A Key-recovery Attack on 855-Round Trivium

Ximing Fu, Xiaoyun Wang, **Xiaoyang Dong**, Willi Meier

Tsinghua University, Beijing, China
FHNW, Windisch, Switzerland

June 6,2018

Outline

Trivium

- Initialization:

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_0, \ldots, K_{79}, 0, \ldots, 0)$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_0, \ldots, IV_{79}, 0, \ldots, 0)$
$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1).$
for $i \leftarrow 1 : 4 \cdot 288$ **do**
$t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$
$t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$
$t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$
$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
end for

Trivium

Generate the keystreams:

for $i \leftarrow N$ **do**

$t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$

$t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$

$t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$

$o_i \leftarrow s_{66} + s_{93} + s_{162} + s_{177} + s_{243} + s_{288}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for

Trivium

Iterative expression: let s_w^r $(0 \leq w \leq 2)$ denote s_1, s_{94} and s_{178} at round r.

$$\begin{aligned} s_0^r &= s_2^{r-66} + s_2^{r-109} s_2^{r-110} + s_2^{r-111} + s_0^{r-69}, \\ s_1^r &= s_0^{r-66} + s_0^{r-91} s_0^{r-92} + s_0^{r-93} + s_1^{r-78}, \\ s_2^r &= s_1^{r-69} + s_1^{r-82} s_1^{r-83} + s_1^{r-84} + s_2^{r-87}. \end{aligned} \tag{1}$$

Output: $z_r = s_0^{r-65} + s_0^{r-92} + s_1^{r-68} + s_1^{r-83} + s_2^{r-65} + s_2^{r-110}$

Outline

Cube-like Attack

ANF: The output bit or state bit for a stream cipher over m IV bits and n key bits is

$$s = \sum_{I,J} \prod_{i \in I} v_i \prod_{j \in J} k_j. \tag{2}$$

IV term: $t_I = \prod_{i \in I} v_i$
Coefficient function: $g_I(k) = \prod_{j \in J} k_j$

Theorem 1

Cube sum of s over set I is $g_I(k)$, i.e.,

$$\sum_{i \in I} s = g_I(k), \tag{3}$$

where the IV bits v_k $(k \notin I)$ are fixed.

1. $g_I(k)$ is linear or of low degree over partial key bits (key-recovery)
2. $g_I(k) = 0$: $t_I(k)$ is a missing IV term (distinguisher)

Outline

A new polynomial reduction technique

Lemma 2

Suppose z is the output polynomial of a cipher, and

$$z = P_1 P_2 + P_3. \tag{4}$$

Then the polynomial can be reduced to a simpler one $(1 + P_1)z = (1 + P_1)P_3$ by multiplying $1 + P_1$ in both sides of Eq. (4) if $\deg(P_1 P_2) > \deg((1 + P_1)P_3)$.

How to distinguish right and wrong key guesses

1. Right guess: $(1 + P_1)z = (1 + P_1)P_3$
2. Wrong guesses: $(1 + P_1')z = (1 + P_1')P_1 P_2 + (1 + P_1')P_3$

Outline of our attack

Preprocess phase

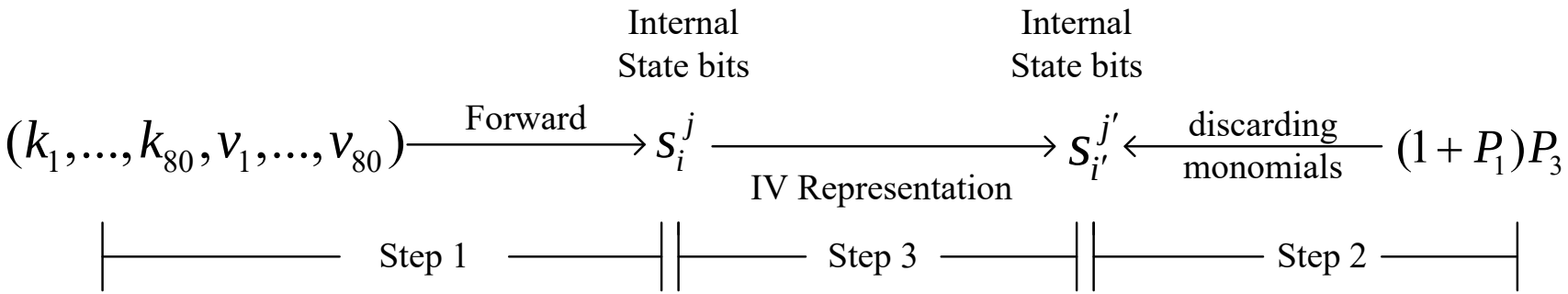
1. Determine P_1 and obtain the reduced polynomial $(1+P_1)P_3$. There are 3 criteria for choice of P_1: (1) the frequency of P_1 in high degree state terms is high; (2) the degree of P_1 is low; (3) the equivalent key guesses in P_1 are minimized.
2. Compute the degree bound of $(1+P_1)P_3$ as d, then $d+1$-dimensional cubes can serve as distinguishers.

Online attack phase

Guess the partial key bits in P_1 and compute the sum of $(1+P_1)z$ over $d+1$ cubes:

1. For right guess, the result is always 0.
2. For wrong guesses, the results are 0-1 balanced.

The preprocessing phase



1. Compute the state bits s_i^j $(j \in [0,2])$ for $i \in [0,340]$ over key and IV bits.
2. Decompose the output bit and obtain $(1+P_1)P_3$ over state bits at rounds less than 450.
3. "Meet-in-the-middle": decomposition & IV representation

Key techniques

In **Step 2** and **Step 3**, repeated-term removing algorithm and fast discarding techniques are used during decomposition, including degree evaluation and degree reduction techniques, set a bound d:

1. if the evaluated degree of a state term $\deg T_i$, then T_i can be deleted;
2. if $\deg(T_i) - d_t(T_i) < d$, then T_i can be deleted, where $d_t(T_i)$ is the degree reduction of T_i.

Repeated-(state)term Removing Algorithm

Algorithm 1 Repeated-(state)term Removing Algorithm

Input: The vector $\vec{T}$ with n terms, i.e., T_1, T_2, ..., T_n.
Output: Updated $\vec{T}$ with m terms, where $m \leq n$.

1: Initialize an empty Hash Set $\mathbf{H}$.
2: **for** $i \leftarrow 1 : n$ **do**
3: Compute the Hash value of T_i, i.e., $H(T_i)$
4: **if** $H.contains(T_i)$ is **true then**
5: $H.delete(T_i)$
6: **else**
7: $H.insert(T_i)$
8: **end if**
9: **end for**

The complexity of Algorithm 1 is $O(n)$ for processing n state terms.

Degree evaluation algorithm

Algorithm 2 Degree Evaluation Algorithm ($\mathcal{DEG}$) of State Bit

Input: The value t and r which indicates the state bit s_t^r.
Output: $\mathcal{DEG}(s_t^r)$=d.

1: Initialize the degree bound d similar to the above **Step 2.**, the end point end.
2: $len \leftarrow 0$
3: **while** $len = 0$ **do**
4: Iteratively express s_t^r using state bits s_i^j, where $0 \leq j \leq 2$ and $0 \leq j < end$. During each expression, discard the state terms of degree lower than d. Let len be the number of remaining state terms.
5: **if** $len = 0$ **then**
6: $d \leftarrow d - 1$
7: **end if**
8: **end while**
9: **return** d

Where $end = \lfloor \frac{r}{32} \rfloor \times 32 - 128$ in the cryptanalysis of Trivium.

Degree evaluation: example

Degree evaluation of s_1^{341} $(end = \lfloor \frac{r}{32} \rfloor \times 32 - 128 = 192)$:

- **Step 1.** First, we decompose $s_2^{341} = s_1^{272} + s_1^{259}s_1^{258} + s_1^{257} + s_2^{254}$.
- **Step 2.** Let $d = \mathtt{max}\{\deg(s_1^{272}), \deg(s_1^{259}) + \deg(s_1^{258}), \deg(s_1^{257}), \deg(s_2^{254}\} = 10$.
- **Step 3.** Discarding the state terms of degree lower than 10, we get $s_2^{341*} = s_1^{259}s_1^{258}$. Decompose and discard again, there is no state term surviving. Reset $d = d - 1 = 9$ and repeat the above process. We can get the result $s_2^{341**} = s_0^{166}s_0^{167}s_0^{193} + s_0^{167}s_0^{168}s_0^{192} + ...$
- **Step 4.** Continue to decompose and discard, and we get:

$$s_2^{341***} = s_2^{56}s_2^{57}s_2^{83}s_2^{84}s_2^{101} + s_2^{57}s_2^{58}s_2^{83}s_2^{84}s_2^{100} + ... \quad (5)$$

- **Step 5.** The decomposition ends and there are still state terms surviving. $d = 9$ is the estimated degree of s_2^{341}.
- **Step 6.** Note that, if there is no state item in s_2^{341***} surviving, which means the degree must be less than 9. We reset $d = 8$ and continue the above steps 3-5.

Degree reduction algorithm

Algorithm 3 Degree Evaluation Algorithm ($\mathcal{DEG}$) of State Bit

Input: The value i, r, t which indicates the state term degree reduction.
Output: The degree reduction $d_t = \sum_{j=l}^{l+t-1} \deg(s_i^j) - \deg(\prod_{j=l}^{l+t-1} s_i^j)$.

1: Initialize the degree bound $d = \sum_{i=l}^{l+t-1} \mathcal{DEG}(s_i^j)$, degree reduction $d_t = 0$, end point end and number of survived state terms len.
2: **while** $len = 0$ **do**
3: Express the state term $\prod_{j=l}^{l+t-1} s_i^j$ using state bits s_i^j, where $0 \leq i \leq 2$ and $0 \leq j < end$, discard the state terms of degree lower than $d - d_t$. Let len be the number of remaining state terms.
4: **if** $len = 0$ **then**
5: $d_t \leftarrow d_t + 1$
6: **end if**
7: **end while**
8: **return** d_t

Where $end = \lfloor \frac{r}{32} \rfloor \times 32 - 128$ in the cryptanalysis of Trivium.

Degree reduction: example

Degree reduction of $s_1^{340}s_1^{341}(end = \lfloor \frac{r}{32} \rfloor \times 32 - 128 = 192)$:

- Initialize $d = \mathcal{DEG}(s_1^{340}) + \mathcal{DEG}(s_1^{341})$ and $d_t = 0$.
- Express the $s_1^{340}s_1^{341}$, discard the state terms of degree lower than $d - d_t = d$, there is no state term surviving.
- Increase the d_t by 1, such that $d_t = 1$.
- Express $s_1^{340}s_1^{341}$ again and discard the state terms of degree lower than $d - d_t = d - 1$, the result is $s_0^{249}s_0^{250}s_1^{262} + s_0^{248}s_0^{249}s_1^{263}$.
- Continue to compute iteratively, the remaining state terms are $s_0^{170}s_0^{171}s_0^{180}s_2^{140}s_2^{141} + s_0^{170}s_0^{171}s_0^{181}s_2^{139}s_2^{140} + s_0^{171}s_0^{172}s_0^{179}s_2^{139}s_2^{140} + s_0^{171}s_0^{172}s_0^{180}s_2^{138}s_2^{139}$. There is no state bits s_i^j with j bigger than $end = 192$ in all the state terms, hence the expression ends.
- Degree reduction $d_t = 1$ is returned. Thus $\deg(s_1^{340}s_1^{341}) \leq \mathcal{DEG}(s_1^{340}) + \mathcal{DEG}(s_1^{341}) - d_t = 7 + 7 - 1 = 13$.

IV representation

Definition 3

Given a Boolean polynomial $s = \sum_{I,J} \prod_{i \in I} v_i \prod_{j \in J} k_j$, the corresponding IV representation is $s_{IV} = \sum_{I,J} \prod_{i \in I} v_i$.

Example 4

For $s = v_0k_1 + v_0k_0k_2 + v_1k_1k_2 + v_0v_1k_2$, the representation is $s_{IV} = v_0 + v_0 + v_1 + v_0v_1$

Property 1

If an IV term exists in s, it must also exist in s_{IV}, but not the opposite. If an IV term is not in s_{IV}, it can be concluded that it is not in s.

Using IV representation can compute the missing IV terms, which can serve as distinguishers.

Repeated-IV term Removing Algorithm

Algorithm 4 Repeated-IV term Removing Algorithm

Input: The vector $\vec{T}$ with n IV terms, i.e., T_1, T_2, ..., T_n.
Output: Updated $\vec{T}$ with m IV terms, where $m \leq n$.

1: Initialize an empty Hash set $\mathbf{H}$.
2: **for** $i \leftarrow 1 : n$ **do**
3: Compute the Hash value of T_i, i.e., $H(T_i)$.
4: **if** $H.contains(T_i)$ is **false then**
5: $H.insert(T_i)$.
6: **end if**
7: **end for**

The time complexity is $O(n)$ for processing n IV terms.

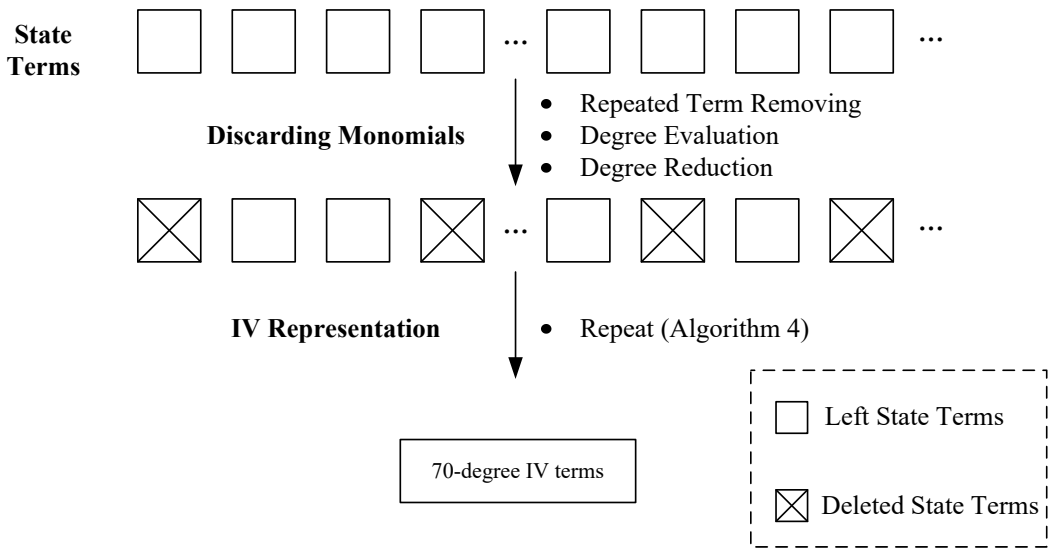
Outline

Attack on Trivium

Compute the exact Boolean polynomial of state bits s_w^r ($w \in [0, 2]$) for $r \leq 340$ and obtain the degree bound of the other state bits by applying Algorithm 2.

- Determine $P_1 = s_1^{210}$: decompose the output bit of 855-round Trivium and preserve the high degree state terms (1) s_1^{210} occurs in about $\frac{3}{4}$ of all the preserved high state terms; (2) the degree of s_1^{210} is 5 and can be reduced to 2 after nullifying the 5 IV bits; (3) there are only 3 equivalent key bits to be guessed.
- Nullify 5 IV bits to reduce the degree of s_1^{210} and update the Boolean polynomials and degrees of state bits.
- Determine the key bits in P_1, i.e., k_{19}, k_{20}, $k_{57} + k_{63} + k_{21} + k_{28}k_{29} + k_3 + k_{30} + k_{12} + k_{37}k_{38} + k_{39}$.

Preprocessing Phase



- degree evaluation: remove the state terms of degree lower than 70
- degree reduction: remove the state terms of degree lower than $d < 70 + d_t$, where d_t is the corresponding degree reduction for state terms
- IV representation: compute the existent 70-degree IV terms

It is proved that $\deg((1 + s_1^{210})z_{855}) < 70$.

Online Phase

Algorithm 5 On-line Attack

1: Initialize the possible key space KEY with size of 2^3.
2: **for** $i \leftarrow 1:3$ **do**
3: **for** Each possible key in KEY **do**
4: Compute the value s_1^{210}, so that obtain the value of $(1+s_1^{210})z$,
5: Compute cube sums z_{sum} of $(1+s_1^{210})z$,
6: **if** $z_{sum}=1$ **then**
7: Delete key from KEY.
8: **end if**
9: **end for**
10: **end for**

Complexity analysis: the time complexity is $(2^3+2^2+2^1)2^{70} \approx 2^{74}$ bit operations.

Thanks for Your Attention