针对简化版 Trivium 算法的线性分析

孙文龙 1 关杰 1 刘建东 2
1 (信息工程大学电子技术学院 郑州 450004)
2 (北京市2857信箱四室 北京 100085)

摘要 流密码 Trivium 算法是欧洲密码工程 eSTREAM 的 7 个最终获选算法之一。该文针对初始化为 288 轮的简化版 Trivium 算法进行了线性分析，更正了 Turan 等人给出的关于密钥、初始化向量和密钥流比特的表达式，并给出了当允许选取特殊的密钥和 IV 时，搜索最佳线性逼近式的算法。据此算法，找到了 3 个线性偏差为 $2^{-25}$ 的线性逼近式，改进了 Turan 等人给出的线性偏差为 $2^{-31}$ 的线性分析结果。

关键词 密码分析；线性分析；Trivium；流密码

1 引言


Trivium 是由 Cannière 和 Preneel 基于分组密码设计的面向硬件实现的同步流密码算法，密钥和初始化向量规模均为 80 bit，内部状态为 288 bit，由 3 个级数分别为 93、84 和 111 的非线性反馈移位寄存器 (NFSR) 互控更新内部状态。该结构设计简单，便于硬件实现。

目前针对 Trivium 的安全性分析主要有代数攻
击、滑动攻击、立方攻击和猜测决定攻击等。Maximov 和 Biryukov 对 Trivium 进行状态恢复攻击和统计测试，恢复 Trivium 的内部状态需要的计算量为 $c \cdot 2^{33.5}$，其中常数 $c$ 是求解线性方程组的复杂度。Raddum 通过求解一个线性方程组来恢复 288 bit 的初始状态，其计算复杂度为 $O(2^{164})$。Priezmut-Schmidt 等人针对 Trivium 算法提出了滑动攻击，指出 Trivium 算法在多于 2 个滑动对，Dinur 和 Shamir 针对 767 个时刻的初始化 Trivium 算法实现了滑动攻击，所需的计算复杂度为 $2^{45}$ 比特操作。孙国平等 [1] 通过错误注入的方法改变 Trivium 内部状态中的 52 bit，提出了一种基于选择差分的 Trivium 滑动攻击，攻击的计算复杂度为 $2^{15}$。


本文针对初始化为 288 轮的简化版 Trivium 算法进行线性分析，更正了 Turan 等人给出的关于密钥、初始化向量和密钥流比特的表达式，并给出了当允许选取特殊的密钥和 IV 时，搜索最佳线性逼近式的算法，据此算法，找到了 3 个线性偏差为 $2^{-23}$ 的线性逼近式，改进了 Turan 等人提出的线性偏差为 $2^{-31}$ 的线性分析结果。

## 2 Trivium 算法描述

### 2.1 符号说明

- **K**：80 bit 密钥，$K = (k_1, k_2, \ldots, k_{80})$；
- **IV**：80 bit 初始化向量，$IV = (iv_1, iv_2, \ldots, iv_{80})$；
- $s_i(t)$：第 $t$ 时刻内部状态的第 $i$ 比特，$1 \leq i \leq 288$；
- $s_i$：第 $t$ 时刻 288 bit 内部状态，$s_i = (s_i(1), s_i(2), \ldots, s_i(288))$；
- $t$：更新比特，$1 \leq i \leq 3$；
- $z_i$：第 $i$ 时刻的输出密钥流比特；
- $+$：GF(2) 上的加法运算；
- $\cdot$：GF(2) 上的乘法运算。

### 2.2 算法描述

Trivium 算法分为初始化算法和密钥流生成算法两个部分：在初始化阶段，系统在 $t$ 时刻使用特定位置的 15 bit 内部状态更新 $t+1$ 时刻的 3 bit 内部状态，然后运行初始化算法 4 • 288 个时钟周期。不输出任何密钥流；在密钥流生成阶段，由 3 个移位寄存器的各 2 比特异或生成密钥流。具体算法用伪代码描述如下：

$$
(s_1, s_2, \ldots, s_{288}) \leftarrow (k_1, \ldots, k_{80}, 0, \ldots, 0),
(s_{94}, \ldots, s_{177}) \leftarrow (iv_1, \ldots, iv_{80}, 0, \ldots, 0),
(s_{178}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1),
$$

for $i = 1$ to $N$ do

end for

### 3 288 轮 Trivium 的线性分析

#### 3.1 基础知识

线性密码分析 [10] 是 Matsui 在 1993 年欧洲密码年会上提出的一种对迭代型分组密码算法的已知明文攻击方法，其基本思想是通过寻找分组密码算法的一个有效的线性逼近来破译密码系统。

线性密码分析的方法是寻找一个给定密码算法的具有下列形式的“有效的”线性表达式：

$$
P_{i_1, i_2, \ldots, i_n} + C_{j_1, j_2, \ldots, j_n} = K_{k_1, k_2, \ldots, k_n} \quad (1)
$$

其中 $i_1, i_2, \ldots, i_n, j_1, j_2, \ldots, j_n$ 和 $k_1, k_2, \ldots, k_n$ 表示固有的比特位置，并且对随机给定的明文 $P$ 和对应的密文 $C$，等式 (1) 成立的概率 $p \neq 1/2$，用 $|p - 1/2|$ 来刻画等式 (1) 的有效性，称 $|p - 1/2|$ 为线性偏差 $\varepsilon$，如果 $|p - 1/2|$ 是最大的，将对应的线性表达式称为最佳线性逼近式。

针对多轮的分组密码 [11]，首先对不同轮的非线性函数进行逼近，然后将各个逼近有效地组合，最终得出有效的线性逼近。分组密码的线性逼近的概率与每一轮线性逼近的概率都要有关，可由下面的堆积引理来计算形如等式 (1) 成立的概率。

**引理** [11]。设 $X_i (1 \leq i \leq n)$ 是独立的随机变量，$Pr(X_i = 0) = p_i$, $Pr(X_i = 1) = 1 - p_i$，则

$$
Pr(X_1 \oplus X_2 \oplus \cdots \oplus X_n = 0) = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} (p_i - 1/2) \quad (2)
$$

任意一个流密码都可以看作布尔函数 $F$: $\mathbb{F}_2 \rightarrow \mathbb{F}_2$ 的函数。
我们发现文献 [8] 中的结论是错误的。对于第 (6) 中第 144 轮的内部状态变化，通过迭代对密钥和 IV 比特表示出来，在这些表达式中：

（1）没有出现 \(s_0(79)\) 这个变量，因此，\(z_i\) 关于密钥和 IV 比特的表达式中不存在二次项 \(s_0(77) \cdot s_0(79)\)；

（2）用二次项 \(s_0(76) \cdot s_0(77)\) 表示的，只有 \(s_{144}(144)\) 和 \(s_{144}(237)\) 这两项，因此，由于 \(z_i\) 关于密钥和 IV 比特的表达式中不存在二次项 \(s_0(76) \cdot s_0(77)\)；

（3）用三次项 \(s_0(146) \cdot s_0(147) \cdot s_0(148)\) 和 \(s_0(146) \cdot s_0(147) \cdot s_0(149)\) 表示的只有 \(s_{144}(6)\) 一项，因此，\(z_i\) 关于密钥和 IV 比特的表达式中存在三次项 \(s_0(146) \cdot s_0(147) \cdot s_0(148)\) 和 \(s_0(146) \cdot s_0(147) \cdot s_0(149)\) 的是存在的。

故对于式 (5) 文献 [8] 附录中 Turan 等人的表示式可知，进一步，我们使用 MATLAB 软件编程验证了 (1)、(2) 和 (3)，\(z_i\) 关于密钥和 IV 比特的正确表达式如下所示：

\[
z_i = s_{144}(6) + s_{144}(33) + s_{144}(57) + s_{144}(14) + s_{144}(96) + s_{144}(111) + s_{144}(129) + s_{144}(143) + s_{144}(144) + s_{144}(150) + s_{144}(163) + s_{144}(164) + s_{144}(165) + s_{144}(186) + s_{144}(192) + s_{144}(208) + s_{144}(209) + s_{144}(210) + s_{144}(231) + s_{144}(235) + s_{144}(236) + s_{144}(237) + s_{144}(252)
\]

由堆积引理，式 (5) 以 \(2^n \cdot 0.25^k = 2^{-\Delta}\) 的线性偏差逼近下式：

\[
z_i = s_{144}(6) + s_{144}(33) + s_{144}(57) + s_{144}(84) + s_{144}(96) + s_{144}(99) + s_{144}(111) + s_{144}(129) + s_{144}(143) + s_{144}(144) + s_{144}(150) + s_{144}(163) + s_{144}(164) + s_{144}(165) + s_{144}(186) + s_{144}(192) + s_{144}(208) + s_{144}(209) + s_{144}(210) + s_{144}(231) + s_{144}(235) + s_{144}(236) + s_{144}(237) + s_{144}(252)
\]

对于式 (6)，将等式左边的密钥流比特 \(z_i\) 通过迭代用密钥和初始化向量比特表示出来。经过分析，我们发现文献 [8] 附录中 Turan 等人给出的密钥流比特 \(z_i\) 关于密钥和 IV 比特的表达式有误，下面给出说明：
\[
\begin{align*}
& s_0(63) \cdot s_0(64) + s_0(64) \cdot s_0(65) + \\
& s_0(64) \cdot s_0(143) + s_0(64) \cdot s_0(170) + \\
& s_0(65) \cdot s_0(142) + s_0(65) \cdot s_0(169) + \\
& s_0(67) \cdot s_0(68) + s_0(70) \cdot s_0(71) + \\
& s_0(103) \cdot s_0(104) + s_0(106) \cdot s_0(107) + \\
& s_0(118) \cdot s_0(119) + s_0(124) \cdot s_0(125) + \\
& s_0(127) \cdot s_0(128) + s_0(130) \cdot s_0(131) + \\
& s_0(133) \cdot s_0(149) + s_0(134) \cdot s_0(148) + \\
& s_0(142) \cdot s_0(143) + s_0(147) \cdot s_0(148) + \\
& s_0(151) \cdot s_0(152) + s_0(154) \cdot s_0(155) + \\
& s_0(160) \cdot s_0(161) + s_0(163) \cdot s_0(164) + \\
& s_0(166) \cdot s_0(167) + s_0(13) \cdot s_0(39) \cdot s_0(40) + \\
& s_0(14) \cdot s_0(38) + s_0(39) + \\
& s_0(19) \cdot s_0(45) + s_0(46) + s_0(20) \cdot s_0(44) + \\
& s_0(45) + s_0(37) + s_0(63) \cdot s_0(64) + \\
& s_0(38) \cdot s_0(39) \cdot s_0(40) + s_0(43) \cdot s_0(39) + \\
& s_0(41) + s_0(38) + s_0(39) \cdot s_0(119) + \\
& s_0(38) \cdot s_0(62) + s_0(63) + s_0(39) \cdot s_0(40) + \\
& s_0(118) + s_0(44) \cdot s_0(45) \cdot s_0(46) + \\
& s_0(44) \cdot s_0(45) \cdot s_0(47) + s_0(44) \cdot s_0(45) + \\
& s_0(125) + s_0(45) \cdot s_0(46) + s_0(124) + \\
& s_0(62) \cdot s_0(63) + s_0(64) + s_0(62) \cdot s_0(63) + \\
& s_0(65) + s_0(62) + s_0(63) \cdot s_0(143) + \\
& s_0(63) \cdot s_0(64) + s_0(142) + s_0(133) \cdot s_0(147) + \\
& s_0(148) + s_0(134) + s_0(146) \cdot s_0(147) + \\
& s_0(146) + s_0(147) + s_0(148) + \\
& s_0(146) \cdot s_0(147) \cdot s_0(149) \cdot (7)
\end{align*}
\]

### 3.2.2 作线性逼近式的选择算法

观察可知，式(7)中共有 79 个非线性项，57 个二次项和 22 个三次项，如果令式(7)中所有的非线性项为 0，则我们找到了该式的一个线性逼近：

\[
z_1 = 1 + k_1 + k_2 + k_{15} + k_{21} + k_{23} + k_{29} + k_{34} + \\
k_{57} + k_{67} + k_{65} + k_{72} + iv_3 + iv_4 + iv_{21} + \\
v_3 + v_{30} + v_{13} + v_{19} + v_{15} + iv_{17} + iv_{12} + iv_{13}
\]

由堆形引理，式(8)由式(7)逼近的线性偏差为

\[
2^{77} \cdot (0.25)^{57} \cdot (0.375)^{22} = 2^{-67.13}.
\]

合成式(5)~(8)，由堆形引理，式(8)成立的线性偏差为 \( \varepsilon = 2 \cdot 2^{-67.13} = 2^{-75.13} \ll 2^{-8} \cdot|K|/2 = 2^{-40} \)

（|K| 表示密钥规模，对 Trivium 而言，密钥规模 |K| = 80），但是这个偏差太小，对于线性密码分析没有意义。

此时，可以通过选择特殊的密钥和 IV 的方法来增大式(8)成立的线性偏差。设 \( \Omega_k = \{ k_i | k_i = 0, 1 \leq i \leq 80 \} \) 表示选择为 0 的密钥比特组成的集合，

\( |\Omega_k| \) 表示 \( \Omega_k \) 的规模；\( \Omega_{IV} = \{ i | i = 0, 1 \leq i \leq 80 \} \) 表示选择为 0 的 IV 比特组成的集合，

\( |\Omega_{IV}| \) 表示 \( \Omega_{IV} \) 的规模；\( n_i \) 和 \( n_k \) 分别表示选择 \( \Omega_k \) 和 \( \Omega_{IV} \) 后式(7)中剩余二次项和三次项的数量，式(8)成立的线性偏差 \( \varepsilon \) 为

\[
\varepsilon = 2 \cdot 2^{-67} \cdot 2^{n_i + n_k - 1} \cdot (0.25)^{n_i} \cdot (0.375)^{n_k} \\
= (0.5)^{n_i + n_k} \cdot (0.75)^{n_k} \quad (9)
\]

经分析，对式(7)，选择不同的 \( \Omega_k \) 和 \( \Omega_{IV} \) 将会影响到式(7)中二次项数量 \( n_i \) 和三次项数量 \( n_k \)。进而影响到式(8)成立的线性偏差 \( \varepsilon \) 的大小。但是文献[8]和文献[9]中并没有给出 \( \Omega_k \) 和 \( \Omega_{IV} \) 的选择准则。下面，我们就在这一问题给出搜索最佳线性逼近式的算法 1。

### 算法 1

#### 1. 最佳线性逼近式的搜索算法

输入：\( \Omega_k \) 选择的规模 \( n_k \), \( \Omega_{IV} \) 选择的规模 \( n_{IV} \):

1. 集合 \( \Omega \) 和 II 初始化为空集，统计式(7)的所有非线性项中各个比特的频次，将频次最大的比特存入集合 \( \Omega \) 中；

2. 1. 若 \( |\Omega| = 1 \)，判断 \( \Omega \) 中比特类型，若为密钥则存入 \( \Omega_k \) 中，否则存入 \( \Omega_{IV} \) 中；

2. 2. 若 \( |\Omega| \geq 2 \)，统计 \( \Omega \) 中各个比特涉及的二次项的频次，将频次最大的比特存入集合 II 中；

2. 2. 1. 若 \( |\Omega| = 1 \)，判断 \( \Omega \) 中比特类型，若为密钥则存入 \( \Omega_k \) 中，否则存入 \( \Omega_{IV} \) 中；

2. 2. 2. 若 \( |\Omega| \geq 2 \)，则任选 \( \Omega \) 中一个比特，判断该比特类型，若为密钥则存入 \( \Omega_k \) 中，否则存入 \( \Omega_{IV} \) 中；

3. 根据选择的 \( \Omega_k \) 和 \( \Omega_{IV} \)，重新计算式(7)；

4. if \( (|\Omega_k| < n_k) \&\& (|\Omega_{IV}| < n_{IV}) \) 返回步 1；

else if \( (|\Omega_k| = n_k) \&\& (|\Omega_{IV}| \neq n_{IV}) \) 则停止搜索密钥比特，返回步 1。继续搜索满足条件的 IV 比特；

else if \( (|\Omega_k| \neq n_k) \&\& (|\Omega_{IV}| = n_{IV}) \) 则停止搜索 IV 比特，返回步 1，继续搜索满足条件的密钥比特；

else if \( (|\Omega_k| = n_k) \&\& (|\Omega_{IV}| = n_{IV}) \) 输出：\( (\Omega_k, \Omega_{IV}, \varepsilon) \)。

### 定理 1

当给定选择为 0 的密钥和 IV 比特的规模时，算法 1 搜索到的是最佳线性逼近式，即设定算法 1 搜索到的密钥和 IV 比特为 0 后，式(8)成立的线性偏差 \( \varepsilon \) 最大。

证明。当 \( |\Omega| = 1 \)，将该比特设定为 0，则消掉
的非线性项数量最多，显然式 (8) 成立的线性偏差最大。当 \(|\Omega| \geq 2\)，不妨设 \(\Omega = \{a, b\}\)，设比特 \(a\) 和 \(b\) 的频次为 \(n\)，分别统计比特 \(a\) 和 \(b\) 涉及到的二次项频次，记为 \(m_1\) 和 \(m_2\)（不妨设 \(m_1 \geq m_2\)），设未选择 \(a\) 或者 \(b\) 为 0 之前，式 (7)中有 \(r\) 个二次项，\(t\) 个三次项。若把 \(a\) 设定为 \(0\)，式 (8) 成立的线性偏差为 \(\varepsilon_\ell = (0.5)^{-(m_1 + 3)} \cdot (0.75)^{-(m_2 + 3)}\)；若把 \(b\) 设定为 \(0\)，式 (8) 成立的线性偏差为 \(\varepsilon_\ell = (0.5)^{-(m_2 + 3)} \cdot (0.75)^{-(m_1 + 3)}\)。显然 \(\varepsilon_\ell > \varepsilon_\ell\)，因此欲使偏差最大，应选择 \(a = 0\)。

算法 1 就是按照这样的准则搜索满足要求的密钥和 IV 比特的。因此，当设定算法 1 搜索到的密钥和 IV 比特为 0 后，式 (8) 成立的线性偏差最大，即算法 1 搜索的是最佳线性逼近式。

此外，我们给出的最佳线性逼近式的搜索算法具有普适性，可以把它进一步推广，用于其它密码算法的线性分析中。

根据算法 1，表 1 给出了不同的 \(|\Omega_k|\) 和 \(|\Omega_n|\) 对应的最大偏差。

<table>
<thead>
<tr>
<th>(\Omega_n)</th>
<th>(0)</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
<th>(6)</th>
<th>(7)</th>
<th>(8)</th>
<th>(9)</th>
<th>(10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>9</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>10</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

注："-"表示在 \(|\Omega_k|\) 和 \(|\Omega_n|\) 取相应值时的偏差 \(\varepsilon < 2^{-40}\)。

根据表 1 的结论，攻击者可以依据攻击能力的大小利用算法 1 选择最佳的 \(\Omega_k\) 和 \(\Omega_n\)。例如，若攻击者仅能够选择 IV，则可以得到线性逼近式的最大线性偏差为 \(2^{-30.98}\)；若攻击者能够选择 10 个密钥比特和 15 个 IV 比特，则可以得到线性逼近式的最大偏差为 \(2^{-20}\)。附录 1 中给出了利用算法 1 搜索到的最佳密钥和 IV 比特的部分实例。

特别地，根据算法 1，如果选择如下 10 个特定的密钥比特和 10 个特定的 IV 比特时，有以下结论。

| \(\Omega_n\) | \(|\Omega_n| = 10\) | \(|\Omega_k| = 10\) | \(\Omega_k\)及 \(\varepsilon\) | \(\varepsilon_k\) |
|--------------|-----------------|-----------------|-----------------|-----------------|
| \(\varepsilon_k\) | \(\varepsilon_k\) | \(\varepsilon_k\) | \(\varepsilon_k\) |
| \(k_1, k_2, k_3, k_4, k_5\) | \(k_6\) | \(k_7\) | \(k_8\) | \(k_9\) |

根据表 2 中 \(\Omega_k\) 的不同选择，我们找到了如下 3 个线性逼近式，其线性偏差均为 \(2^{-25}\)。
基础上，选择规模为 $|\Omega_k| = 10$ 和 $|\Omega_N| = 10$ 的 $\Omega_k$ 和 $\Omega_N$，找到了另一个线性偏差为 $2^{-31}$ 的线性逼近式;本文选择规模为 $|\Omega_k| = 10$ 和 $|\Omega_N| = 10$ 的 $\Omega_k$ 和 $\Omega_N$，找到了 3 个具有相同线性偏差 $2^{-25}$ 的线性逼近式; 另外，若选取规模为 $|\Omega_k| = 10$ 和 $|\Omega_N| = 13$ 的 $\Omega_k$ 和 $\Omega_N$，可得到 3 个具有相同线性偏差 $2^{-22}$ 的线性逼近式。

因此，本文的分析结果优于 Turan 和贾艳艳等人的结果，对比如表 3 所示。

### 表 3 结果对比

| $|\Omega_k|$ | $|\Omega_N|$ | 线性 偏差 $\varepsilon$ | 线性逼近个数 | 数据量 $N$ | 成功率/ % |
|---|---|---|---|---|---|
| (10, 10) | 2^{-31} | 1 | $2^{12}$ | 97.77 |
| (10, 10) | 2^{-25} | 3 | $2^{10}$ | 97.77 |
| (10, 13) | 2^{-31} | 2 | $2^{11}$ | 97.77 |
| (10, 13) | 2^{-22} | 3 | $2^{14}$ | 97.77 |

4 结束语

本文针对初始化为 288 轮的简化版 Trivium 算法进行了线性分析，更正了 Turan 等人给出的关于密钥初始化向量和密钥流比特的表达式，并给出了当允许选取特定的密钥和 IV 时，搜索最佳线性逼近式的算法。据此算法，找到了 3 个线性偏差为 $2^{-25}$ 的线性逼近式，改进了 Turan 等人提出的线性偏差为 $2^{-31}$ 的线性分析结果。此外，该搜索算法可以进一步推广到其他密码算法的线性分析中，如何对更多轮的 Trivium 算法进行线性分析是值得进一步研究的问题。

参考文献


SUN Wen-Long. born in 1988, master. His research interests focus on the design and analysis of stream ciphers.

GUAN Jie. born in 1974, Ph. D., associate professor, master supervisor. Her research interests include design and analysis of symmetric ciphers.

LIU Jian-Dong. born in 1974, senior engineer. His research interest is communication security.
Background

The initialization processes of stream ciphers are our main research areas, due to the secure faults often existing in this stage. With trying all kinds of attacks on the ciphers, the secure faults in the initialization process are hoped to be found. Based on our analysis, some secure suggestions are hoped to be presented, to improve the design and to enhance the security of ciphers.

For some ciphers, such as SNOW2.0, Trivium, Py-family, Salsa20 and so on, our analysis results have been published. Stream ciphers SNOW2.0, Trivium, Py and Salsa20 are all submitted to the eSTREAM project. We point out that SNOW2.0 can’t be against the Guess and Determine Attack, that some differential chains with high possibility in Trivium can be found by a differential cryptanalysis based on automatic deduction, that improved related-key attacks on the Py-family of stream ciphers are presented by extending the least significant bitwise distinguisher to a word-based distinguisher, and that the differential algebraic analysis techniques can obtain the best result on 5-round Salsa20.

In general, the results above are based on our careful analysis on the initialization process of these ciphers with trying all kinds of attacks. So it is very important to analyze the initialization for ensuring the security of stream ciphers.

In this paper, we apply linear cryptanalysis to the simplified Trivium with the initialization of 288 rounds. Trivium is a hardware-oriented stream cipher designed in 2005 by Cannière and Preneel, and has been chosen as one of the final ciphers by eSTREAM project. It is well known that Trivium is on the edge of low cost and compactness, with a simple and elegant structure. Although Trivium has attracted a lot of interest, there are no any standard attacks faster than the exhaustive attack, except for side-channel attacks. For the simplified Trivium, Turan et al. found one linear approximation with bias $2^{-11}$. In this paper, the equation, which involves the key bits, initial vector bits and the first key stream bit in linear approximations of Turan, is corrected. In addition, when special Key bits and IV bits are allowed to be chosen, an algorithm to search the best linear approximations is presented. Based on this algorithm, 3 linear approximations with the same linear bias $2^{-21}$ are found, which is better than Turan’s, and our results are the best so far.