北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

《现代密码学》第五讲

PRG和流密码（二）

本章主要内容

- OTP与伪随机数生成器
- 流密码技术的发展
- 基于LFSR的PRG
- RC4算法
- Estream 算法举例
- PRG安全与流密码安全应用

本章主要内容

- OTP与伪随机数生成器
- 流密码技术的发展
- 基于LFSR的PRG
- RC4算法
- Estream 算法举例
- PRG安全与流密码安全应用

北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

软件算法

Salsa20

BUPT ISC 信息安全中心

Salsa20

Salsa20密码由密码学家aniel J. Bernstein (The University of Illinois at Chicago)设计

输入：

密钥k：32个字节(256比特)或者16个字节(128比特)

初始变量(nonce) v：8个字节；

输出：

l个字节的序列（a fixed key and nonce），$l \in \{0, 1, 2, \cdots, 2^{70}\}$

Salsa20

Salsa20分为$Salsa20_{k0,k1}(n)$或$Salsa20_{k}(n)$

输入：k:32-byte或者16-byte序列

n:16-byte序列（8字节的随机数nonce和8字节的block-counter）

输出：64-byte序列

- 对于32-byte的k和16-byte的n，定义

$Salsa20_{k0,k1}(n)=Salsa20(\sigma_0, k_0, \sigma_1, n, \sigma_2, k_1, \sigma_3)$.

其中，$\sigma_0=(101,120,112,97)$，$\sigma_1=(110,100,32,51)$，$\sigma_2=(50,45,98,121)$，$\sigma_3=(116,101,32,107)$.

- 对于16-byte的k，和16-byte的n，定义

$Salsa20_{k}(n)=Salsa20(\tau_0, k, \tau_1, n, \tau_2, k, \tau_3)$.

其中 $\tau_0=(101,120,112,97)$，$\tau_1=(110,100,32,49)$，$\tau_2=(54,45,98,121)$，$\tau_3=(116,101,32,107)$.

Salsa20

Salsa20(x) 定义为：

$$Salsa\ 20(x) = x + doubleroun\ \ d^{10}(x)$$

Doubleround函数

输入：16-word序列

输出：16-word序列

该函数为一个columnround之后接一个rowround，

即： doubleround(x)=rowround(columnround(x))

Salsa20

Salsa20　输入的64个字节x=(x[0]，x[1]，x[2]，…，x[63])输入，到Doubleround函数的16个字输入转换：

$$x_0 = littleendian(x[0], x[1], x[2], x[3]),$$

$$x_1 = littleendian(x[4], x[5], x[6], x[7]),$$

$$x_2 = littleendian(x[8], x[9], x[10], x[11]),$$

$$\vdots$$

$$x_{15} = littleendian(x[60], x[61], x[62], x[63]),$$

Salsa20

Littleendian函数

- 输入：4-byte序列b=(b0,b1,b2,b3)
- 输出：1-word

$$littleendi\ an(b) = b_0 + 2^8 b_1 + 2^{16} b_2 + 2^{24} b_3$$

Salsa20

rowround函数

- 输入：16-word序列y=($y_0, y_1, y_2, y_3, \cdots, y_{15}$)
- 输出：16-word序列z =($z_0, z_1, z_2, z_3, \cdots, z_{15}$)

rowround(y)=($z_0, z_1, z_2, z_3, \cdots, z_{15}$)定义为：

```
(z[0], z[1], z[2], z[3])=QR(y[0], y[1], y[2], y[3]); // row 1
(z[4], z[5], z[6], z[7])=QR(y[5], y[6], y[7], y[4]); // row 2
(z[8], z[9], z[10], z[11])=QR(y[10], y[11], y[8], y[9]); // row 3
(z[12], z[13], z[14],z[15])=QR(y[15], y[12], y[13], y[14]); //
row 4
```

Salsa20

Columnround函数

- 输入：16-word序列x=(x0, x1, x2, x3, …, x15)
- 输出：16-word序列y=(y0, y1, y2, y3, …, y15)

columnround(x)=(y0, y1, y2, y3, …, y15)定义为：

```
(y[0], y[4], y[8], y[12]) = QR(x[0], x[4], x[8], x[12]); // column 1
(y[1]) y[5], y[9], y[13]) = QR(x[5], x[9], x[13], x[ 1]); // column 2
(y[2], y[6], y[10], y[14]) = QR(x[10], x[14], x[2], x[6]); // column 3
(y[3], y[7], y[11], y[15]) = QR(x[15], x[3], x[7], x[11]); // column 4
```

Salsa20

quarterround函数

- 输入：4-word序列
  $a=(a_0, a_1, a_2, a_3)$
- 输出：4-word序列
  $b=(b_0, b_1, b_2, b_3)$

quarterround$(a)=(b_0, b_1, b_2, b_3)$

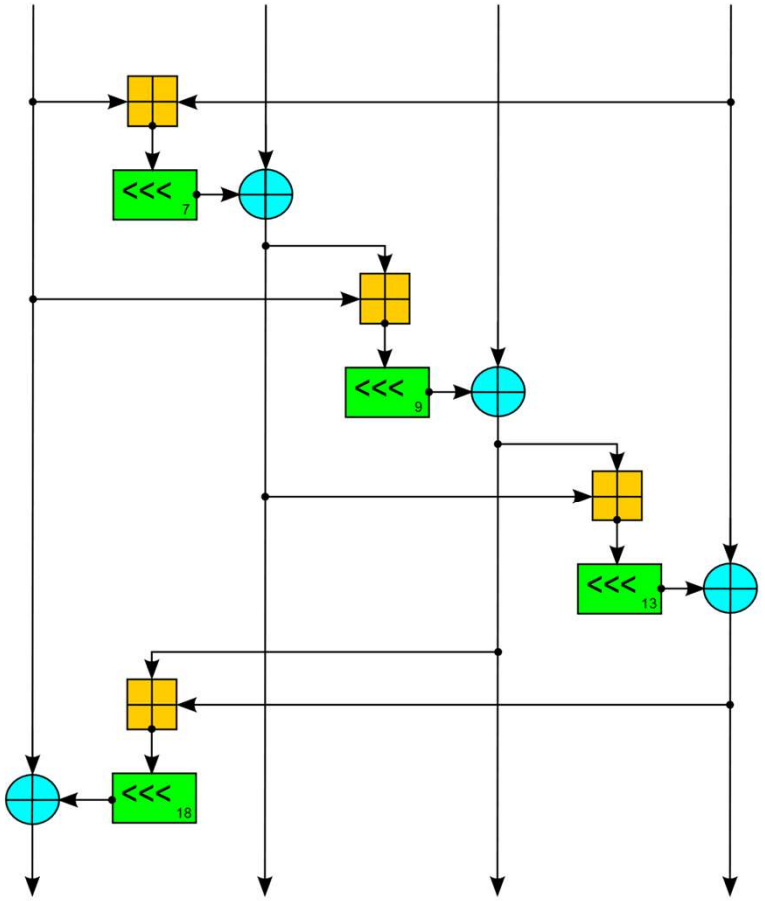
定义为：

$$b_1 = a_1 \oplus ((a_0 + a_3) <<< 7)$$

$$b_2 = a_2 \oplus ((b_1 + a_0) <<< 9)$$

$$b_3 = a_3 \oplus ((b_2 + b_1) <<< 13)$$

$$b_0 = a_0 \oplus ((b_3 + b_2) <<< 18)$$



Salsa20

设Salsa20 的输出与10轮Doubleround 函数的输出 $(z_0, z_1, z_2, z_3, \cdots, z_{15})$ 的转化：

$$littleendian^{-1}(z_0 + x_0),$$
$$littleendian^{-1}(z_1 + x_1),$$
$$littleendian^{-1}(z_2 + x_2),$$
$$\vdots$$
$$littleendian^{-1}(z_{15} + x_{15})$$

北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

硬件算法

Trivium

BUPT ISC 信息安全中心

Trivium

- Trivium由Belgium密码学家C. DeCannière和 B. Preneel设计

- Trivium从一个80-bit的密钥和一个80-bit的起始变量中生成多达2^{64}比特的密钥流.

Trivium

初始化过程

- 密钥长度：80比特， 记为K_1，．．．．，K_{80}
- IV 大小：80比特，记为IV_1，．．．．，IV_{80}
- 中间状态大小： 288比特，记为s_1,．．．．，s_{93}， s_{94}，．．．．，s_{177}， s_{178}，．．．．，s_{288}

1. 将80-bit 的密钥和80-bit 的起始变量分别赋值给内部状态比特s_1～s_{80}和s_{94} ～s_{173}。同时将s_{286}，s_{287}和 s_{288}置为1，剩余所有位置零。

- $(s_1, s_2, \ldots, s_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$
- $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots, 0)$
- $(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$

Trivium

2. 内部状态进行如下4个循环,不输出:

for i = 1 ～ 4 • 288 :

- $t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$
- $t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$
- $t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$
- $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
- $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
- $(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for

Trivium

密钥流生成

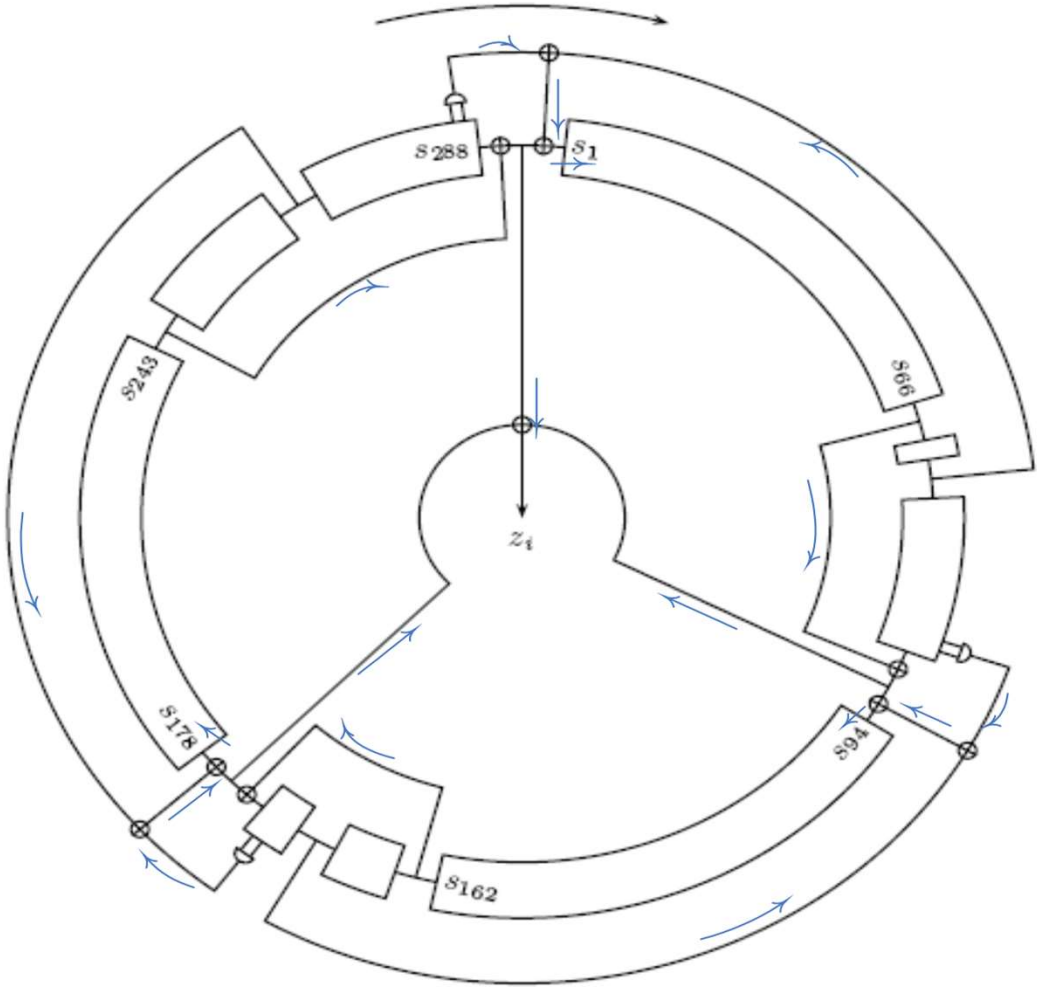
每次迭代，在288-bit的内部状态（s1，. . . . ，s288）中提取15个比特，用它们更新3个内部状态比特，并计算1比特的密钥流z_i. 若需要N位密钥则过程在重复 N次后停止 (N≤ 264). 具体算法如以下代码所示：

for i = 1 to N do (N为所需密钥位数)

- $t_1 \leftarrow s_{66} + s_{93}$
- $t_2 \leftarrow s_{162} + s_{177}$
- $t_3 \leftarrow s_{243} + s_{288}$
- $z_i \leftarrow t_1 + t_2 + t_3$
- $t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$
- $t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$
- $t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$
- $(s_1, s_2, . . . , s_{93}) \leftarrow (t_3, s_1, . . . , s_{92})$
- $(s_{94}, s_{95}, . . . , s_{177}) \leftarrow (t_1, s_{94}, . . . , s_{176})$ $(s_{178}, s_{279}, . . . , s_{288}) \leftarrow (t_2, s_{178}, . . . , s_{287})$

符号‘+’代表异或XOR运算，符号‘.’代表与运算.

Trivium



本章主要内容

- OTP与伪随机数生成器
- 流密码技术的发展
- 基于LFSR的PRG
- RC4算法
- Estream 算法举例
- PRG安全与流密码安全应用

选择密文攻击

目标密文：$(IV, c^*) \in \{0,1\}^t \times \{0,1\}^n$

攻击者：任给$r \in \{0,1\}^n$, 计算$c' = c^* \oplus r$

$(IV, c') \rightarrow$ Decryption oracle

$m' \leftarrow$ Decryption oracle

计算$m' \oplus r$等于m0 or m1

攻击资源	攻击结果
惟密文	恢复密钥
已知明密文	恢复目标明文
选择明文	获得明文信息
选择密文	
自适应选择密文	

攻击方法

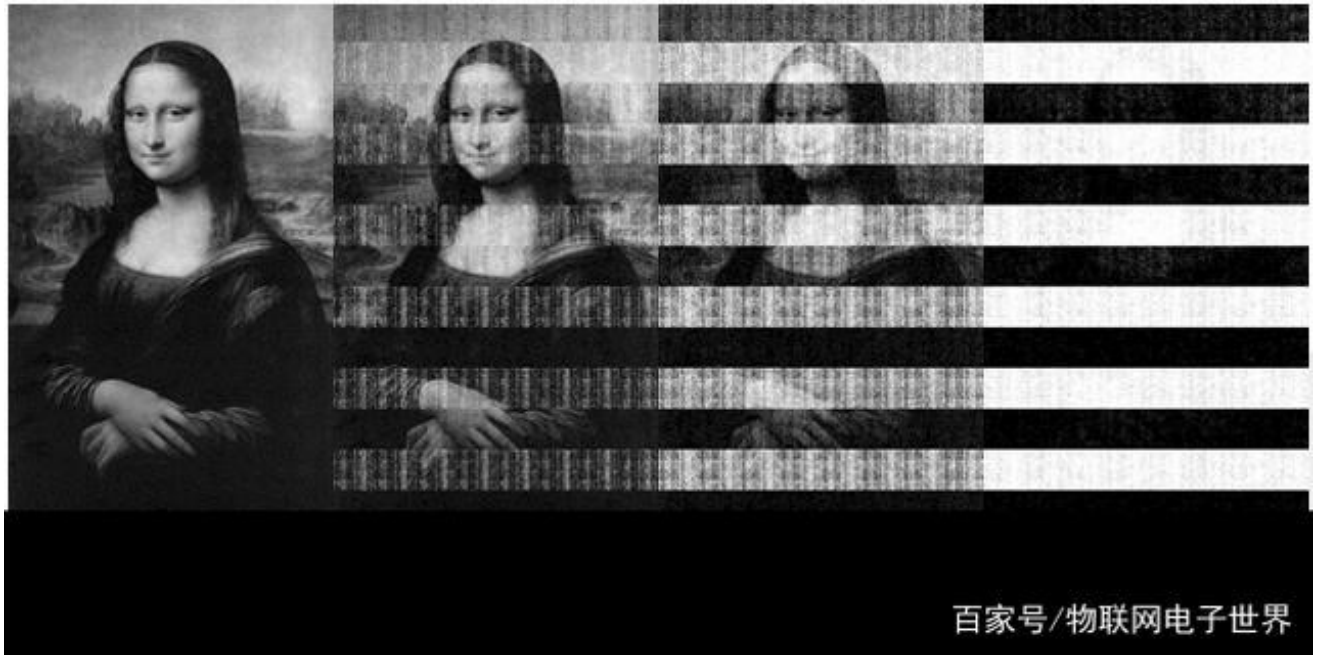
- 代数攻击方法
  - 穷举
  - 中间相遇
  - 频率、重合指数
  - 线性
  - 差分
  - 代数

  。。。。。。

Side Channel Attack-冷冻攻击

在摄氏30度的情况下，数据会残留1至2秒，而在低温状态中，这个时间会长达数分钟

Side Channel Attack-冷冻攻击



掉电后的5秒，30秒，60秒然后5分钟后存储在RAM内的图像慢慢的在消失

THE END !