密码学伪代码总结

1　线性密码分析

算法 3.2　线性攻击$(\mathcal{T}, T, \pi_S^{-1})$

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)

　do $\text{Count}[L_1, L_2] \leftarrow 0$

for each $(x, y) \in \mathcal{T}$

　do {
　　for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
　　do {
　　　$v^4_{<2>} \leftarrow L_1 \oplus y_{<2>}$
　　　$v^4_{<4>} \leftarrow L_2 \oplus y_{<4>}$
　　　$u^4_{<2>} \leftarrow \pi_S^{-1}(v^4_{<2>})$
　　　$u^4_{<4>} \leftarrow \pi_S^{-1}(v^4_{<4>})$
　　　$z \leftarrow x_5 \oplus x_7 \oplus x_8 \oplus u^4_6 \oplus u^4_8 \oplus u^4_{14} \oplus u^4_{16}$
　　　if $z = 0$
　　　　then $\text{Count}[L_1, L_2] \leftarrow \text{Count}[L_1, L_2] + 1$

　$\max \leftarrow -1$

　for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)

　do {
　　$\text{Count}[L_1, L_2] \leftarrow |\text{Count}[L_1, L_2] - T/2|$
　　if $\text{Count}[L_1, L_2] > \max$
　　　then {
　　　　$\max \leftarrow \text{Count}[L_1, L_2]$
　　　　$\text{maxkey} \leftarrow (L_1, L_2)$

output(maxkey)

2　差分密码分析

算法 3.3　差分攻击$(\mathcal{T}, T, \pi_S^{-1})$

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)

　do $\mathrm{Count}[L_1, L_2] \leftarrow 0$

for each $(x, y, x^*, y^*) \in \mathcal{T}$

do
- **if** $(y_{<1>} = (y_{<1>})^*)$ **and** $(y_{<3>} = (y_{<3>})^*)$
- **then**
  - **for** $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
  - **do**
    - $v^4_{<2>} \leftarrow L_1 \oplus y_{<2>}$
    - $v^4_{<4>} \leftarrow L_2 \oplus y_{<4>}$
    - $u^4_{<2>} \leftarrow \pi_S^{-1}(v^4_{<2>})$
    - $u^4_{<4>} \leftarrow \pi_S^{-1}(v^4_{<4>})$
    - $(v^4_{<2>})^* \leftarrow L_1 \oplus (y_{<2>})^*$
    - $(v^4_{<4>})^* \leftarrow L_2 \oplus (y_{<4>})^*$
    - $(u^4_{<2>})^* \leftarrow \pi_S^{-1}((v^4_{<2>})^*)$
    - $(u^4_{<4>})^* \leftarrow \pi_S^{-1}((v^4_{<4>})^*)$
    - $(u^4_{<2>})' \leftarrow u^4_{<2>} \oplus (u^4_{<2>})^*$
    - $(u^4_{<4>})' \leftarrow u^4_{<4>} \oplus (u^4_{<4>})^*$
    - **if** $((u^4_{<2>})' = 0110)$ **and** $((u^4_{<4>})' = 0110)$
    - 　**then** $\mathrm{Count}[L_1, L_2] \leftarrow \mathrm{Count}[L_1, L_2] + 1$

$\max \leftarrow -1$

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)

do
- **if** $\mathrm{Count}[L_1, L_2] > \max$
- **then**
  - $\max \leftarrow \mathrm{Count}[L_1, L_2]$
  - $\mathrm{maxkey} \leftarrow (L_1, L_2)$

output(maxkey)

3　AES算法

总流程：

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])

   for round = 1 step 1 to Nr-1
      SubBytes(state)
      ShiftRows(state)
      MixColumns(state)
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

密钥扩展：

算法 3.6 KeyExpansion(key)

external RotWord, SubWord

RCon[1] $\leftarrow$ 01000000

RCon [2] $\leftarrow$ 02000000

RCon [3] $\leftarrow$ 04000000

RCon [4] $\leftarrow$ 08000000

RCon [5] $\leftarrow$ 10000000

RCon [6] $\leftarrow$ 20000000

RCon [7] $\leftarrow$ 40000000

RCon [8] $\leftarrow$ 80000000

RCon [9] $\leftarrow$ 1B000000

RCon [10] $\leftarrow$ 36000000

for $i \leftarrow 0$ **to** 3

  do $w[i] \leftarrow (\text{key}[4i], \text{key}[4i+1], \text{key}[4i+2], \text{key}[4i+3])$

for $i \leftarrow 4$ **to** 43

  do $\begin{cases} \text{temp} \leftarrow w[i-1] \\ \textbf{if } i \equiv 0 (\text{mod } 4) \\ \quad \textbf{then } \text{temp} \leftarrow \text{SubWord}(\text{RotWord}(\text{temp})) \oplus \text{RCon}[i/4] \\ w[i] \leftarrow w[i-4] \oplus \text{temp} \end{cases}$

return $(w[0], \cdots, w[43])$

4 Merkle-Damgård算法

算法 Merkle-Damgård(x)

external compress

注释 **compress**: $\{0,1\}^{m+t} \to \{0,1\}^m$，其中$t \geqslant 2$

$n \leftarrow |x|$

$k \leftarrow \lceil n/(t-1) \rceil$

$d \leftarrow n - k(t-1)$

for $i \leftarrow 1$ **to** $k-1$

  do $y_i \leftarrow x_i$

$y_k \leftarrow x_k \| 0^d$

$y_{k+1} \leftarrow d$ 的二进制表示

$z_1 \leftarrow 0^{m+1} \| y_1$

$g_1 \leftarrow \textbf{compress}(z_1)$

for $i \leftarrow 1$ **to** k

  do $\begin{cases} z_{i+1} \leftarrow g_i \| 1 \| y_{i+1} \\ g_{i+1} \leftarrow \textbf{compress}(z_{i+1}) \end{cases}$

$h(x) \leftarrow g_{k+1}$

return $(h(x))$

5　SHA-1安全哈希

填充算法

算法　SHA-1-PAD(x)

注释　$|x| \leqslant 2^{64}-1$

$d \leftarrow (447 - |x|) \bmod 512$

$\ell \leftarrow |x|$的二进制表示，其中$|\ell| = 64$

$y \leftarrow x \| 1 \| 0^d \| \ell$

6　辗转相除法

算法　Euclidean Algorithm(a, b)

$r_0 \leftarrow a$

$r_1 \leftarrow b$

$m \leftarrow 1$

while $r_m \neq 0$

$$\textbf{do}\begin{cases} q_m \leftarrow \left\lfloor \dfrac{r_{m-1}}{r_m} \right\rfloor \\ r_{m+1} \leftarrow r_{m-1} - q_m r_m \\ m \leftarrow m+1 \end{cases}$$

$m \leftarrow m-1$

return$(q_1, \cdots, q_m; r_m)$

comment:　$r_m = \gcd(a, b)$

7 扩展Euclidean算法

Algorithm EXTENDED EUCLIDEAN ALGORITHM(a, b)

$a_0 \leftarrow a$
$b_0 \leftarrow b$
$t_0 \leftarrow 0$
$t \leftarrow 1$
$s_0 \leftarrow 1$
$s \leftarrow 0$
$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
$r \leftarrow a_0 - qb_0$
while $r > 0$

$$\textbf{do} \begin{cases} temp \leftarrow t_0 - qt \\ t_0 \leftarrow t \\ t \leftarrow temp \\ temp \leftarrow s_0 - qs \\ s_0 \leftarrow s \\ s \leftarrow temp \\ a_0 \leftarrow b_0 \\ b_0 \leftarrow r \\ q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor \\ r \leftarrow a_0 - qb_0 \end{cases}$$

$r \leftarrow b_0$
return (r, s, t)
comment: $r = \gcd(a, b)$ and $sa + tb = r$

8 求逆

Algorithm MULTIPLICATIVE INVERSE(a, b)

$a_0 \leftarrow a$
$b_0 \leftarrow b$
$t_0 \leftarrow 0$
$t \leftarrow 1$
$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
$r \leftarrow a_0 - qb_0$
while $r > 0$

$$\textbf{do} \begin{cases} temp \leftarrow (t_0 - qt) \bmod a \\ t_0 \leftarrow t \\ t \leftarrow temp \\ a_0 \leftarrow b_0 \\ b_0 \leftarrow r \\ q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor \\ r \leftarrow a_0 - qb_0 \end{cases}$$

if $b_0 \neq 1$
  then b has no inverse modulo a
  else return (t)

9 平方乘算法

$$x^c \bmod n$$

$$c=\sum_{i=0}^{\ell-1} c_i 2^i \qquad c_i=0\text{或}1,\ 0\leqslant i\leqslant \ell-1。$$

算法 Square-and-Multiply(x, c, n)

$z \leftarrow 1$

for $i \leftarrow \ell-1$ **downto** 0

do $\begin{cases} z \leftarrow z^2 \bmod n \\ \textbf{if}\ \ c_i = 1 \\ \quad \textbf{then}\ \ z \leftarrow (z \times x) \bmod n \end{cases}$

return(z)

$\ell \leq$ 模乘次数 $\leq 2\ell$

10 Miller-Rabin算法

算法 5.7 Miller-Rabin (n)

把 $n-1$ 写成 $n-1=2^k m$，其中 m 是一个奇数

随机选取整数 a，使得 $1\leqslant a\leqslant n-1$

$b \leftarrow a^m \bmod n$

if $b \equiv 1 \pmod n$

  then return ("n is prime")

for $i \leftarrow 0$ **to** $k-1$

do $\begin{cases} \textbf{if}\ b \equiv -1 \pmod n \\ \quad \textbf{then return}(\text{“}n \text{ is prime”}) \\ \quad \textbf{else}\ b \leftarrow b^2 \bmod n \end{cases}$

return("n is composite")

11 Solovay-Strassen算法

算法 Solovay-Strassen (n)

随机选取整数 a，使得$1 \leqslant a \leqslant n-1$

$x \leftarrow \left(\dfrac{a}{n}\right)$

if $x=0$

then return ("n is composite")

$y \leftarrow a^{(n-1)/2} (\bmod n)$

if $x \equiv y (\bmod n)$

then return ("n is prime")

else return ("n is composite")

12 Pollard p-1算法

Algorithm POLLARD $p-1$ FACTORING ALGORITHM(n, B)

$a \leftarrow 2$

for $j \leftarrow 2$ **to** B

do $a \leftarrow a^j \bmod n$

$d \leftarrow \gcd(a-1, n)$

if $1 < d < n$

then return (d)

else return ("failure")

13 图灵规约算法

Algorithm RABIN ORACLE FACTORING(n)

external RABIN DECRYPT

choose a random integer $r \in \mathbb{Z}_n{}^*$

$y \leftarrow r^2 \bmod n$

$x \leftarrow$ RABIN DECRYPT(y)

if $x \equiv \pm r \pmod n$

then return ("failure")

else $\begin{cases} p \leftarrow \gcd(x+r, n) \\ q \leftarrow n/p \\ \textbf{return } (\text{“}n = p \times q\text{”}) \end{cases}$

14 SHANKS算法

算法 SHANKS(G, n, α, β)

1. $m \leftarrow \lceil \sqrt{n} \rceil$
2. **for** $j \leftarrow 0$ **to** $m-1$
   do 计算 α^{mj}
3. 对 m 个有序对 (j, α^{mj}) 关于第二个坐标排序，得到一个列表 L_1
4. **for** $i \leftarrow 0$ **to** $m-1$
   do 计算 $\beta\alpha^{-i}$
5. 对 m 个有序对 $(i, \beta\alpha^{-i})$ 关于第二个坐标排序，得到一个列表 L_2
6. 找到对 $(j, y) \in L_1$ 和 $(i, y) \in L_2$（即找到两个具有相同第二坐标的对）
7. $\log_\alpha \beta \leftarrow (mj + i) \bmod n$

15 椭圆曲线点压缩算法

算法 Point-Decompress(x, i)

$z \leftarrow x^3 + ax + b \bmod p$

if z 是模 p 的非二次剩余类

then return("failure")

$$\textbf{else}\begin{cases} y \leftarrow \sqrt{z} \bmod p \\ \textbf{if } y \equiv i \pmod 2 \\ \quad \textbf{then return } (x, y) \\ \quad \textbf{else return } (x, p-y) \end{cases}$$

16 倍加和差算法

算法 倍数-和差算法 Double-And-(Add Or Subtract) $(P, (c_{\ell-1}, \cdots, c_0))$

$Q \leftarrow \mathcal{O}$

for $i \leftarrow \ell - 1$ **downto** 0

$$\textbf{do}\begin{cases} Q \leftarrow 2Q \\ \textbf{if } c_i = 1 \\ \quad \textbf{then } Q \leftarrow Q + P \\ \quad \textbf{else if } c_i = -1 \\ \quad \textbf{then } Q \leftarrow Q - P \end{cases}$$

return(Q)

17 离散对数泄露算法

算法 L_2 Oracle-Discrete-Logarithm (p, α, β)

external L_1, **Oracle** L_2

$x_0 \leftarrow L_1(\beta)$

$\beta \leftarrow \beta / \alpha^{x_0} \bmod p$

$i \leftarrow 1$

while $\beta \neq 1$

do $\begin{cases} x_i \leftarrow \text{Oracle } L_2(\beta) \\ \gamma \leftarrow \beta^{(p+1)/4} \bmod p \\ \textbf{if } L_1(\gamma) = x_i \\ \quad \textbf{then } \beta \leftarrow \gamma \\ \quad \textbf{else } \beta \leftarrow p - \gamma \\ \beta \leftarrow \beta / \alpha^{x_i} \bmod p \\ i \leftarrow i + 1 \end{cases}$

return$(x_{i-1}, x_{i-2}, \cdots, x_0)$

$$\log_\alpha \beta = \sum_{i \geqslant 0} x_i 2^i$$

18 BBS生成器算法

算法 8.5 Blum-Blum-Shub 生成器

设 p，q 是两个满足 $p \equiv q \equiv 3 \bmod 4$ 的 $(k/2)$ 比特素数，定义 $n = pq$。QR(n) 表示模 n 的二次剩余的集合。

一个种子 s_0 是 QR(n) 中的任何一个元素。对 $0 \leqslant i \leqslant \ell - 1$，定义

$$s_{i+1} = s_i^2 \bmod n$$

然后定义

$$f(s_0) = (z_1, z_2, \cdots, z_\ell)$$

其中

$$z_i = s_i \bmod 2$$

$1 \leqslant i \leqslant \ell$。那么 f 是一个 (k, ℓ) PRBG，称为 Blum-Blum-Shub 生成器，简写为 BBS 生成器。

一种选择合适种子的方法是先选择一个 $s_{-1} \in \mathbf{Z}_n^*$，然后计算 $s_0 = s_{-1}^2 \bmod n$。这保证了 $s_0 \in$ QR(n)。

19 QR-TEST算法

算法 8.6 QR-TEST (x, n)

external pbp

$s_1 \leftarrow x^2 \bmod n$

comment: s_1 是一个模 n 的二次剩余

$z_1 \leftarrow s_1 \bmod 2$

由种子 s_1 使用 BBS 生成器计算出 $z_2, \cdots, z_\ell$

$z \leftarrow \mathrm{pbp}(z_1, \cdots, z_\ell)$

if $(x \bmod 2) = z$

  then return (yes)

  else return (no)