

祖冲之算法的安全分析

杜红红¹, 张文英^{1,2,3}

- (1. 山东师范大学 信息科学与工程学院, 山东 济南 250014;
2. 中国科学院研究生院信息安全国家重点实验室, 北京 100049;
3. 山东分布式软件新技术重点实验室, 山东 济南 250014)

摘要:分析了面向字序列的流密码体制祖冲之算法(ZUC)的安全性。ZUC算法使用128比特的初始密钥和128比特的初始向量,生成32比特的密钥字序列。其整体结构由线性反馈移位寄存器(LFSR)、比特重组(BR)和非线性函数F组成。文中先猜一部分内部单元,然后去推导剩余的,从而得知密钥生成所用的全部内部单元。其穷尽搜索复杂度为 $O(2^{128})$,文中所讲述的方法复杂度为 $O(2^{126})$ 。因此减少了搜索复杂度。实际应用的ZUC算法中前32步只用来初始化,没有生成密钥,而且丢弃第33步的结果。而文中是让算法初始化两步,从第三步开始进行密钥流的输出,进而说明ZUC算法的安全性,这也说明ZUC在前33步不输出密钥的初衷是为了增进安全性,避免受到此类攻击。

关键词:祖冲之算法;流密码;安全性

中图分类号:TP309

文献标识码:A

文章编号:1673-629X(2012)06-0151-05

Security Analysis on ZUC Stream Cipher

DU Hong-hong¹, ZHANG Wen-ying^{1,2,3}

- (1. School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China;
2. State Key Laboratory of Information Security, Graduate University, Chinese Academy of Sciences, Beijing 100049, China;
3. Shandong Provincial Key Laboratory for Distributed Computer Software Novel Technology, Jinan 250014, China)

Abstract: ZUC is a word-oriented stream cipher. It uses a 128-bit initial key and a 128-bit initial vector to produce a keystream of 32-bit words named by ancient mathematician Zu Chongzhi. ZUC consists of a linear feedback shift register (LFSR), bit-reorganization (BR) and a nonlinear function F. In this paper it first guesses some of the internal states and deduces the others. The time complexity is $O(2^{126})$, which is far from $O(2^{128})$, the complexity of the exhaustive key search. The first 32 steps of ZUC is the initialization stage and it doesn't produce keystream. And the step 33 discards its result. But this work lets it initial 2 steps and produce keystream from the step three. So it proves the security of the algorithm and avoids the attack of this type.

Key words: ZUC; stream cipher; security

0 引言

2004年,3GPP(3rd Generation Partnership Project)启动长期演进计划(LTE)的研究,即4G国际通信标准。由我国自主设计的加密算法128-EEA3和完整性算法128-EIA3也参与了LTE通信加密标准的申报工作^[1]。上述两个算法的核心是祖冲之算法(ZUC)^[2],ZUC算法由中国科学院数据保护和通信安全研究中

心(DACAS)研制,被3GPP初步确定为LTE国际标准。现已在9月的SA#53会议上被正式通过成为国际标准,这是第一个成为国际标准的我国自主研发的密码算法。对我国按照国际惯例掌握通信产业的主动权有非常重要的意义。文中受文献[3]中密码分析方法的启发,用Guess and Determine密码分析方法^[4-9]分析了ZUC算法的安全性。先猜一部分内部单元,然后去推导剩余的。之后用这些已知量生成密钥来检测与原密钥是否相同,若相同则说明所猜的是对的,否则需要重新搜索^[10,11]。其穷尽搜索复杂度为 2^{128} ,文中的方法复杂度为 2^{126} 。实际应用的ZUC算法中前32步只用来初始化,并丢弃第33步的结果。而文中是让算法只初始化2步。也就是若ZUC算法从第3步开始产生密钥,那它就不安全了,进而说明ZUC算法的

收稿日期:2011-11-03;修回日期:2012-02-08

基金项目:山东省自然科学基金(Y2008G01);山东省高等学校优秀青年教师国内访问学者项目;山东省高等学校基于身份的特殊数字签名的应用研究(ZR2011FQ032)

作者简介:杜红红(1988-),女,硕士生,研究领域为流密码;张文英,教授,硕士生导师,通讯作者,主要研究领域为布尔函数、密码分析。

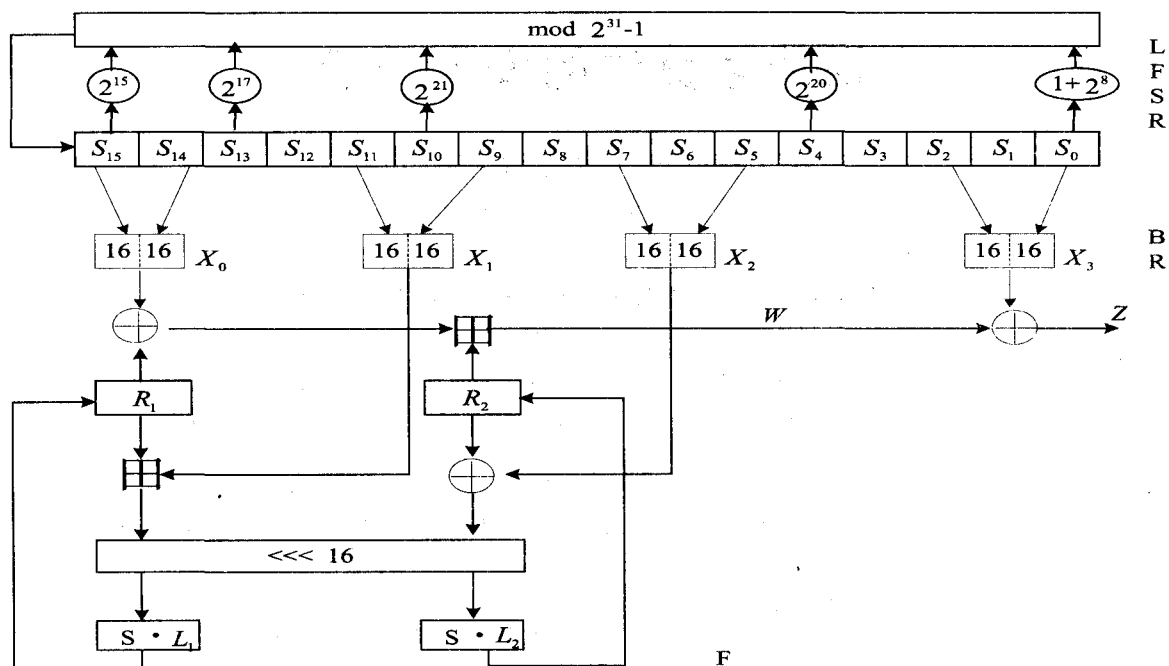


图 1 ZUC 的整体结构

安全性。这也说明 ZUC 在前 33 步不输出密钥的初衷是为了增进安全性,避免受到此类攻击^[12]。

1 ZUC 算法简介

1.1 ZUC 结构简介

ZUC 算法的输入为 128 比特的初始密钥和 128 比特的初始向量,输出为 32 比特的密钥字序列。其整体结构如图 1 所示,由线性反馈移位寄存器(LFSR)、比特重组(BR)和非线性函数 F 组成。

1.1.1 线性反馈移位寄存器(LFSR)

LFSR 由 16 个 31 位的寄存器($S_0, S_1, \dots, S_{14}, S_{15}$)组成,每一个都是定义在素域 $GF(2^{31} - 1)$ 上。LFSR 有 2 种状态:初始化状态和工作状态。详细步骤为:

LFSRWithInitialisationMode(u) {

1. $v = 2^{15}S_{15} + 2^{17}S_{13} + 2^{21}S_{10} + 2^{20}S_4 + (1 + 2^8)S_0 \bmod(2^{31} - 1)$;

2. $S_{16} = (v + u) \bmod(2^{31} - 1)$; // $u = W > 1, W$ 为非线性函数 F 的输出;

3. If $S_{16} = 0$, then set $S_{16} = 2^{31} - 1$;

4. $(S_1, S_2, \dots, S_{15}, S_{16}) \rightarrow (S_0, S_1, \dots, S_{14}, S_{15})$ 。

}

LFSRWithWorkMode() {

1. $S_{16} = 2^{15}S_{15} + 2^{17}S_{13} + 2^{21}S_{10} + 2^{20}S_4 + (1 + 2^8)S_0 \bmod(2^{31} - 1)$;

2. If $S_{16} = 0$, then set $S_{16} = 2^{31} - 1$;

3. $(S_1, S_2, \dots, S_{15}, S_{16}) \rightarrow (S_0, S_1, \dots, S_{14}, S_{15})$ 。

1.1.2 比特重组(BR)

比特重组是一个过渡层,其主要从 LFSR 的 8 个寄存器单元抽取 128 比特内容组成 4 个 32 比特的字,以供下层非线性函数 F 和密钥输出使用。详细步骤为:

Bitreorganization() {

1. $X_0 = S_{15H} || S_{14L}$;

2. $X_1 = S_{11L} || S_{9H}$;

3. $X_2 = S_{7L} || S_{5H}$; 4. $X_3 = S_{2L} || S_{0H}$ 。

}

1.1.3 非线性函数 F

F 有 2 个 32 位的存储单元 R_1, R_2 , 输入为 X_0, X_1, X_2 , 输出为 32 位的字 W 。详细步骤为:

$F(X_0, X_1, X_2)$ {

1. $W = (X_0 \oplus R_1) \boxtimes R_2$;

2. $W_1 = R_1 \boxtimes X_1$;

3. $W_2 = R_2 \oplus X_2$;

4. $R_1 = S(L_1(W_{1L} || W_{2H}))$; // $L_1(X) = X \oplus (X \ll 32) \oplus (X \ll 10) \oplus (X \ll 18) \oplus (X \ll 24)$;

5. $R_2 = S(L_2(W_{2L} || W_{1H}))$; // $L_2(X) = X \oplus (X \ll 32) \oplus (X \ll 8) \oplus (X \ll 14) \oplus (X \ll 22) \oplus (X \ll 30)$;

// 下标 32 表示 X 是 32 比特的数; S 为 S 盒运算。

}

1.2 密钥封装

LFSR 的 $S_i = k_i || d_i || iv_i (0 \leq i \leq 15)$, 每个 S_i 含 8 比特 key, 8 比特 IV, 15 比特 D , S_i 长度为 31 比特。其中 k 为 128 比特的初始密钥, iv 为 128 比特的初始向量, D 为已知的 240 比特的长整型常量字符串。 k, iv, D 分别被表示成 16 个字串级联的形式, $k = k_0 || \dots || k_{15}$, $iv = iv_0 || \dots || iv_{15}$, $D = d_0 || \dots || d_{15}$ 。

1.3 ZUC 的运行

ZUC 的运行过程分为初始化阶段和工作阶段。

1.3.1 初始化阶段

首先用密钥重载方法将 LFSR 进行初始状态载入, R_1, R_2 也初始化为全 0。初始化阶段会将下面的操作运行 32 轮。

1. Bitreorganization();
2. $w = F(X_0, X_1, X_2)$;
3. LFSRWithInitialisationMode($w \gg 1$)。

1.3.2 工作阶段

工作阶段需要先将下面的操作运行一轮。

1. Bitreorganization();
2. $F(X_0, X_1, X_2)$; // 此处丢弃输出结果;
3. LFSRWithWorkMode()。

然后就可以生成密钥流了, 将下面的操作运行一次就会生成一个 32 比特密钥 Z 。

1. Bitreorganization();
2. $Z = F(X_0, X_1, X_2) \oplus X_3$;
3. LFSRWithWorkMode()。

2 ZUC 安全分析过程

本节用 Guess and Determine 密码分析方法分析 ZUC 算法的安全性。

实际上 ZUC 算法的前 33 步 ($t=0$ 到 $t=32$) 都属于算法的初始化, 包括初始化阶段 ($t=0$ 到 $t=31$) 和工作阶段的第一步 ($t=32$)。但是假设 ZUC 算法初始化 2 步, 密钥流在第 3 步 ($t=2$) 开始输出。根据 Guess and Determine 密码分析方法, 文中找出的是第 5 时刻的全部内部单元, 包括 $S_0^5 \sim S_{15}^5, R_1^5, R_2^5$ (上标为 5 表示 $t=5$ 时刻), 共 $31 \times 16 + 32 \times 2 = 560$ 比特。其中算法的初始状态为 $S_i^0 = k_i || d_i || iv_i^0 (0 \leq i \leq 15)$ (上标为 0 表示 $t=0$ 时刻), d_i^0 为 16 个 15 比特的已知常量, iv_i^0 为 16 个 8 比特的已知向量, R_1^0, R_2^0 为已知的全 0 初值, 因此只有 k_i^0 为 16 个 8 比特的未知初始密钥。由 $(S_1, S_2, \dots, S_{15}, S_{16}) \rightarrow (S_0, S_1, \dots, S_{14}, S_{15})$ 可知 $S_0^5 \sim S_{10}^5$ 由 $S_0^5 \sim S_{15}^5$ 移位得到, 因此只有每一个的高 8 位的初始密钥为未知, 而 $S_{11}^5 \sim S_{15}^5$ 每一个的 31 位都为未知。

文中的分析主要分为两个阶段。第一个阶段是推

导 R_1^2 和 R_2^2 , 这是因为若直接猜测 R_1^2 和 R_2^2 需要猜测 64 比特, 而 R_1^0, R_2^0 为已知的全 0 初值, 若用其推导 R_1^2 和 R_2^2 则只需猜测 16 比特; 第二个阶段是从 $t=2$ 时刻 (密钥流的输出时刻) 利用已推导的 R_1^2 和 R_2^2 开始分析并找到 $t=5$ 时刻的全部内部单元。

2.1 R_1^2 和 R_2^2 的推导过程

由于 R_1^0 和 R_2^0 为已知的初值全 0, 因此可以首先通过猜 S_9^0 和 S_5^0 的高 8 位推导出 R_1^1 和 R_2^1 , 然后通过猜 $t=1$ 时刻 S_9^1 和 S_5^1 的高 8 位推导出 R_1^2 和 R_2^2 。

$t=0$ 时刻 R_1^1 和 R_2^1 的推导过程如下:

$$X_1^0 = S_{11L}^0 || S_{9H}^0 \quad X_2^0 = S_{7L}^0 || S_{5H}^0$$

已猜测 S_9^0 和 S_5^0 的高 8 位, 由于 S_9^0, S_5^0 的其余位和 S_{11L}^0, S_{7L}^0 均为已知的常量和 IV 向量, 故由这两个公式可求 X_1^0 和 X_2^0 。

$$W_1^0 = R_1^0 \boxplus X_1^0 \quad W_2^0 = R_2^0 \boxplus X_2^0$$

R_1^0 和 R_2^0 均为已知的初值全 0, X_1^0 和 X_2^0 已由上面公式推导出, 故由这两个公式可求 W_1^0 和 W_2^0 。

$$R_1^1 = S(L_1(W_{1L}^0 || W_{2H}^0)) \quad R_2^1 = S(L_2(W_{2L}^0 || W_{1H}^0))$$

W_1^0 和 W_2^0 已由上面公式推导出, 故由这两个公式可求 R_1^1 和 R_2^1 。

R_1^1 和 R_2^1 已经推导出, 同理 $t=1$ 时刻 R_1^2 和 R_2^2 的推导过程如下:

$$X_1^1 = S_{11L}^1 || S_{9H}^1 \quad X_2^1 = S_{7L}^1 || S_{5H}^1$$

已猜测 S_9^1 和 S_5^1 的高 8 位, 由于 S_9^1, S_5^1 的其余位和 S_{11L}^1, S_{7L}^1 均已知, 故由这两个公式可求 X_1^1 和 X_2^1 。

$$W_1^1 = R_1^1 \boxplus X_1^1 \quad W_2^1 = R_2^1 \boxplus X_2^1$$

R_1^1 和 R_2^1 已由 $t=0$ 时刻推导出, X_1^1 和 X_2^1 已由上面公式推导出, 故由这两个公式可求 W_1^1 和 W_2^1 。

$$R_1^2 = S(L_1(W_{1L}^1 || W_{2H}^1)) \quad R_2^2 = S(L_2(W_{2L}^1 || W_{1H}^1))$$

W_1^1 和 W_2^1 已由上面公式推导出, 故由这两个公式可求 R_1^2 和 R_2^2 。

至此已猜测 S_9^0, S_5^0, S_9^1 和 S_5^1 的高 8 位共 32 比特, 推导出 R_1^2 和 R_2^2 。由 $(S_1, S_2, \dots, S_{15}, S_{16}) \rightarrow (S_0, S_1, \dots, S_{14}, S_{15})$ 可以得出 $t=2$ 时刻 S_3^2, S_4^2, S_7^2 和 S_8^2 均已知。下面将从 $t=2$ 时刻开始进行分析。

2.2 分析过程

2.2.1 $t=2$ 时刻分析过程

$t=2$ 时刻的已知量为 $S_3^2, S_4^2, S_7^2, S_8^2, R_1^2$ 和 R_2^2 。在此时刻我们预猜测一部分内部单元, 然后由此推导出一部分。在此需要注意的是 S_{15}^2 和 S_{14}^2 已不是由 S_1^1 移位得到, 因此其 31 位均为未知。分析过程如下:

$$X_3^2 = S_{2L}^2 || S_{0H}^2$$

$$Z^2 = W^2 \boxplus X_3^2$$

$$W^2 = (X_0^2 \boxplus R_1^2) \boxplus R_2^2$$

$$X_0^2 = S_{15H}^2 || S_{14L}^2$$

预猜测 S_0^2 的高 8 位, 因 S_{0H}^2 的其余位和 S_{2L}^2 均已知, 故可求 X_0^2 。将 X_0^2 代入第二个公式, 因 Z^2 为已知密钥, 故可求 W^2 。将 W^2 代入第三个公式, 因 R_1^2 和 R_2^2 均已知, 故可求 X_0^2 。将 X_0^2 代入第四个公式即可求 S_{15H}^2 和 S_{14L}^2 。

$$X_1^2 = S_{11L}^2 || S_{9H}^2$$

$$W_1^2 = R_1^2 \oplus X_1^2$$

预猜测 S_9^2 高 8 位, 因 S_{9H}^2 的其余位和 S_{11L}^2 均已知, 故可求 X_1^2 。将 X_1^2 代入第二个公式, 因 R_1^2 为已知, 故可求 W_1^2 。

$$X_2^2 = S_{7L}^2 || S_{5H}^2$$

$$W_2^2 = R_2^2 \oplus X_2^2$$

同理预猜测 S_5^2 的高 8 位, 因 S_{5H}^2 的其余位和 S_{7L}^2 均已知, 故可求 X_2^2 。将 X_2^2 代入第二个公式, 因 R_2^2 为已知, 故可求 W_2^2 。

$$R_1^3 = S(L_1(W_{1L}^2 || W_{2H}^2))$$

$$R_2^3 = S(L_2(W_{2L}^2 || W_{1H}^2))$$

W_1^2 和 W_2^2 已由前面公式分别求出, 将 W_1^2 和 W_2^2 代入上述公式即可求 R_1^3 和 R_2^3 。

$$S_{16}^2 = 2^{15} S_{15}^2 + 2^{17} S_{13}^2 + 2^{21} S_{10}^2 + 2^{20} S_4^2 + (1 + 2^8) S_0^2 \bmod (2^{31} - 1)$$

$$\text{If } S_{16}^2 = 0, \text{ then set } S_{16}^2 = 2^{31} - 1.$$

预猜测 S_{15}^2 的低 15 位, S_{13}^2 的高 8 位和 S_{10}^2 的高 8 位, 因 S_{15H}^2 已由前面公式求出, S_{13}^2 和 S_{10}^2 的其余位均为已知的常量和 IV 向量, S_4^2 为已知, S_0^2 在分析第一步已猜, 故可求 S_{16}^2 。

至此在 $t=2$ 时刻, 已猜测 $S_0^2, S_5^2, S_9^2, S_{10}^2, S_{13}^2$ 的高 8 位和 S_{15}^2 的低 15 位, 共 55 比特。推导出来的有 $S_{14L}^2, S_{15H}^2, S_{16}^2, R_1^3, R_2^3$ 。因此 $t=2$ 时刻的全部已知量为 $S_0^2, S_3^2, S_4^2, S_5^2, S_7^2, S_8^2, S_9^2, S_{10}^2, S_{13}^2, S_{14L}^2, S_{15}^2, S_{16}^2$ 。

2.2.2 $t=3$ 时刻分析过程

在 $t=3$ 时刻, 由 $(S_1^2, S_2^2, \dots, S_{15}^2, S_{16}^2) \rightarrow (S_0^3, S_1^3, \dots, S_{14}^3, S_{15}^3)$ 可知该时刻的已知量为 $S_2^3, S_3^3, S_4^3, S_6^3, S_7^3, S_8^3, S_9^3, S_{12}^3, S_{13L}^3, S_{14}^3, S_{15}^3$, 此外还有 R_1^3 和 R_2^3 。在此需要注意的是 S_{13}^3 已不是由 S_0^2 移位得到, 因此其高 15 位均为未知。分析过程如下:

$$X_0^3 = S_{15H}^3 || S_{14L}^3$$

$$W^3 = (X_0^3 \oplus R_1^3) \oplus R_2^3$$

$$Z^3 = W^3 \oplus X_3^3$$

$$X_3^3 = S_{2L}^3 || S_{0H}^3$$

S_{15H}^3, S_{14L}^3 为已知, 故可求 X_0^3 。将 X_0^3 代入第二个公式, 因 R_1^3 和 R_2^3 均已知, 故可求 W^3 。将 W^3 代入第三个公式, 因 Z^3 为已知密钥, 故可求 X_3^3 。将 X_3^3 代入第四个

公式即可求 S_{0H}^3 。因 S_{0L}^3 已知, 故可求 S_0^3 。

$$X_1^3 = S_{11L}^3 || S_{9H}^3$$

$$W_1^3 = R_1^3 \oplus X_1^3$$

S_{9H}^3 为已知, S_{11L}^3 为已知的常量和 IV 向量, 故可求 X_1^3 。将 X_1^3 代入第二个公式, 因 R_1^3 已知, 故可求 W_1^3 。

$$X_2^3 = S_{7L}^3 || S_{5H}^3$$

$$W_2^3 = R_2^3 \oplus X_2^3$$

预猜测 S_5^3 的高 8 位, 因 S_5^3 的其余位和 S_{7L}^3 均已知, 故可求 X_2^3 。将 X_2^3 代入第二个公式, 因 R_2^3 已知, 故可求 W_2^3 。

$$R_1^4 = S(L_1(W_{1L}^3 || W_{2H}^3))$$

$$R_2^4 = S(L_2(W_{2L}^3 || W_{1H}^3))$$

W_1^3 和 W_2^3 已由前面公式求出, 将其代入上述公式即可求 R_1^4 和 R_2^4 。

$$S_{16}^3 = 2^{15} S_{15}^3 + 2^{17} S_{13}^3 + 2^{21} S_{10}^3 + 2^{20} S_4^3 + (1 + 2^8) S_0^3 \bmod (2^{31} - 1)$$

$$\text{If } S_{16}^3 = 0, \text{ then set } S_{16}^3 = 2^{31} - 1$$

预猜测 S_{13}^3 的高 15 位和 S_{10}^3 的高 8 位, 因 S_{10}^3 的其余位和 $S_{15}^3, S_{13L}^3, S_4^3$ 均已知, S_0^3 已求出, 故可求 S_{16}^3 。

至此在 $t=3$ 时刻, 已猜测 S_5^3, S_{10}^3 的高 8 位和 S_{13}^3 的高 15 位, 共 31 比特。推导出的有 $S_0^3, S_{16}^3, R_1^4, R_2^4$ 。因此 $t=3$ 时刻的全部已知量为 $S_0^3, S_2^3, S_3^3, S_4^3, S_5^3, S_6^3, S_7^3, S_8^3, S_9^3, S_{10}^3, S_{12}^3, S_{13}^3, S_{14}^3, S_{15}^3, S_{16}^3$ 。

2.2.3 $t=4$ 时刻分析过程

在 $t=4$ 时刻, 由 $(S_1^3, S_2^3, \dots, S_{15}^3, S_{16}^3) \rightarrow (S_0^4, S_1^4, \dots, S_{14}^4, S_{15}^4)$ 可知该时刻的已知量为 $S_1^4, S_2^4, S_3^4, S_4^4, S_5^4, S_6^4, S_7^4, S_8^4, S_9^4, S_{11}^4, S_{12}^4, S_{13}^4, S_{14}^4, S_{15}^4$, 此外还有 R_1^4 和 R_2^4 。分析过程如下:

$$X_0^4 = S_{15H}^4 || S_{14L}^4$$

$$W^4 = (X_0^4 \oplus R_1^4) \oplus R_2^4$$

$$Z^4 = W^4 \oplus X_3^4$$

$$X_3^4 = S_{2L}^4 || S_{0H}^4$$

S_{15H}^4, S_{14L}^4 为已知, 故可求 X_0^4 。将 X_0^4 代入第二个公式, 因 R_1^4 和 R_2^4 均已知, 故可求 W^4 。将 W^4 代入第三个公式, 因 Z^4 为已知密钥, 故可求 X_3^4 。将 X_3^4 代入第四个公式即可求 S_{0H}^4 。因 S_{0L}^4 已知, 故可求 S_0^4 。

$$X_1^4 = S_{11L}^4 || S_{9H}^4$$

$$W_1^4 = R_1^4 \oplus X_1^4$$

S_{9H}^4 和 S_{11L}^4 均已知, 故可求 X_1^4 。将 X_1^4 代入第二个公式, 因 R_1^4 为已知, 故可求 W_1^4 。

$$X_2^4 = S_{7L}^4 || S_{5H}^4$$

$$W_2^4 = R_2^4 \oplus X_2^4$$

S_{5H}^4 和 S_{7L}^4 均已知, 故可求 X_2^4 。将 X_2^4 代入第二个公

式,因 R_2^4 为已知,故可求 W_2^4 。

$$R_1^5 = S(L_1(W_{1L}^4 || W_{2H}^4))$$

$$R_2^5 = S(L_2(W_{2L}^4 || W_{1H}^4))$$

W_1^4 和 W_2^4 已由前面公式求出,将其代入上述公式即可求 R_1^5 和 R_2^5 。

$$S_{16}^4 = 2^{15} S_{15}^4 + 2^{17} S_{13}^4 + 2^{21} S_{10}^4 + 2^{20} S_4^4 + (1 + 2^8) S_0^4 \bmod (2^{31} - 1)$$

$$\text{If } S_{16}^4 = 0, \text{ then set } S_{16}^4 = 2^{31} - 1$$

预猜测 S_{10}^4 的高 8 位,因 S_{10}^4 的其余位和 $S_{15}^4, S_{13}^4, S_4^4$ 均已知, S_0^4 已求出,故可求 S_{16}^4 。

至此在 $t=4$ 时刻,已猜测 S_{10}^4 的高 8 位,共 8 比特。推导出来的有 $S_0^4, S_{16}^4, R_1^5, R_2^5$ 。因此 $t=4$ 时刻的全部已知量为 $S_0^4, S_1^4, S_2^4, S_3^4, S_4^4, S_5^4, S_6^4, S_7^4, S_8^4, S_9^4, S_{10}^4, S_{11}^4, S_{12}^4, S_{13}^4, S_{14}^4, S_{15}^4, S_{16}^4$ 。

在 $t=5$ 时刻,由 $(S_1^4, S_2^4, \dots, S_{15}^4, S_{16}^4) \rightarrow (S_0^5, S_1^5, \dots, S_{14}^5, S_{15}^5)$ 可知该时刻的已知量为 $S_0^5, S_1^5, S_2^5, S_3^5, S_4^5, S_5^5, S_6^5, S_7^5, S_8^5, S_9^5, S_{10}^5, S_{11}^5, S_{12}^5, S_{13}^5, S_{14}^5, S_{15}^5$, 此外还有 R_1^5 和 R_2^5 。因此 $t=5$ 时刻的全部内部单元均已找到。之后即可用所求内部单元生成密钥来检测与原密钥是否相同,若相同则说明猜的是对的,否则需要重新搜索。

3 攻击的时间复杂度

截止到 $t=4$ 时刻 $S_0^5 \sim S_{15}^5, R_1^5$ 和 R_2^5 都求出来了。计算过程中用到了 Z^2, Z^3, Z^4 。猜测的位置是 $t=0$ 时刻的 S_9^0, S_5^0 的高 8 位,共 16 比特; $t=1$ 时刻的 S_9^1, S_5^1 的高 8 位,共 16 比特; $t=2$ 时刻的 $S_9^2, S_5^2, S_2^2, S_{10}^2, S_{13}^2$ 的高 8 位和 S_{15}^2 的低 15 位,共 55 比特; $t=3$ 时刻的 S_3^3, S_{10}^3 的高 8 位和 S_{13}^3 的高 15 位,共 31 比特; $t=4$ 时刻的 S_{10}^4 的高 8 位,共 8 比特。因此总共猜测 $16+16+55+31+8=126$ 比特。此时搜索复杂度为 $O(2^{126})$,而其穷尽搜索复杂度为 $O(2^{128})$,所以提高了搜索效率。

4 结束语

文中提出了对 ZUC 算法的 Guess and Determine 攻击,先猜一部分内部单元,然后去推导剩余的,从而得到全部的内部单元。其搜索复杂度为 $O(2^{126})$,而穷尽搜索的复杂度为 $O(2^{128})$,因此提高了搜索效率。

参考文献:

- [1] 中国通信标准化协会. ZUC 算法公开评估[S/OL]. 2011. <http://www.ccsa.org.cn/zuc.php>.
- [2] CCSA. 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3[S]. 2011.
- [3] Feng Xiutao, Liu Jun, Zhou Zhaocun, et al. A Byte-based Guess and Determine Attack on SOSEMANUK[M]//Asia-crypt. [s. l.]:[s. n.], 2010:146-157.
- [4] Hastad J, Naslund M. The Stream Cipher Polar Bear[R/OL]. 2005. <http://www.ecrypt.eu.org/stream>.
- [5] Mattsson J. A Guess-and-Determine Attack on the Stream Cipher Polar Bear[EB/OL]. 2006. <http://www.ecrypt.eu.org/stream/polarbear.html>.
- [6] Hasanzadeh M, Shakour E, Khazaei S. Improved Cryptanalysis of Polar Bear[EB/OL]. 2006. <http://www.ecrypt.eu.org/stream>.
- [7] Hawkes P, Rose G. Guess and Determine Attacks on SNOW[C]//SAC, 2002, LNCS 2595. [s. l.]:[s. n.], 2002:37-46.
- [8] 张海霞. 流密码算法 SOSEMANUK 的安全性分析[D]. 西安:西安电子科技大学, 2011.
- [9] 刘树凯, 关杰, 常亚勤. 针对流密码 K2 算法的猜测决定攻击[J]. 计算机工程, 2011(7):168-170.
- [10] 冯登国, 裴定. 密码学导引[M]. 北京:科学出版社, 1999.
- [11] 龙冬阳. 应用编码与计算机密码学[M]. 北京:清华大学出版社, 2005.
- [12] 丁存生, 肖国镇. 流密码学及其应用[M]. 北京:国防工业出版社, 1994.

(上接第 134 页)

[//www.cs.wustl.edu/mobilab/projects/agilla/index.html](http://www.cs.wustl.edu/mobilab/projects/agilla/index.html).

- [9] 丁月华, 杨敏, 文贵华, 等. 基于 XML 的异构数据源集成与交换的实现[J]. 计算机应用与软件, 2006, 23(10):34-38.
- [10] Kamran S, Maarten W, van Sinderen. Middleware support for quality of context in pervasive context-aware systems[C]//Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops. New

York, USA:IEEE Computer Society, 2007:461-466.

- [11] Rasheed F, Lee Young-Koo, Lee Sung-Young. Applying Context Summarization Techniques in Pervasive Computing System[C]//3rd Workshop on Software Technologies for Future Embedded & Ubiquitous Systems (SEUS 2006). Gyeongju, Korea:IEEE Computer Society, 2006:107-112.
- [12] 宋欢, 邬家伟, 成永常. 基于普适计算的学习框架的研究[J]. 计算机技术与研究, 2010, 20(4):117-123.