第 36 卷 第 9 期 计 算 机 工 程 2010 年 5 月
Vol.36 No.9 Computer Engineering May 2010
·安全技术· 文章编号：1000—3428(2010)09—0129—02 文献标识码：A 中图分类号：TP309.2

# 基于选择差分的 Trivium 猜测攻击

孙国平，胡予濮，白生江
(西安电子科技大学计算机网络与信息安全教育部重点实验室，西安 710071)

**摘 要：**给出一种基于选择差分对 Trivium 算法进行猜测攻击的方法。通过分析 Trivium 密钥流生成方程，确定需要改变 Trivium 288 bit 内部状态中的 52 bit，使用错误注入改变所确定的 52 bit，并生成密钥流，与原始密钥流进行差分。该方法只需猜测 45 bit 即可使密钥流生成方程中的 177 个非线性方程成为线性方程，加上已有的 66 个线性方程，使用高斯消元法获得剩余的 243 bit，从而攻破 Trivium。
**关键词：**Trivium 算法；选择差分攻击；猜测攻击

# Guess Attack on Trivium Based on Chosen Differential

SUN Guo-ping, HU Yu-pu, BAI Sheng-jiang
(Key Laboratory of Computer Network and Information Security of Ministry of Education, Xidian University, Xi'an 710071)

**【Abstract】**This paper proposes a guess attack method on Trivium based on chosen differential. By analyzing the key generation equations of Trivium and determining 52 bit of its interior state which need to be altered, and then fault injections are used to alter these 52 bit and a faulty key stream can be obtained. The difference between the faulty key stream and the original key stream is computed, so that it only need guess 45 bit to make 177 nonlinear equations of key stream generation equations become linear equations. With the addition of 66 original linear equations, the rest 243 bit can be obtained by gauss elimination, thus Trivium is broken.
**【Key words】**Trivium; chosen differential attack; guess attack

## 1 概述

流密码是一类重要的数据加密方法。Trivium[1]是 eSTREAM 工程的最终候选算法之一，eSTREAM 工程是欧洲 ECRYPT[2]的子项目。Trivium 使用密钥 $K$ 和一个初始化向量 $IV$ 产生一个长的伪随机序列，称为密钥流序列 $u$。密文是由明文和密钥做异或(XOR)得到的，即 $c=m+u$，其中，“+”表示异或运算。

Trivium 有 288 bit 内部状态和 80 bit 的密钥，虽然 Trivium 是基于硬件实现的，但是软件实现也很快。由于设计非常简单，因此成为最有竞争力的候选算法之一。文献[3]给出的猜测攻击结果是猜测 135 bit。文献[4]针对 Salsa 20 和 Trivium 提出了滑动攻击，指出 Trivium 有 $2^{39}$ 个滑动对。文献[5]提出的密钥差分攻击将穷举搜索的复杂度降低到 $2^{68}$。

## 2 Trivium算法

Trivium 使用 3 个非线性移位寄存器[6]，长度分别是 93 bit, 84 bit 和 111 bit。密钥流产生函数是若干比特异或，由 3 个移位寄存器的各 2 bit(共 6 bit)进行模二加法，寄存器 $i$ $(i=1,2,3)$ 的反馈函数由寄存器 $i$ 的二次函数和寄存器 $(i+1)\bmod 3$ 的一个比特确定。但这些反馈函数只包含一个二次项，其余都是线性项，而且，这个二次项很特殊，是 $S_jS_{j+1}$ 形式的。

### 2.1 相关符号

本文用 $s_1,s_2,\cdots,s_{288}$ 表示 Trivium 在 $t$ 时刻的 288 个内部状态，用 $\cdots z_{-5}z_{-4}\cdots z_0z_1z_2\cdots$ 表示 Trivium 由 $t$ 时刻的内部状态得到的密钥流，用“+”表示异或运算，即 (mod 2) 加法，用“×”表示与运算，由于只有 0 和 1，因此与乘法意义相同，此处“×”可省略。

### 2.2 内部状态的更新过程

根据文献[3]，Trivium 的状态转移如下：

$$(s_1\sim s_{93})_{t+1}=(s_{243}+s_{286}\times s_{287}+s_{288}+s_{69},s_1\sim s_{92})_t$$

$$(s_{94}\sim s_{177})_{t+1}=(s_{66}+s_{91}\times s_{92}+s_{93}+s_{171},s_{94}\sim s_{176})_t$$

$$(s_{178}\sim s_{288})_{t+1}=(s_{162}+s_{175}\times s_{176}+s_{177}+s_{264},s_{178}\sim s_{287})_t$$

容易得出内部状态的逆过程为

$$(s_1\sim s_{93})_t=(s_2\sim s_{93},s_{67}+s_{92}\times s_{93}+s_{94}+s_{172})_{t+1}$$

$$(s_{94}\sim s_{177})_t=(s_{95}\sim s_{177},s_{163}+s_{176}\times s_{177}+s_{178}+s_{265})_{t+1}$$

$$(s_{178}\sim s_{288})_t=(s_{179}\sim s_{288},s_{244}+s_{287}\times s_{288}+s_1+s_{70})_{t+1}$$

密钥流的生成如下：

$$z_t=s_{66}+s_{93}+s_{162}+s_{177}+s_{243}+s_{288}$$

可以看出，$t$ 时刻生成的密钥流比特 $z_t$ 是 $t$ 时刻内部状态 $(s_1,s_2,\cdots,s_{288})$ 的一个线性函数。从 $t$ 时刻到 $t$+1 时刻的内部状态更新是通过移位寄存器和非线性反馈实现的。

## 3 基于选择差分的猜测攻击

差分攻击的种类很多，常见的有传统差分攻击、截短差分攻击、高阶差分攻击等。

### 3.1 基本概念

**定义** 设 $f$ 是 $GF(2)^n\to GF(2)$ 的一个映射，$x_1,x_2,\cdots,x_n$ 是自变量，改变 $x_1,x_2,\cdots,x_n$ 中的若干分量值，得到 $x_1',x_2',\cdots,x_n'$，计算

**基金项目：**国家自然科学基金资助项目(60833008)；广西信息与通信重点实验室基金资助项目
**作者简介：**孙国平(1984－)，男，硕士研究生，主研方向：密码分析，密码算法；胡予濮，教授、博士生导师；白生江，硕士研究生
**收稿日期：**2009-11-23 **E-mail：**xdguoping@hotmail.com

$$\Delta f(x_1,x_2,\cdots,x_n)=f(x_1,x_2,\cdots,x_n)+f(x_1',x_2',\cdots,x_n')$$

其中，$x_i'=x_i$，$x_i'=x_i+1$，$i=1,2,\cdots,n$，则 $\Delta f$ 为选择差分。

### 3.2 选择差分攻击

选择差分攻击是指利用选择差分尽可能地降低目标函数或非线性方程的非线性度，从而使方程组可以求解，达到破解密码的目的。通过一次选择差分，至少能将函数的次数降低一次。

很多流密码的破解依赖于求解下列非线性方程组：

$$\pi:\begin{cases}f_1(s_1,s_2,\cdots,s_n)=0\\ f_2(s_1,s_2,\cdots,s_n)=0\\ \quad\cdots\\ f_m(s_1,s_2,\cdots,s_n)=0\end{cases}$$

一般此方程组的求解是一个 NP 困难问题，其可以表示成以下形式：

$$\begin{cases}a^1(0)+a^1(1)s_1+\cdots+a^1(n)s_n+a^1(1,2)s_1s_2+\cdots+\\ \qquad a^1(n-1,n)s_{n-1}s_n+a^1(1,2,3)s_1s_2s_3+\cdots+\\ \qquad a^1(1,2,\cdots,n)s_1s_2,\cdots,s_n=0\\ a^2(0)+a^2(1)s_1+\cdots+a^2(n)s_n+a^2(1,2)s_1s_2+\cdots+\\ \qquad a^2(n-1,n)s_{n-1}s_n+a^2(1,2,3)s_1s_2s_3+\cdots+\\ \qquad a^2(1,2,\cdots,n)s_1s_2,\cdots,s_n=0\\ \qquad\cdots\\ a^n(0)+a^n(1)s_1+\cdots+a^n(n)s_n+a^n(1,2)s_1s_2+\cdots+\\ \qquad a^n(n-1,n)s_{n-1}s_n+a^n(1,2,3)s_1s_2s_3+\cdots+\\ \qquad a^n(1,2,\cdots,n)s_1s_2,\cdots,s_n=0\end{cases}\tag{1}$$

其中，$a(0),a(1),\cdots,a(n),a(1,2),\cdots,a(n-1,n),\cdots,a(1,2,\cdots,n)$ 为系数，取值为 0 或 1。

重写方程如下：

$$\begin{cases}a^1(0)+a^1(1)s_1'+\cdots+a^1(n)s_n'+a^1(1,2)s_1's_2'+\cdots+\\ \qquad a^1(n-1,n)s_{n-1}'s_n'+a^1(1,2,3)s_1's_2's_3'+\cdots+\\ \qquad a^1(1,2,\cdots,n)s_1's_2',\cdots,s_n'=0\\ a^2(0)+a^2(1)s_1'+\cdots+a^2(n)s_n'+a^2(1,2)s_1's_2'+\cdots+\\ \qquad a^2(n-1,n)s_{n-1}'s_n'+a^2(1,2,3)s_1's_2's_3'+\cdots+\\ \qquad a^2(1,2,\cdots,n)s_1's_2',\cdots,s_n'=0\\ \qquad\cdots\\ a^n(0)+a^n(1)s_1'+\cdots+a^n(n)s_n'+a^n(1,2)s_1's_2'+\cdots+\\ \qquad a^n(n-1,n)s_{n-1}'s_n'+a^n(1,2,3)s_1's_2's_3'+\cdots+\\ \qquad a^n(1,2,\cdots,n)s_1's_2',\cdots,s_n'=0\end{cases}\tag{2}$$

设

$$T(i)=\begin{cases}1 & \text{if}\quad s_i'=s_i+1\\ 0 & \text{otherwise}\end{cases}\tag{3}$$

恰当地选取 $T(i)$ 值，代入式(2)，就得到一个关于 $s_1,s_2,\cdots,s_n$ 的方程组，然后与式(1)对应地进行差分，得到一个次数低并且含有更多线性方程的方程组。由于式(1)中的线性方程是不做差分的，因此从总体上得到了比原来多得多的线性方程。

### 3.3 针对Trivium的攻击

假设攻击者可以改变加密机中的 3 个非线性移位寄存器的有限比特，那么他先启动加密机，获得 $z_0,z_1,\cdots$，然后通过错误注入等物理手段改变寄存器的状态，再次启动加密机。

分析 Trivium 的密钥流生成方程，特别是这些方程中的非线性项，然后确定 $T(i)$=1 的 $i$ 值，原则是降低方程的非线性性，同时尽量得到内部状态的一个比特。$T(i)$=1 时 $i$ 的一种取值方式如图 1 所示，攻击者根据其改变寄存器的若干特定比特，将得到的密钥流和原始密钥流进行差分，从而得到 Trivium 的选择差分。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4 | 8 | 93 | 97 | 101 | 105 | 109 | 113 |
| 117 | 121 | 125 | 129 | 133 | 137 | 141 | 145 |
| 149 | 153 | 157 | 161 | 165 | 169 | 173 | 177 |
| 178 | 182 | 186 | 190 | 194 | 198 | 202 | 206 |
| 210 | 214 | 218 | 222 | 226 | 230 | 234 | 238 |
| 242 | 246 | 250 | 254 | 258 | 262 | 266 | 270 |
| 274 | 278 | 282 | 286 | | | | |

**图 1　$T(i)$=1 时 $i$ 的取值**

按上述方法改变 Trivium 中 $s_1,s_2,\cdots,s_{288}$ 的 52 个分量的值，得到关于 $s_1',s_2',\cdots,s_{288}'$ 的方程组，根据式(3)，将此方程组表示为 $s_1,s_2,\cdots,s_{288}$ 的方程，然后做差分。差分主要有 3 种情况：

(1)原来的方程都是线性方程，差分后的方程不含未知数，这些方程不做差分，例如：

$$s_{66}+s_{93}+s_{162}+s_{177}+s_{243}+s_{288}=z_0$$

$$s_{66}'+s_{93}'+s_{162}'+s_{177}'+s_{243}'+s_{288}'=z_0'$$

由式(3)得

$$s_{66}+(s_{93}+1)+s_{162}+(s_{177}+1)+s_{243}+s_{288}=z_0'$$

差分后得 $0=z_0+z_0'=z_0''$。

(2)原方程是非线性的，但差分后方程是线性的。

**例 1**

$$s_{243}+s_{288}+s_{286}\times s_{287}+s_{69}+s_{27}+s_{96}+s_{111}+s_{162}+s_{177}+s_{175}\times s_{176}+s_{264}+s_{222}=z_{66}$$

$$s_{243}'+s_{288}'+s_{286}'\times s_{287}'+s_{69}'+s_{27}'+s_{96}'+s_{111}'+s_{162}'+s_{177}'+s_{175}'\times s_{176}'+s_{264}'+s_{222}'=z_{66}'$$

由式(3)得

$$s_{243}+s_{288}+(s_{286}+1)\times s_{287}+s_{69}+s_{27}+s_{96}+s_{111}+s_{162}+(s_{177}+1)+s_{175}\times s_{176}+s_{264}+(s_{222}+1)=z_{66}'$$

差分后得 $s_{287}=z_{66}+z_{66}'=z_{66}''$。

**例 2**

$$s_{215}+s_{260}+s_{258}\times s_{259}+s_{242}+s_{287}+s_{285}\times s_{286}+s_{66}\times s_{67}+s_{146}+s_{56}+s_{83}+s_{81}\times s_{82}+s_{161}+s_{134}+s_{149}+s_{147}\times s_{148}+s_{236}+s_{194}=z_{94}$$

方法同上，最终得 $z_{94}''=s_{259}+s_{285}$。

(3)原来的方程是非线性的，差分后还是非线性的，例如：

$$s_{139}+s_{187}+s_{77}\times s_{78}+s_{124}+s_{205}+s_1+s_{247}+s_{226}+s_{52}+s_{232}+s_{157}+s_{169}+s_{90}\times s_{91}\times s_{171}+s_{65}\times s_{93}+s_{65}\times s_{171}+s_{28}+s_{64}+s_{91}+s_{79}+s_{181}+s_{94}+s_{26}\times s_{27}+s_{170}\times s_{171}+s_{92}\times s_{66}+s_{92}\times s_{93}+s_{92}\times s_{171}+s_{170}\times s_{66}+s_{170}\times s_{93}+s_{203}\times s_{204}+s_{11}\times s_{12}+s_{91}\times s_{92}+s_{90}\times s_{91}\times s_{66}+s_{65}\times s_{91}\times s_{92}+s_{90}\times s_{91}\times s_{92}+s_{65}\times s_{66}+s_{145}+s_{158}\times s_{159}+s_{137}\times s_{138}+s_{160}+s_{89}\times s_{90}+s_{90}\times s_{91}\times s_{93}+s_{170}\times s_{91}\times s_{92}+s_{230}\times s_{231}+s_{106}=z_{149}$$

差分后得 $z_{149}''=s_{92}+s_{170}+s_{231}+s_{138}+s_{90}\times s_{91}+s_{65}$。此时可以通过猜测一些比特使方程成为线性方程。

### 3.4 本文的猜测

为了表达方便，将方程 $z_i''=f(s_1,s_2,\cdots,s_{288})$ 简写为 $z_i''$，将 $z_i=f(s_1,s_2,\cdots,s_{288})$ 简写为 $z_i$，$i=\cdots,-2,-1,0,1,2,\cdots$。猜测 $s_{95}$，$s_{93}$，$s_{91},\cdots$，$s_{11}$，$s_1$，$s_{176}$ 这 45 bit，使方程 $z_{-3}'',z_{-2}'',z_{-1}'',z_{66}'',z_{67}'',z_{239}''$ 成为线性方程，再加上 $z_0,z_1,\cdots,z_{65}$ 这 66 个线性方程，使用高斯消元法即可解出其余 243 bit。

（下转第 133 页）