对 Sosemanuk 算法改进的猜测决定攻击*

谢端强[1]，李 恒[1]，李瑞林[2]，戴清平[1]
（1. 国防科技大学 理学院，湖南 长沙 410073；
2. 国防科技大学 电子科学与工程学院，湖南 长沙 410073）

摘 要：Sosemanuk 算法是欧洲 eSTREAM 计划最终获选的七个算法之一。从比特层面对该算法进行剖析，通过对 Serpent1 组件 S 盒、模 2^{32} 加法和线性反馈移位寄存器的研究，找到了关于内部状态的一个方程组，并利用 Groebner 基方法改进了对 Sosemanuk 算法基于字的猜测决定攻击。结果表明只需要猜测 7 个 32 比特的字就可以完全确定出其余 5 个 32 比特的内部状态，其攻击的复杂度为 $O(2^{192})$。

关键词：eSTREAM 计划；Sosemanuk 算法；猜测决定攻击；Groebner 基
中图分类号：TN918.1 **文献标志码**：A **文章编号**：1001-2486(2012)06-0079-05

Improved guess-and-determine attack on sosemanuk

XIE Duanqiang[1], *LI Heng*[1], *LI Ruilin*[2], *DAI Qingping*[1]
(1. College of Science, National University of Defense Technology, Changsha 410073, China;
2. College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China)

Abstract: Sosemanuk is one of the stream ciphers that have been selected into the final portfolio for the eSTREAM project. Sosemanuk from the view point of bit-level was described. By studying the S-box of Serpent1, the modular addition, and the linear feedback shift register, a kind of equation between the bits of the internal states was obtained, based on which an improved word-oriented guess-and-determine attack on Sosemanuk is proposed. This improved attack utilizes the Groebner basis method to solve the equation system and can determine the other 5 words of the internal state by guessing just 7 words, and thus the complexity is pvoved to be $O(2192)$.

Key words: eSTREAM project; Sosemanuk; guess-and-determine attack; Groebner basis

Sosemanuk 算法[2]是欧洲 eSTREAM 计划[1]最终获选的七个候选算法之一。Sosemanuk 算法设计思想源于流密码 SNOW2.0[3] 和分组密码 Serpent[4]，是一种新型的面向软件的同步流密码，该算法的密钥长度在 128 和 256 比特之间。

猜测决定攻击是一种很常用分析流密码的方法。其分析过程分为三个过程：首先猜测部分内部状态，其次由猜测的内部状态决定出其余的内部状态，最后根据密钥流检验猜测内部状态的正确性。猜测决定攻击已被应用到很多流密码算法的分析中[3,5-6,12-13]。Sosemanuk 算法的设计者给出一种非常粗略的基于字的猜测决定攻击方法[2]，其复杂度为 $O(2^{256})$。文献[7-8]找到了不同的限制关系式，给出了更精确的基于字的猜测决定攻击的方法，其复杂度分别为 $O(2^{226})$ 和 $O(2^{224})$。文献[9,14]都给出了复杂度为 $O(2^{192})$ 的猜测决定攻击，但是他们在分析过程都忽略了同样的问题：Sosemanuk 算法中 Serpent1 要求一次处理连续 4 个字（共 128 比特），因此，攻击的复杂度达不到 $O(2^{192})$。文献[10]在基于字节分析的基础上给出了复杂度为 $O(2^{176})$ 的猜测决定攻击，这是目前已知最好的分析结果。

本文分析了 Sosemanuk 算法内部状态在比特层面上的关系式，通过对 Serpent1 组件 S 盒、模 2^{32} 加法以及线性反馈移位寄存器的研究，找到了关于算法内部状态的一个方程组。在此基础上，攻击者只需猜测一个内部状态，利用 Grobner 基方法就可以完全确定出另外一个内部状态。从而使得基于字的猜测决定攻击方法需要猜测的内部状态减少 32 比特，攻击的复杂度达到 $O(2^{192})$。

1 Sosemanuk 算法简介

Sosemanuk 算法由 $GF(2^{32})$ 上的线性反馈移位寄存器（LFSR）、有限状态机（FSW）和 Serpent1 组成。其原理结构如图 1（⊕代表模 2 加，⊞代表模 2^{32}）。

* **收稿日期**：2012-03-22
基金项目：国家自然科学基金资助项目（61070215，61103192）
作者简介：谢端强（1966—），男，教授，硕士生导师，E-mail：dqxie001@163.com

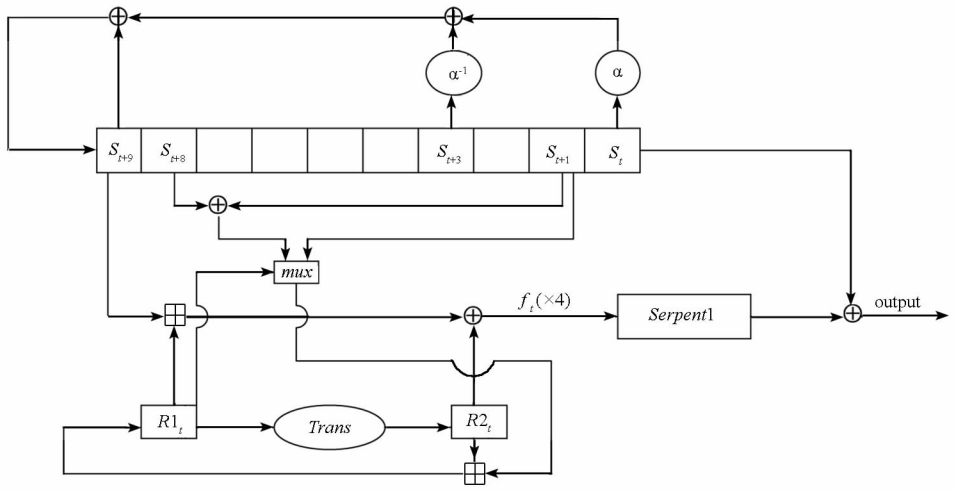

图 1 Sosemanuk 算法流程图

Fig. 1 The structure of Sosemanuk

1.1 线性反馈移位寄存器

线性反馈移位寄存器的反馈多项式为

$$\pi(X)=\alpha X^{10}+\alpha^{-1}X^{17}+X+1\in F_{2^{32}}[X] \quad (1)$$

其中 α 是如下本元多项式的根：

$$P(X)=X^4+\beta^{23}X^3+\beta^{245}X^2+\beta^{48}+\beta^{239}\in F_{2^8}[X]$$

其中 β 是如下本元多项式的根：

$$Q(X)=X^8+X^7+X^5+X^3+1\in F_2[X]$$

线性反馈移位寄存器的更新函数为

$$s_{t+10}=s_{t+9}+\alpha^{-1}s_{t+3}+\alpha s_t,\ \forall t\geqslant 1 \quad (2)$$

1.2 有限状态机

FSM 包含两个 32 比特的存储器 $R1$ 和 $R2$。每一步将线性反馈寄存器中的 $(s_{t+1},s_{t+8},s_{t+9})$ 作为输入，再输出一个 32 比特的字。对于 $t\geqslant 1$，更新过程如下

$$(R1_{t-1},R2_{t-1},s_{t+1},s_{t+8},s_{t+9})\rightarrow(R1_t,R2_t,f_t) \quad (3)$$

$$R1_t=(R2_{t-1}+\mathrm{mux}(lsb(R1_{t-1}),s_{t+1},s_{t+1}+s_{t+8}))\bmod 2^{32} \quad (4)$$

$$R2_t=\mathrm{Trans}(R1_{t-1}) \quad (5)$$

$$f_t=(s_{t+9}+R1_t\bmod 2^{32})\oplus R2_t \quad (6)$$

1.3 输出函数

Sosemanuk 算法的输出函数用到了 Serpent1 函数。Serpent1 函数是分组密码 Serpent 的一轮输出函数，以 bit-slice 模式运算，最终输出的密钥流 z_t，$t\geqslant 1$：

$$(z_{t+3},z_{t+2},z_{t+1},z_t)=Serpent1(f_{t+3},f_{t+2},f_{t+1},f_t)\oplus(s_{t+3},s_{t+2},s_{t+1},s_t) \quad (7)$$

2 Sosemanuk 算法的组件结构分析

Sosemanuk 算法的设计吸收了 serpent 算法和 Snow2.0 算法的优点。与 Snow2.0 算法相比，Sosemanuk 算法减少内部状态来达到其加解密的效率。算法中 Serpent1 组件通过并列的 32 个 S 盒、以 bit-slice 模式，每次处理连续 4 个字。Serpent1 组件对其安全性有至关重要的影响。所以要在比特的关系式上对 Serpent1 进行分析。

2.1 Sosemanuk 算法的代数结构

$GF(2^{32})$ 是在 $GF(2)$ 上先由多项式 $Q(X)$ 进行扩张，此时 $\{1,\beta,\beta^2,\cdots,\beta^7\}$ 为其一组基。在此扩域上再由多项式 $P(X)$ 进行扩张，此时 $\{1,\alpha,\alpha^2\alpha^3\}$ 为其一组基。于是 $\forall\ X\in GF(2^{32})$，X 可以有以下几种表示方法：

(1)向量表示：

$$X=(x_{31},x_{30},\cdots,x_1,x_0),$$

其中 $x_i,i=0,1,\cdots,31$ 为 X 的第 i 比特；

(2)字节表示：

$$X=X^{(3)}\parallel X^{(2)}\parallel X^{(1)}\parallel X^{(0)},$$

其中 $X^{(i)},i=0,1,2,3$ 表示 8 个比特的字节；

(3)基表示：

$$\begin{aligned}X=&(x_{31}\beta^7+x_{30}\beta^6+\cdots+x_{24})\alpha^3\\&+(x_{23}\beta^7+x_{22}\beta^6+\cdots+x_{16})\alpha^2\\&+(x_{15}\beta^7+x_{14}\beta^6+\cdots+x_8)\alpha^1\\&+(x_7\beta^7+x_6\beta^6+\cdots+x_0)\end{aligned}$$

2.2 Sosemanuk 算法在比特层面上的表达式

Sosemanuk 算法是处理字的加密方法。文献[10]给出了其在字节层面上的表达形式，本文给出其在比特层面上的表达形式。在线性反馈移位寄存器的更新函数中出现了 $\alpha^{-1}X$ 和 αX 这两项，现在我们来讨论它们该如何表示。首先讨论 $\alpha^{-1}X$ 的表达形式。

由于 α 是多项式 $P(X)$ 的根,所以有

$$\alpha^4+\beta^{23}\alpha^3+\beta^{245}\alpha+\beta^{239}=0$$

经过变换可得

$$\alpha^{-1}=\beta^{16}\alpha^3+\beta^{39}\alpha^2+\beta^6\alpha+\beta^{64}$$

设 $X=(c_{31},c_{30},\cdots,c_1,c_0)$,$\alpha^{-1}X=(c'_{31},c'_{30},\cdots,c'_1,c'_0)$。故

$$\begin{aligned}\alpha^{-1}X&=\alpha^{-1}(X^{(3)}\alpha^3+X^{(2)}\alpha^2+X^{(1)}\alpha+X^{(0)})\\&=\beta^{16}\Big(\sum_{i=0}^{7}c_i\beta^i\Big)\alpha^3+\Big(\sum_{i=24}^{31}c_i\beta^i+\beta^{39}\sum_{i=0}^{7}c_i\beta^i\Big)\alpha^2\\&+\Big(\sum_{i=16}^{23}c_i\beta^i+\beta^6\sum_{i=0}^{7}c_i\beta^i\Big)\alpha\\&+\Big(\sum_{i=8}^{15}c_i\beta^i+\beta^{64}\sum_{i=0}^{7}c_i\beta^i\Big)\\&\triangleq\Big(\sum_{i=0}^{7}c'_{24+i}\beta^i\Big)\alpha^3+\Big(\sum_{i=0}^{7}c'_{16+i}\beta^i\Big)\alpha^2\\&+\Big(\sum_{i=0}^{7}c'_{8+i}\beta^i\Big)\alpha+\Big(\sum_{i=0}^{7}c'_i\beta^i\Big)\end{aligned}$$

其中

$$\begin{aligned}\beta^{16}\Big(\sum_{i=0}^{7}c_i\beta^i\Big)&=c_0\beta^{16}+c_1\beta^{17}+c_2\beta^{18}+c_3\beta^{19}\\&+c_4\beta^{20}+c_5\beta^{21}+c_6\beta^{22}+c_7\beta^{23}\\&=(c_7+c_6+c_3)\beta^7+(c_7+c_5+c_3+c_2)\beta^6\\&+(c_7+c_6+c_4+c_2+c_1)\beta^5+(c_5+c_1+c_0)\beta^4\\&+(c_4+c_0)\beta^3+c_6\beta^2+c_5\beta+(c_7+c_4)\end{aligned}$$

所以有

$$\begin{aligned}&c'_{31}=c_7+c_6+c_3;\ c'_{30}=c_7+c_5+c_3+c_2;\\&c'_{29}=c_7+c_6+c_4+c_2+c_1;\ c'_{28}=c_5+c_1+c_0;\\&c'_{27}=c_4+c_0;\ c'_{26}=c_6;\\&c'_{25}=c_5;c'_{24}=c_7+c_4。\end{aligned}$$

同理可得 $\alpha^{-1}X$ 的每个比特的具体表达形式,如表 1。

按照同样的方法很容易得出 αX 的每个比特的具体表达形式。

表 1　$\alpha^{-1}X$ 的每个比特的表达式

Tab. 1　The expression of each bit in $\alpha^{-1}X$

位置	c'_i 取值	位置	c'_i 取值
$i=0$	$c_8+c_7+c_4+c_1+c_0$	$i=16$	$c_{24}+c_7+c_5+c_0$
$i=1$	$c_9+c_5+c_2+c_1$	$i=17$	$c_{25}+c_6+c_1+c_0$
$i=2$	$c_{10}+c_6+c_3+c_2+c_0$	$i=18$	$c_{26}+c_7+c_2+c_1+c_0$
$i=3$	$c_{11}+c_3+c_1$	$i=19$	$c_{27}+c_7+c_5+c_3+c_2+c_1+c_0$
$i=4$	$c_{12}+c_4+c_1$	$i=20$	$c_{28}+c_6+c_4+c_3+c_2+c_1$
$i=5$	$c_{13}+c_7+c_5+c_4+c_2+c_1$	$i=21$	$c_{29}+c_4+c_3+c_2$
$i=6$	$c_{14}+c_6+c_5+c_3+c_2+c_0$	$i=22$	$c_{30}+c_5+c_4+c_3$
$i=7$	$c_{15}+c_6+c_3+c_0$	$i=23$	$c_{31}+c_7+c_6+c_4$
$i=8$	$c_{16}+c_7+c_6+c_4+c_3+c_2$	$i=24$	c_7+c_4
$i=9$	$c_{17}+c_7+c_5+c_4+c_3$	$i=25$	c_5
$i=10$	$c_{18}+c_6+c_5+c_4$	$i=26$	c_6
$i=11$	$c_{19}+c_4+c_5+c_3+c_2$	$i=27$	c_4+c_0
$i=12$	$c_{20}+c_6+c_4+c_5+c_3$	$i=28$	$c_5+c_1+c_0$
$i=13$	$c_{21}+c_5+c_3+c_2$	$i=29$	$c_7+c_6+c_4+c_2+c_1$
$i=14$	$c_{22}+c_6+c_4+c_3+c_0$	$i=30$	$c_7+c_5+c_3+c_2$
$i=15$	$c_{23}+c_6+c_5+c_3+c_2+c_1$	$i=31$	$c_7+c_6+c_3$

2.3　Serpent1 函数中 S 盒的性质

Serpent1 中 S 盒定义[2]为:$S(x)=\{8,6,7,9,3,12,10,15,13,1,14,4,0,11,5,2\}$. 不妨设进入 S 盒的四个比特分别为 x_3,x_2,x_1,x_0,经过 S 盒后的 4 个比特记为:y_3,y_2,y_1,y_0。由 S 盒的定义可以得到以下布尔函数:

$$\begin{cases}x_0=y_0+y_1+y_2+y_1y_2+y_1y_3\\x_1=y_1+y_0y_1+y_2+y_0y_3+y_0y_1y_3\\\qquad+y_2y_3+y_0y_2y_3\\x_2=y_0+y_0y_1+y_2+y_3+y_0y_3+y_1y_3\\\qquad+y_0y_1y_3+y_0y_2y_3+1\\x_3=y_0y_1+y_1y_2+y_0y_1y_2+y_3\\\qquad+y_0y_2y_3+1\end{cases}\tag{8}$$

可以看出 x_0 是关于 y_3, y_2, y_1, y_0 的一个 2 次布尔函数。

2.4 运算⊞的展开式

设 $X, Y, Z \in GF(2^{32})$，其中 $X=(x_{31}, \cdots, x_0)$，$Y=(y_{31}, \cdots, y_0)$，$Z=(z_{31}, \cdots, z_0)$，则 $Z = X \boxplus Y$ 按照分量可以展开为[11]

$$\begin{cases} z_0 = x_0 + y_0 \\ z_1 = x_1 + y_1 + x_0 y_0 \\ z_2 = x_2 + y_2 + x_1 y_1 + (x_1 + y_1)(x_1 + y_1 + z_1) \\ \quad \vdots \\ z_i = x_i + y_i + x_{i-1} y_{i-1} + (x_{i-1} + y_{i-1})(x_{i-1} + y_{i-1} + z_{i-1}) \\ \quad \vdots \\ z_{31} = x_{31} + y_{31} + x_{30} y_{30} + (x_{30} + y_{30})(x_{30} + y_{30} + z_{30}) \end{cases} \tag{9}$$

3 对 Sosemanuk 算法的猜测决定攻击

从对 Sosemanuk 算法已有的面向字的猜测决定分析的文献[2,7-9,14]中可以看出，猜测决定过程一般分为两个部分：第一部分至少猜测 6 个字才可以决定出 s_5、s_6 和 s_{10}，但是决定不出 s_7、s_8、s_9；第二部分还需在 s_7、s_8 和 s_9 中至少猜测两个字才能决定出另外一个状态。这样分析的复杂度就会很高。本文在此基础上进行了改进，其中第一部分的分析过程与已有的分析过程类似，具体可见文献[8]；第二部分运用第 2 节给出的性质，只需在 s_7、s_8 和 s_9 中猜测 s_8 一个字，就能找到相应的状态方程，并利用 Grobner 基等方法决定出其余的两个状态。

首先假设 $lsb(R1_0)=0$，于是（4）式可以简化为

$$R1_1 = R2_0 \oplus s_2 \tag{4'}$$

对 Sosemanuk 算法的猜测决定攻击过程如下

第一步　选取密钥流 z_1, z_2, z_3, z_4，猜测 $s_1, s_2, s_3, s_4, R1_0, R2_0$，决定 s_5, s_6, s_{10}

首先，可以按照以下步骤决定出 $f_1, f_2, f_3, f_4, R1_1, R1_2, R2_1, R2_2, s_{10}, s_{11}$。

$$\{z_4 \oplus s_4, z_3 \oplus s_3, z_2 \oplus s_2, z_1 \oplus s_1\} \xrightarrow{(7)} \{f_4, f_3, f_2, f_1\}$$

$$\{R2_0, s_2\} \xrightarrow{(4')} R1_1,\ R1_0 \xrightarrow{(5)} R2_1$$

$$\{f_1, R1_1, R2_1\} \xrightarrow{(6)} s_{10},\ \{s_{10}, s_4, s_1\} \xrightarrow{(5)} s_{11}$$

$$\{R2_1, R1_1, s_3, s_{11}\} \xrightarrow{(4')} R1_2,\ R1_1 \xrightarrow{(5')} R2_2$$

但是另一方面，

$$f_2 = (s_{11} \boxplus R1_2) \oplus R2_2 \tag{*}$$

这里 $f_2, s_{11}, R1_2, R2_2$ 都可以由猜测的字决定出来，并且它们还要满足此方程，所以此时可以使初始猜测状态空间降低 2^{-32}。

下面可以决定出：$s_5, s_6, s_{12}, s_{13}, R1_3, R2_3, R1_4, R2_4, R1_5, R2_5, R1_6$。

$$\{R2_2, R1_2, s_4, s_{11}\} \xrightarrow{(4)} R1_3,\ R1_2 \xrightarrow{(5)} R2_3$$

$$\{f_3, R1_3, R2_3\} \xrightarrow{(6)} s_{12},\ \{s_{12}, s_{11}, s_2\} \xrightarrow{(2)} s_5$$

$$\{R2_3, R1_3, s_5, s_{12}\} \xrightarrow{(4)} R1_4,\ R1_3 \xrightarrow{(5)} R2_4$$

$$\{f_4, R1_4, R2_4\} \xrightarrow{(6)} s_{13},\ \{s_{13}, s_{12}, s_3\} \xrightarrow{(2)} s_6$$

$$\{R2_4, R1_4, s_6, s_{13}\} \xrightarrow{(4)} R1_5,\ R1_4 \xrightarrow{(5)} R2_5$$

$$R1_5 \xrightarrow{(5)} R2_6$$

第二步　选取密钥流 z_5, z_6, z_7, z_8，猜测 s_8，决定 s_7

根据 Sosemanuk 算法的更新过程可得

$$\begin{cases} s_{14} = s_{13} \oplus \alpha^{-1} s_7 \oplus \alpha s_4 \\ f_5 = (s_{14} \boxplus R1_5) \oplus R2_5 \end{cases}$$

即　$f_5 = ((s_{13} \oplus \alpha^{-1} s_7 \oplus \alpha s_4) \boxplus R1_5) \oplus R2_5$ (**)

另一方面，可以选取密钥流 z_5, z_6, z_7, z_8，根据 Serpent1 中 S 盒的性质得到如下关系式：

$$(f_5)_i = m_i + n_i + (c_i + e_i) + n_i(c_i + e_i) + n_i o_i,\quad i = 0, \cdots, 31 \tag{***}$$

其中

$M = (m_{31}, m_{30}, \cdots, m_1, m_0) = s_5 \oplus z_5$，

$N = (n_{31}, n_{30}, \cdots, n_1, n_0) = s_6 \oplus z_6$，

$s_7 = (c_{31}, c_{30}, \cdots, c_1, c_0), z_7 = (e_{31}, \cdots, e_1, e_0)$，

$O = (o_{31}, o_{30}, \cdots, o_1, o_0) = s_8 \oplus z_8$。

设 $R2_5 = (d_{31}, \cdots, d_1, d_0), R1_5 = (b_{31}, \cdots, b_1, b_0)$，$s_{13} \oplus \alpha s_4 = (a_{31}, \cdots, a_1, a_0), n_i + 1 \triangleq \bar{n}_i$，$g_i = m_i + n_i + e_i + n_i e_i + n_i o_i + d_i$，$0 \leqslant i \leqslant 31$ 根据(**)和(***)式，消去$(f_5)_i$得

$$((s_{13} \oplus \alpha^{-1} s_7 \oplus \alpha s_4) \boxplus R1_5)_i \oplus (R2_5)_i = m_i + n_i + (c_i + e_i) + n_i(c_i + e_i) + n_i o_i, i = 0, \cdots, 31$$

即　$X \boxplus R1_5 = \bar{Z}$ (10)

其中 $x_i = a_i + c_i', \bar{z}_i = m_i + n_i + (c_i + e_i) + n_i(c_i + e_i) + n_i o_i + o_i = g_i + c_i \bar{n}_i, i = 0, \cdots, 31$。

再由第 2.4 节中关于⊞运算(9)的分解形式可以把(10)式按照比特展开得到

$$\begin{cases} g_0 + c_0 \bar{n}_0 = c_0' + a_0 + b_0, \\ g_1 + c_1 \bar{n}_1 = c_1' + a_1 + b_1 + (c_0' + a_0) b_0, \\ g_2 + c_2 \bar{n}_2 = c_2' + a_2 + b_2 + (c_1' + a_1) b_1 + (c_1' + a_1 + b_1)(c_1' + a_1 + b_1 + g_1 + c_1 \bar{n}_1), \\ \quad \vdots \\ g_i + c_i \bar{n}_i = c_i' + a_i + b_i + (c_{i-1}' + a_{i-1}) b + (c_{i-1}' + a_{i-1} + b_{i-1})(c_{i-1}' + a_{i-1} + b_{i-1} + g_{i-1} + c_{i-1} \bar{n}_{i-1}), \\ \quad \vdots \\ g_{31} + c_{31} \bar{n}_{31} = c_{31}' + a_{31} + b_{31} + (c_{31}' + a_{31}) b_{31} + (c_{31}' + a_{31} + b_{31})(c_{31}' + a_{31} + b_{31} + g_{31} + c_{31} \bar{n}_{31}). \end{cases} \tag{11}$$

对此方程组而言，$a_i, b_i, g_i, \bar{n}_i$，$i=0,\cdots,31$ 都为已知，c_i'即为前面表 1 中数据，且方程的最高次数为 2 次，根据 Groebner 基方法[15]可以解出 c_i 的值，即确定 s_7 的值。

第三步　决定 s_9

再由以下关系式即可确定 s_9 的值。

$$\{R2_5, R1_5, s_7, s_{14}\} \xrightarrow{(4)} R1_6,\ \{f_6, R1_6, R2_6\} \xrightarrow{(6)} s_{15}$$

$$\{R2_6, R1_6, s_8, s_{15}\} \xrightarrow{(4)} R1_7,\ R1_6 \xrightarrow{(5)} R1_7$$

$$\{f_7, R1_7, R2_7\} \xrightarrow{(6)} s_{16},\ \{s_{16}, s_{15}, s_6\} \xrightarrow{(2)} s_9$$

到此已经把 $s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, R1_1, R2_1$ 这 12 个连续的中间状态完全确定下来了。

4　攻击的复杂度

猜测之前我们假设 $lsb(R1_0)=0$，此式以 1/2 的概率成立。在第一步中需要猜测 191 比特经过等式(*)限制后可以淘汰 2^{-32}的猜测空间。在第二步中又需要猜测 32 个比特的位置。故其复杂度：

$$T = 2^{191} \times 2^{-32} \times 2^{1} \times 2^{32} = 2^{192}。$$

另外，在猜测决定阶段需要 8 个字的密钥流，在测试阶段还需要 8 个字的密钥流，所以此分析方法共需要 16 个字的密钥流。

对 Sosemanuk 算法的猜测决定攻击分为两类，一类是面向字的，另一类是面向字节的。对此算法的分析已有的结果见表 2，可知本文的结果在面向字的猜测决定攻击中是最优的。

5　结束语

本文改进了对 Sosemanuk 算法基于字的猜测决定攻击，攻击复杂度达到了 $O(2^{192})$，这在基于字的猜测决定攻击中是最优的。本文首先分析了 Sosemanuk 算法中内部状态在比特层面上的关系式；接着对此算法的 Serpent1 组件中 S 盒的性质进行了分析；最后结合模 2^{32}加法在比特层面上的具体展开形式，找到了关于线性反馈移位寄存器内部状态的一个方程组。这样，在猜测一个内部状态的前提下，利用 Groebner 基方法可以完全确定出另外一个内部状态。从而使得需要猜测的内部状态减少 32 比特。下一步将考虑如何基于本文所给的方法来进一步提高对 Sosemanuk 算法基于字节的猜测决定攻击效率。

表 2　Sosemanuk 算法已有的猜测决定分析结果比较

Tab. 2　The results of guess-and-determine attack on sosemanuk

	基于字(32 比特)的猜测决定攻击				基于字节(8 比特)的猜测决定攻击
文献	[2]	[7]	[8]	本文结果	[10]
复杂度	$O(2^{256})$	$O(2^{226})$	$O(2^{224})$	$O(2^{192})$	$O(2^{176})$

参考文献(References)

[1] The eSTREAM project [EB/OL]. [2012-2-11]. http://www.ecrypt.eu.org/stream/.

[2] Berbain C, Billet O, Canteaut A, et al. SOSEMANUK: A fastSoftware-oriented stream cipher[M]//New Stream Cipher Designs: The eSTREAM Finalists. Berlin, Germany: Springer-Verlag, 2008.

[3] Ekdahl P, Johansson T. A new version of the stream cipher SNOW[C]//Proceedings of SAC'03. Berlin, Germany: Springer-Verlag, 2003.

[4] Biham E, Anderson R, Knudsen L. SERPENT: A new block cipher proposal[C]//Proceedings of FSE'98, LNCS 1372, Springer, 1998: 222-238.

[5] Hawks P, Rose G. Guess and determine attacks on SNOW [C]// Proceedings of SAC 2002, LNCS 2595, Springer, 2003: 37-46.

[6] Cannier C D. Guess and determine attacks on SNOW[EB/OL]. [2012-02-011]. http://www.cryptonessie.org.

[7] Ahmadi H, Eghlidos T, Khazaei S. Impr-oved guess and determine attack on SOSEMANUK[EB/OL]. [2012-02-11]. http://www.ecrypt.eu.org/stream/papersdir/085.pdf.

[8] Tsunoo Y, Saito T, Shigeri M, et al. Evaluation of SOSEMANUK with regard to guess and determine attacks[EB/OL]. [2012-01-02]. http://www.ecrypt.eu.org/stream/papersdir/2006/009.pdf.

[9] Ding L, Guan J. Guess and determine attack on SOSEMANUK[C]//Proceedings of ISA'09, Xi'an, China, 2009.

[10] Feng X, Liu J, et al. A Byte-based guess and determine attack on SOSEMANUK [C]//Proceedings of ASIACRYPT 2010, LNCS 6477, Springer, 2010: 146-157.

[11] Courtois N, Debraize B. Algebraic description and simultaneous linear approximations of addition in snow 2.0 [C]//Proceedings of ICICS, LNCS 5308, Springer, 2008: 146-157.

[12] Lu Y, Wang H, Ling S. Cryptanalysis of rabbit [C]// Proceedings of ISC2008, LNCS 5222, Springer, 2008: 204-214.

[13] Abdelraheem M A, Borghoff J, et al. Cryptanalysis of the iight-weight cipher A2U2[C]// Proceedings of Cryptography and Coding 2011, LNCS 7089, Springer, 2011: 375-390.

[14] 张海霞，胡予濮，柴进. 针对 SOSEMANUK 的猜测-确定攻击[J]. 计算机工程，2011，37(4)，170-171.
ZHANG Haixia, HU Yupu, CAI Jin. Guess and determine attack on SOSEM-ANUK [J]. Computer Engineering, 2011, 37(4), 170-171. (in Chinese)

[15] 刘木兰. Groebner 基理论及其应用[M]. 北京：科学出版社，2000，1(3)：159-207.
LIU Mulan. Groebner basis and its applications [M]. Beijing: Science Press, 2000, 1(3): 159-207. (in Chinese)