Improved Differential Fault Analysis of Trivium

Mohamed Saied Emam Mohamed*, Stanislav Bulygin†, Johannes Buchmann*
*Technische Universität Darmstadt
†Center for Advanced Security Research Darmstadt
CASED

COSADE 2011
Darmstadt, Germany, February 2011
Motivation
Description of Trivium
Differential fault analysis (DFA)
Algebraic Attack
Improved (DFA)
Our results
Conclusion
Motivation

Algebraic Cryptanalysis

Breaking a good cipher should require “as much work as solving a system of simultaneous equations in a large number of unknowns” (Shannon 1949)
Motivation

Algebraic Cryptanalysis

Breaking a good cipher should require “as much work as solving a system of simultaneous equations in a large number of unknowns” (Shannon 1949)

Differential fault analysis

An attacker can induce faults into a system, compare the correct and the faulty behavior, and use this information to retrieve the secret information inside the system.
Description of Trivium

- Hardware oriented stream cipher
- One of the selected eSTREAM project stream ciphers to the final portfolio
- Very fast and simple internal structure
- Secret key $K = (k_1, \ldots, k_{80})$
- Initial vector $IV = (v_1, \ldots, v_{80})$
- Inner state 288-bit stored in 3 shift registers, at time $t$

\[
(a_{t+1}, \ldots, a_{t+93}, b_{t+1}, \ldots, b_{t+84}, c_{t+1}, \ldots, c_{t+111})
\]

- Initial inner state

\[
(0, \ldots, 0, k_{80}, \ldots, k_1, 0, 0, 0, 0, v_{80}, \ldots, v_1, 1, 1, 1, 0, \ldots, 0)
\]
Description of Trivium

Inner state update

1. \( a_{i+93} = a_{i+24} + c_{i+45} + c_i + c_{i+1} \cdot c_{i+2}, \ i \geq 1 \)
2. \( b_{i+84} = b_{i+6} + a_{i+27} + a_i + a_{i+1} \cdot a_{i+2}, \ i \geq 1 \)
3. \( c_{i+111} = c_{i+24} + b_{i+15} + b_i + b_{i+1} \cdot b_{i+2}, \ i \geq 1 \)
Description of Trivium

Output keystream

- Initial phase, Trivium loops 1152 without producing keystream
- After the initial phase, Trivium starts to generate output keystream (time $t = 0$)
- Inner state at $t = 0$, $(a_1, \ldots, a_{93}, b_1, \ldots, b_{84}, c_1, \ldots, c_{111})$
- $z_i = a_i + a_{i+27} + b_i + b_{i+15} + c_i + c_{i+45}, i \geq 1$
Differential Fault Analysis

Fault analysis models

- Bit-failures in the data (memory cell)
- Faults affect the implementation code of the device
- Applying some faults to the internal registers of the device
  - Number of faults
  - Location
  - timing
- Attacker can rest the device to its initial state

Successful attacks

- Fault analysis of stream ciphers (Hoch and Shamir), CHES 2004
- Floating fault analysis of Trivium (Hojsík and Rudolf), IndoCrypt 2008
Attacker is able to inject a one-bit fault at a random position within the inner state.

**DFA attack assumption**

- Attacker can obtain up to \( n \) keystream bits \( z_t \) generated by the inner state \( (IS_t) \), \( 0 \leq t \leq n \)
- Attacker can inject one-bit fault into \( IS_0 \)
- Attacker can obtain both the cipher output before and after the fault injection \( z \) and \( z' \)
- Attacker can reset the cipher and repeat the previous two steps \( m \) times
Attack description

**Require** $m$ fault injection positions ($l_1, \ldots, l_m$) and the vectors

$$Z(z_1, \ldots, z_n), \ Z^{(i)}(z^{(1)}, \ldots, z^{(n)}), \ 1 \leq i \leq m$$

$$P \leftarrow \text{EQgenerator}(l_1, \ldots, l_m, Z, Z^{(1)}, \ldots, Z^{(m)})$$

$$IS \leftarrow \text{extract} \ (a_1, \ldots, a_{n+93}, b_1, \ldots, b_{n+84}, c_1, \ldots, c_{n+111})$$

Recover the secret key $K$ from $IS$

**return** $K$
Differential Fault Analysis of Trivium

The DFA equations were initially introduced by Hojsík-Rudolf, IndoCrypt 2008.

\[
\begin{align*}
\Delta a_i + 93 &= \Delta a_i + 24 + \Delta c_i + 45 + \Delta (c_i + 1 \cdot c_i + 2) \\
\Delta b_i + 84 &= \Delta b_i + 6 + \Delta a_i + 27 + \Delta (a_i + 1 \cdot a_i + 2) \\
\Delta c_i + 111 &= \Delta c_i + 24 + \Delta b_i + 15 + \Delta (b_i + 1 \cdot b_i + 2) \\
\Delta z_i &= \Delta a_i + \Delta a_i + 27 + \Delta b_i + \Delta b_i + 15 + \Delta c_i + \Delta c_i + 45, \quad i \geq 1
\end{align*}
\]
DFA additional equations

- The DFA equations were initially introduced by Hojsík-Rudolf, IndoCrypt 2008
- $\triangle a_{i+93} = \triangle a_{i+24} + \triangle c_{i+45} + \triangle c_i + \triangle (c_{i+1} \cdot c_{i+2})$
- $\triangle b_{i+84} = \triangle b_{i+6} + \triangle a_{i+27} + \triangle a_i + \triangle (a_{i+1} \cdot a_{i+2})$
- $\triangle c_{i+111} = \triangle c_{i+24} + \triangle b_{i+15} + \triangle b_i + \triangle (b_{i+1} \cdot b_{i+2})$
The DFA equations were initially introduced by Hojsík-Rudolf, IndoCrypt 2008.

\[ \Delta a_{i+93} = \Delta a_{i+24} + \Delta c_{i+45} + \Delta c_i + \Delta (c_{i+1} \cdot c_{i+2}) \]

\[ \Delta b_{i+84} = \Delta b_{i+6} + \Delta a_{i+27} + \Delta a_i + \Delta (a_{i+1} \cdot a_{i+2}) \]

\[ \Delta c_{i+111} = \Delta c_{i+24} + \Delta b_{i+15} + \Delta b_i + \Delta (b_{i+1} \cdot b_{i+2}) \]

\[ \Delta z_i = \Delta a_i + \Delta a_{i+27} + \Delta b_i + \Delta b_{i+15} + \Delta c_i + \Delta c_{i+45}, \quad i \geq 1 \]
Differential Fault Analysis of Trivium

Equation system describing $n$ keystream bits

\[
\text{TRIV}(Z = (z_1, \ldots, z_n))
\]

\[
\text{for } i = 1 \text{ to } n
\]

\[
P \leftarrow P \cup \{z_i + a_i + a_{i+27} + b_i + b_{i+15} + c_i + c_{i+45} = 0\}
\]

\[
P \leftarrow P \cup \{a_{i+93} + a_{i+24} + c_{i+45} + c_i + c_{i+1} \cdot c_{i+2} = 0\}
\]

\[
P \leftarrow P \cup \{b_{i+84} + b_{i+6} + a_{i+27} + a_i + a_{i+1} \cdot a_{i+2} = 0\}
\]

\[
P \leftarrow P \cup \{c_{i+111} + c_{i+24} + b_{i+15} + b_i + b_{i+1} \cdot b_{i+2} = 0\}
\]

\[
\text{return } P
\]
Generating additional DFA equations

\textbf{EQgenerator\((l_1, \ldots, l_m, Z, Z^{(1)}, \ldots, Z^{(m)})\)}

\begin{align*}
P & \leftarrow \text{TRIV}(Z) \\
\text{for } j = 1 \text{ to } m & \\
\text{InjectFault} & ((a_1, \ldots, a_{93}, b_1, \ldots, b_{84}, c_1, \ldots, c_{111}), 1 \leq l_j \leq 288) \\
\text{for } i = 1 \text{ to } n & \\
\triangle a_{i+93} & = \triangle a_{i+24} + \triangle c_{i+45} + \triangle c_i + \triangle (c_{i+1} \cdot c_{i+2}) \\
\triangle b_{i+84} & = \triangle b_{i+6} + \triangle a_{i+27} + \triangle a_i + \triangle (a_{i+1} \cdot a_{i+2}) \\
\triangle c_{i+111} & = \triangle c_{i+24} + \triangle b_{i+15} + \triangle b_i + \triangle (b_{i+1} \cdot b_{i+2}) \\
\triangle z_i & \leftarrow z_i + z_i^{(j)} \\
P & \leftarrow \triangle z_i + \triangle a_i + \triangle a_{i+27} + \triangle b_i + \triangle b_{i+15} + \triangle c_i + \triangle c_{i+45} = 0 \\
S & \leftarrow \text{CyclicSubstitute}(P) \\
& \text{Substitute}(\triangle A, \triangle B, \triangle C, A, B, C, S) \\
\text{return } P
\end{align*}

Note: \(\triangle (a_{i+1} \cdot a_{i+2}) = \triangle a_{i+1} \cdot a_{i+2} + a_{i+1} \cdot \triangle a_{i+2} + \triangle a_{i+1} \cdot \triangle a_{i+2}\)
## Equation system specification

<table>
<thead>
<tr>
<th>Generator</th>
<th>$m$</th>
<th>degree 1</th>
<th>degree 2</th>
<th>degree 3</th>
<th>degree 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>H-R</td>
<td>0</td>
<td>800</td>
<td>2400</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>H-R</td>
<td>1</td>
<td>825</td>
<td>2466</td>
<td>35</td>
<td>57</td>
</tr>
<tr>
<td>H-R</td>
<td>2</td>
<td>1017</td>
<td>2419</td>
<td>36</td>
<td>57</td>
</tr>
<tr>
<td>H-R</td>
<td>3</td>
<td>1258</td>
<td>2298</td>
<td>37</td>
<td>56</td>
</tr>
<tr>
<td>H-R</td>
<td>4</td>
<td>2402</td>
<td>498</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>our</td>
<td>0</td>
<td>800</td>
<td>2400</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>our</td>
<td>1</td>
<td>994</td>
<td>2394</td>
<td>28</td>
<td>27</td>
</tr>
<tr>
<td>our</td>
<td>2</td>
<td>1212</td>
<td>2362</td>
<td>64</td>
<td>76</td>
</tr>
<tr>
<td>our</td>
<td>3</td>
<td>1619</td>
<td>1990</td>
<td>82</td>
<td>66</td>
</tr>
<tr>
<td>our</td>
<td>4</td>
<td>2688</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Differential Fault Analysis of Trivium

Number of solved variables

![Graph showing number of solved variables for m=1 and m=2](image-url)
Algebraic Attacks

The MP Problem

- Given finite set of multivariate polynomials $P$ in $X = \{x_1, \ldots, x_n\}$ over finite field $F$
- Find $v \in F^n$, $p(v) = 0 \ \forall p \in P$, for example:
  - $x_1x_2 + x_1x_3 + x_2x_3 = 0$
  - $x_1x_3 + x_2x_3 + x_1 + 1 = 0$
  - $x_1x_2 + x_1x_3 + x_2x_3 + x_1 + x_2 + 1 = 0$

- General MP is NP-hard even if $P$ is over $\mathbb{F}_2$ and quadratic
- Hard in average
Algebraic Attacks

Attacking Block Cipher

- Using a pair of (known) plaintext-ciphertext values, the secret key and a large number of intermediate variables arising in the cipher
- Solving the resulting multivariate system recovers the secret key

Attacking Stream Cipher

Set up a system of polynomial equations in secret key $K$, intermediate variables $X$ and known keystream bits $z_1(f(K, X, Z) = 0, \cdots, f_N(K, X, Z) = 0$.

Solving the multivariate system to get $K$. 

Mohamed Saied Emam Mohamed
Improved Differential Fault Analysis of Trivium
Algebraic Attacks

Attacking Block Cipher

- Using a pair of (known) plaintext-ciphertext values, the secret key and a large number of intermediate variables arising in the cipher
- Solving the resulting multivariate system recovers the secret key

Attacking Stream Cipher

- Set up a system of polynomial equations in secret key $K$, intermediate variables $X$ and known keystream bits $z_t$

$$f_1(K, X, Z) = 0, \cdots, f_N(K, X, Z) = 0$$
- Solving the multivariate system to get $K$
Algebraic attack on Trivium

Algebraic attack steps

- \( P \leftarrow \text{TRIV}(Z = (z_1, \ldots, z_n)) \): return a system of \( 4n \) equations (\( n \) linear and \( 3n \) quadratic) in \( 3n + 288 \) variables
- \( P \leftarrow \text{Solve}(P) \)
- Recover the inner state \( (a_1, \ldots, a_{93}, b_1, \ldots, b_{84}, c_1, \ldots, c_{111}) \)
- Clock Trivium backward to recover the secret key \( k_1, \ldots, k_{80} \)

Best known algebraic attack on Trivium

- Attacking Bivium using SAT solver (\( 2^{43} \) s), SAT 2008
- Attacking Bivium using Grain of salt tool (\( 2^{36.5} \) s), Mate Soos
- Cube attack on scaled version of Trivium (672 and 735 initialization rounds), Dinur and Shamir
Algebraic Attacks

Tools for Solving MP

- Computing Gröbner basis
  Gröbner basis algorithms are the best known techniques for solving multivariate systems

- Cube attack
  A method of cryptanalysis applicable to a wide variety of symmetric-key algorithms, published by Dinur and Shamir

- SAT solver-based attacks
  SAT solvers are used in algebraic attacks specially to stream ciphers in recent years
SAT Solvers

The Satisfiability Problem

Given a Boolean propositional formula, does there exist assignment of variables such that the formula becomes true.
SAT Solvers

The Satisfiability Problem
Given a Boolean propositional formula, does there exist assignment of variables such that the formula becomes true

SAT Solver-Based Attacks

- Construct a system of equations in the ANF form describing the cipher
- Convert ANF to CNF
- Use a SAT solver to find a solution to the problem
  - MiniSat, PrecoSat, CryptoMiniSat, ...

Successful attacks
- Cryptanalysis of stream cipher HiTag2 ($2^{21}$ s), ISC 2009
- Cryptanalysis of Crypto-1 (40 s) using Grain of salt tool, Mate Soos

Improved Differential Fault Analysis of Trivium 19
The Satisfiability Problem

Given a Boolean propositional formula, does there exist assignment of variables such that the formula becomes true

SAT Solver-Based Attacks

- Construct a system of equations in the ANF form describing the cipher
- Convert ANF to CNF
- Use a SAT solver to find a solution to the problem
  - MiniSat, PrecoSat, CryptoMiniSat, ...

Successful attacks

- Cryptanalysis of stream cipher HiTag2 (2^{21} s), ISC 2009
- Cryptanalysis of Crypto-1 (40 s) using Grain of salt tool, Mate Soos
SAT Solvers

**ANF to CNF**

- A polynomial $p$ in the ANF form is a sum of products (terms)
  \[ p = (x \cdot y + x \cdot z + y \cdot z + y \cdot w + z \cdot w + x + z + y + 1) \]
- A polynomial $p$ in the CNF form is a set of clauses ($\land, \lor, \neg$)
- Replace nonlinear terms with new variables
  \[ (t = x \cdot y) \equiv (t \iff x \land y) \equiv \left( \overline{b_1} \lor x \right) \land \left( \overline{b_1} \lor y \right) \land \left( b_1 \lor \overline{x} \lor \overline{y} \right) \]
  \[ t_1 + t_2 \equiv \left( b_1 \lor \overline{b_2} \right) \land \left( \overline{b_1} \lor b_2 \right) \land \left( b_1 \lor \overline{b_2} \right) \]
  \[ 1 \equiv (b \lor \overline{b}) \]
- Cutting number $c$, splitting polynomials into sub-polynomials that have at most $c$ terms, for example let $c = 6$, 
  \[ p = (x \cdot y + x \cdot z + y \cdot z + y \cdot w, z \cdot w + b), \ b = (x + z + y + 1) \]
Improved DFA of Trivium

**Attack description**

**Require** $m$ fault injection positions ($l_1, \ldots, l_m$) and the vectors $Z(z_1, \ldots, z_n)$, $Z^{(i)}(z^{(1)}, \ldots, z^{(n)})$, $1 \leq i \leq m$

$P \leftarrow \text{EQgenerator}(l_1, \ldots, l_m, Z, Z^{(1)}, \ldots, Z^{(m)})$

$\text{CNF}(P) \leftarrow$ Converting $P$ to a satisfiability problem in the CNF form

$\text{Solution} \leftarrow$ Solve the satisfiability problem $\text{CNF}(P)$ by using MiniSat2

$IS \leftarrow \text{extract } (a_1, \ldots, a_{n+93}, b_1, \ldots, b_{n+84}, c_1, \ldots, c_{n+111})$

Recover the secret key $K$ from $IS$

return $K$
Improved DFA of Trivium

Results of Hojsík-Rudolf

For $N = 800$, the attacker succeeds with probability:
- $2\%$ when $m = 2$, $78.5\%$ when $m = 3$, $99.8\%$ when $m = 4$
- $100\%$ when $m \geq 5$
Results of Hojsík-Rudolf
For $N = 800$, the attacker succeeds with probability:
- 2% when $m = 2$, 78.5% when $m = 3$, 99.8% when $m = 4$
- 100% when $m \geq 5$

Our results
For $N \geq 420$, the attacker succeeds with probability:
- 100% when $m = 2$
## Results of using MiniSat2

<table>
<thead>
<tr>
<th>$n$</th>
<th>$m$</th>
<th>degree 1</th>
<th>degree 2</th>
<th>time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>2</td>
<td>1216</td>
<td>2365</td>
<td>0.261</td>
</tr>
<tr>
<td>700</td>
<td>2</td>
<td>1088</td>
<td>2080</td>
<td>0.356</td>
</tr>
<tr>
<td>600</td>
<td>2</td>
<td>1005</td>
<td>1771</td>
<td>0.414</td>
</tr>
<tr>
<td>500</td>
<td>2</td>
<td>890</td>
<td>1437</td>
<td>0.127</td>
</tr>
<tr>
<td>450</td>
<td>2</td>
<td>831</td>
<td>1230</td>
<td>1.573</td>
</tr>
<tr>
<td>430</td>
<td>2</td>
<td>799</td>
<td>1138</td>
<td>1.936</td>
</tr>
<tr>
<td>420</td>
<td>2</td>
<td>769</td>
<td>1117</td>
<td>138.653</td>
</tr>
</tbody>
</table>
Conclusion and Future work

- We used advanced algebraic method (SAT solver) to improve the DFA of Trivium.
- Recover the secret key 100% using two fault injections and 420 keystream output bits.
- Improve our attack to recover the secret key of Trivium using only one fault injection by applying more advanced conversion techniques and tuning SAT solver parameters.
Acknowledgments

We want to thank the useful comments from

- Anonymous referees of the COSADE workshop
- Marcel Medwed

on this paper and their valuable suggestions which helped to improve the paper
Thanks for your attention!