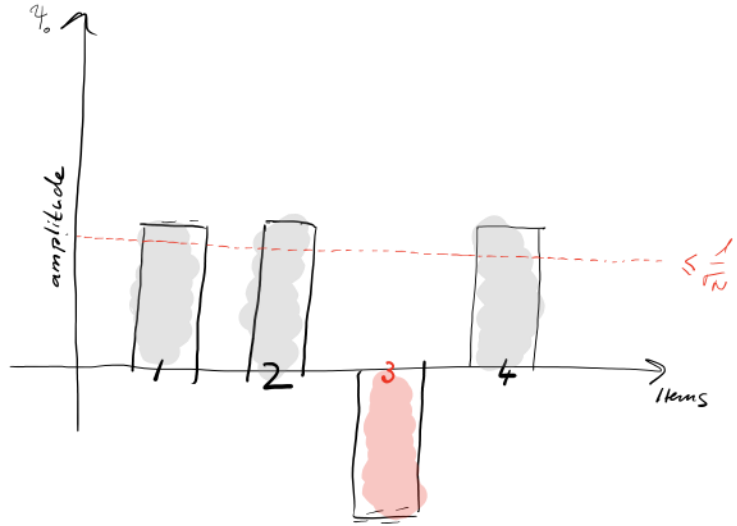
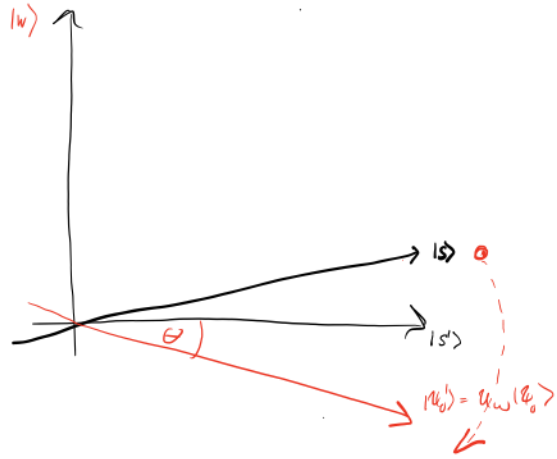
複雜度： $O(\sqrt{N})$



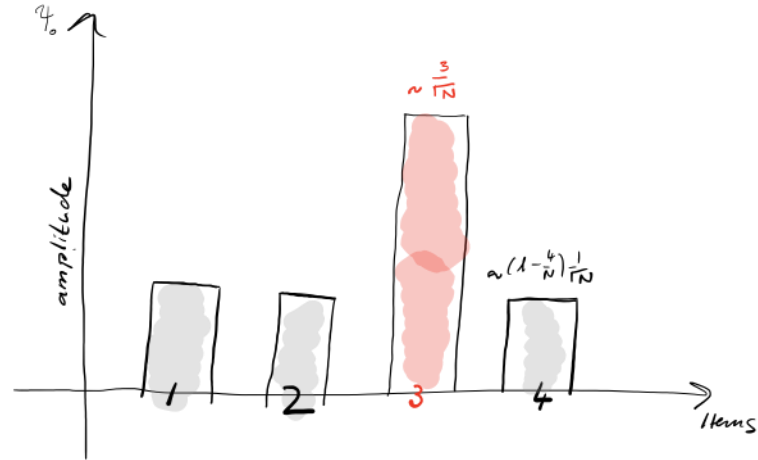
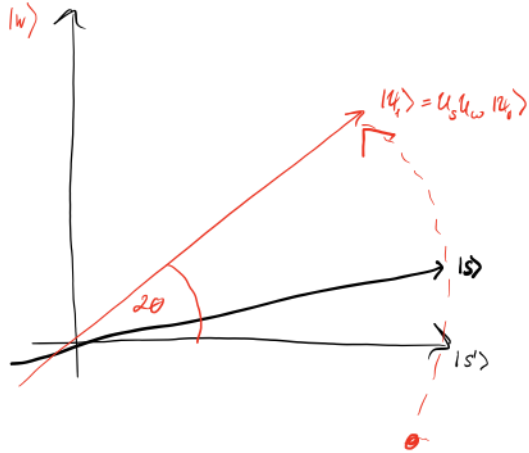
Grover's algorithm



Grover's algorithm



Grover's algorithm



BB84

基於量子不可複製和測不準的特性 (c.f. DHKE，基於Dlog的難度)

Alice的隨機位元	0	1	1	0	1	0	0	1
Alice隨機選擇的基	+	+	×	+	×	×	×	+
Alice所傳光子的偏振態	↑	→	↘	↑	↘	↗	↗	→
Bob隨機選擇測量的基	+	×	×	×	+	×	+	+
Bob測量的光子的偏振態	↑	↗	↘	↗	→	↗	→	→
在公共通道中對比基								
共有的金鑰	0		1			0		1

Survivors

Asymmetric

✗ Factoring: RSA, Paillier ...

✗ Discrete Log: DHKE, ElGamal, ECC ...

✓ Lattice-based

✓ Hash-based: Lamport signature ...

...

Symmetric

✓ Double the key size due to Grover's algorithm

...



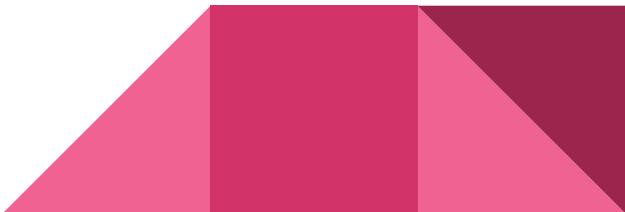
偽亂數產生器 | PRNG

Linear congruential generator (LCG)

定義

- $x_{i+1} = ax_i + b \pmod{m}$

變形

- Raw : $x_{i-1} = a^{-1}(x_i - b) \pmod{m}$
 - 保留 Low order bits : 估上下界
 - 保留 High order bits : LLL reduction
- 

XORShift

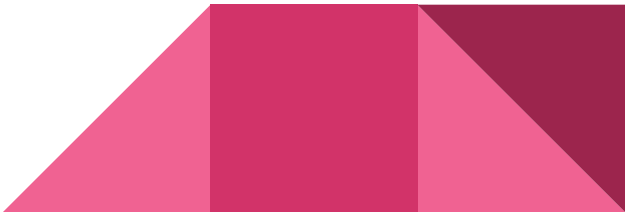
運算速度快，程式碼簡單

定義

- $x \oplus = (x \ll a); x \oplus = (x \gg b); x \oplus = (x \ll c);$

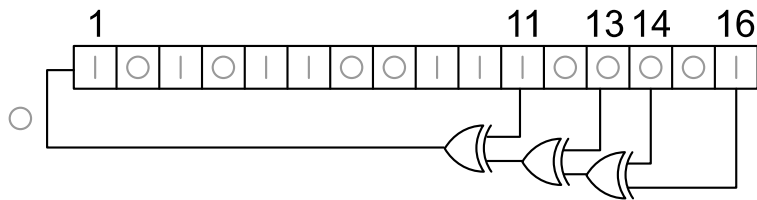
分析技巧

- 把 x 看成 bit vector，XOR 和 Shift 可以寫成 GF(2) 下的矩陣乘法



LFSR

- 暫存器的初始值是 seed
- 一次輸出一個 bit
 - 輸出的 bit 會變成第一個暫存器的值
- 分析技巧
 - 跟 XORShift 一樣用 bit vector 和 GF(2) 矩陣來處理



Filtered LFSR

- 從輸出第一個暫存器改成對整個state做非線性運算

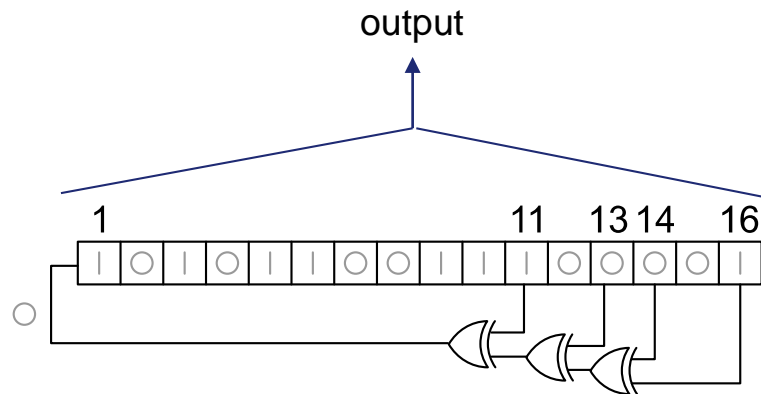
- $s_{i+1} = \text{LFSR}(s_i); \quad o = f(s_{i+1})$

- 分析方法

- 找 f 的線性 (或low degree) annihilator g

- $\forall x : f(x) = 1, \quad g(x) = 0$

- 對於所有 $f(s_i) = 1$ 的地方用 $g(s_i)$ 建聯立方程式並求解



Mersenne Twister (MT19937)

- 週期長、沒專利、通過很多測試 …
- 許多語言的預設算法：MATLAB, PHP, Python, R, Ruby, Octave …
- 狀態為 624 個 32-bits 的數字
- x_i 由 x_{i-624} , x_{i-623} , x_{i-227} 三個數字運算而成
- 輸出前會先經過一個可逆的非線性函數
 - 可以由連續 624 個 32bits 的輸出還原出狀態
 - 可以由足夠多獨立的 bits 解方程式還原出狀態

CSPRNG

- 定義
 - 給定 k 個 bits，沒有已知的多項式時間算法能以高於 $1/2$ 的機率輸出第 $k+1$ bit。
- Cipher-based
 - CTR mode, IV = seed
 - Stream cipher, IV = seed
- Hash-based
 - $x_{i+1} = x_i + 1; \quad o_{i+1} = H(x_i)$