Algebraic Attack Against Trivium

Ilaria Simonetti, Ludovic Perret and Jean Charles Faugère

Abstract. Trivium is a synchronous stream cipher designed to provide a flexible trade-off between speed and gate count in hardware, and reasonably efficient software implementation. It was designed in 2005 by C. De Cannière and B. Preneel for the European project eSTREAM. It has successfully moved into phase two of the selection process and is currently in the focus group under the hardware category. As of yet there has been no attack on Trivium faster than exhaustive search.
Bivium-A and Bivium-B are truncated versions of Trivium that are built on the same design principles. These simplified versions are used for investigating Trivium-like ciphers with a reduced complexity. There have been successful attempts in the cryptanalysis of Bivium ciphers.
The goal of this paper is to compare a basic Gröbner basis attack against these ciphers with other known methods. To do so, we present some experimental results.

1. Introduction

Stream ciphers can be viewed as approximating the action of a theoretically unbreakable cipher, the one-time pad (OTP). A one-time pad uses a keystream of completely random digits. The keystream is combined with the plaintext digits one at a time to form the ciphertext. This system was proved to be theoretically secure by Shannon in 1949. However, the keystream must be (at least) the same length as the plaintext, and generated completely at random. This makes the system very cumbersome to implement in practice, and as a result the one-time pad has not been widely used, except for the most critical applications.
A stream cipher makes use of a much smaller and more convenient key. Based on this key, it generates a pseudorandom keystream which can be combined with the plaintext digits in a similar fashion to the one-time pad. However, this comes at a cost: because the keystream is now pseudorandom, and not truly random, the proof of security associated with the one-time pad no longer holds: it is quite possible for a stream cipher to be completely insecure.

The eStream project [7] was actuated for fixing a standard stream cipher. There are two main categories in the call - software encryption and hardware encryption. The two main goals stated in the specification criteria are that the cipher is secure and fast. In the first phase 34 proposals were received, but only a few of them got the status of "focused algorithm" in the second phase. The call is now in phase three of the evaluation process.
We analyze Trivium, which is one of the algorithm which has been advanced to phase 3 of eSTREAM. It has an internal state of 288 bits and the key of 80 bits. Its design is simple and elegant. Now it is still unbroken. The structure of Trivium can be expressed as a system of sparse equations over $\mathbb{F}_2$.
In this paper, after giving a description of the design of Trivium and its two variants, Bivium A and B, we summarize known attacks against them. Then we show how generate a system of equations over $\mathbb{F}_2$ for Trivium and Bivium. We use two method: in the first we add three variables (or two for Bivium)for each clock of the cipher; in the second one we use as variables only the 288 bits (or 177 bits for Bivium) of the internal state at the beginning. In the last section we use these two approach and we compute the Gröbner basis of the system. We give some experimental complexity results, which are comparable with the previous known results, but further works deserve may be to be done.

2. Trivium and its variant

Trivium consists of a non linear feedback shift register, which operates on a 288-bit state denoted by $(s_1, \ldots, s_{288})$, put together a linear filter function. The linear filter function takes a linear combination of the state to produce the keystream. At each clock Trivium updates three bits of the state and output one bit of keystream. Trivium is designed to generate up to 2^{64} bits of keystrem from an 80-bit secret key and an 80-bit initial value (IV). As for most stream cipher this process consists of two phases:

- first, the internal state of the cipher is initialized using the key and the IV,
- then the state is repeatedly updated and used to generate keystream bits.

Our analysis does not use the initialization process, so we describe only the keystream generation.
The register is divided in three registers of length 93, 84 and 111 bits:

$$(s_1, \ldots, s_{288}) = (s_1, \ldots, s_{93})|(s_{94}, \ldots, s_{177})|(s_{178}, \ldots, s_{288})\,.$$

Trivium operates on these three registers as shown by the following pseudo-code:

for $i = 1$ to N **do**

$$\begin{aligned}
t_1 &\leftarrow s_{66} + s_{93} \\
t_2 &\leftarrow s_{162} + s_{177} \\
t_3 &\leftarrow s_{243} + s_{288} \\
z_i &\leftarrow t_1 + t_2 + t_3 \\
t_1 &\leftarrow t_1 + s_{91} \cdot s_{92} + s_{171} \\
t_2 &\leftarrow t_2 + s_{175} \cdot s_{176} + s_{264} \\
t_3 &\leftarrow t_3 + s_{286} \cdot s_{287} + s_{69} \\
(s_1, s_2, \cdots, s_{93}) &\leftarrow (t_3, s_1, \ldots, s_{92}) \\
(s_{94}, s_{95}, \cdots, s_{177}) &\leftarrow (t_1, s_{94}, \ldots, s_{176}) \\
(s_{178}, s_{179}, \cdots, s_{288}) &\leftarrow (t_2, s_{178}, \ldots, s_{287})
\end{aligned}$$

Note that here, and in the rest of the document, the '+' and '·' operations stand for sum and product over $\mathbb{F}_2$.

2.1. Bivium A and B

Bivium is a truncated version of Trivium, which has two other variants: variant A and B. It uses one register of 177 bits divided in two blocks of 93 and 84 bits. At each clock it updates two bits of the state and outputs one bit of keystream, as follows:

for $i = 1$ to N **do**

$$\begin{aligned}
t_1 &\leftarrow s_{66} + s_{93} \\
t_2 &\leftarrow s_{162} + s_{177} \\
z_i &\leftarrow t_2 \text{ (Variant A) / } t_1 + t_2 \text{ (Variant B)} \\
t_1 &\leftarrow t_1 + s_{91} \cdot s_{92} + s_{171} \\
t_2 &\leftarrow t_2 + s_{175} \cdot s_{176} + s_{69} \\
(s_1, s_2, \cdots, s_{93}) &\leftarrow (t_2, s_1, \ldots, s_{92}) \\
(s_{94}, s_{95}, \cdots, s_{177}) &\leftarrow (t_1, s_{94}, \ldots, s_{176})
\end{aligned}$$

3. Known attacks

There exists three known attacks against Trivium and its two variants. The first one use the sparsity of the sistem generated by the cipher, in the second one they try to obtain as linear equation as possible guessing the value of the state bits and the third one use the SatSolver.

3.1. Graph for sparse system

If a system is sparse, there exists an efficient method to solve it building a graph and inspecting the tree structure generated by it [1]. In [2], after generating the sparse system for Bivium and Trivium as described as the first approach in the previous section, they use this method for solving the system. The complexity results are summarize in the following table:

Bivium A	Bivium B	Trivium
"about a day"	2^{56} sec	2^{162} sec

As shown by the table, they are not be able to break the full Trivium with this kind of approach. Also, we can see that the variant A of Bivium is very weak.

3.2. Guess value of state bits

In [6], they propose two method to attack Trivium and its variant Bivium B. The first one is a state recovering attack: they try to guess the value of some state bit or the value of the product of some state bits. In suitable case, they reduce the system to a system of linear equations, which can be solved, using for example the Gaussian elimination. The other technique is a distinguishing attack: this techniques allows to collect statistics on keystream and to build a distinguisher. With the didtinguishing attack they can't give an estimation for the complexity of Trivium. They obtain the following results:

	Bivium B	Trivium
Method 1	$c \cdot 2^{36.1}$ $c \approx 2^{14}$	$c \cdot 2^{83.5}$ $c \approx 2^{16}$
Method 2	2^{32}	?

3.3. SatSolver

An algebraic system of equations in $\mathbb{F}_2$ can be converted in logical expression, and it can be solved by a satisfiability solver. In [3] they use this approach, with the satisfiability solver MiniSat [4] for attacking Bivium. Also in this attack the system is generated by adding two new variables at each clock. The results obtained are experimental,because the MiniSat algorithm has unpredictable behaviour and complexity. Moreover, the complexity that they obtained in case of Bivium B is only an estimation.

Bivium A	Bivium B	Trivium
21 sec	2^{52} sec	?

4. Systems generated by Trivium an Bivium

We can generate a system of equations in $\mathbb{F}_2$ for Trivium and Bivium A and B. We consider bits $(s_1, \ldots, s_{288})$ of the registers at the time when the keystream generation is about to start as unknown. We can use two approach for generating equations.
The first one is to generate equations adding three new variables for any bit of keystream for Trivium and two new variables for any bit of keystream for Bivium. The second one is to generate equations without adding any variable, to have a system which has $s_1, \ldots, s_{288}$ as unknown.
When clocking Trivium once, we introduce three new bits to the next state. For the first approach, we use these three new bits as new variables. For the first clock we

obtain the following equations to express these three new unknowns as non-linear combination of bits of the state:

$$\begin{aligned} s_{289} &= s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171} \\ s_{290} &= s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264} \\ s_{291} &= s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69} \end{aligned}$$

Moreover we get one equation from the known keystream bit z_1:

$$z_1 = s_{66} + s_{93} + s_{162} + s_{177} + s_{243} + s_{288}$$

We repeat this procedure, so at each clock we have three new variables and four equations. In case of Bivium, at each clock, we get two new variables and three equations. This kind of approach is used to keep the equations sparse enough. The degree of equations in the system described as above is lesser than or equal to two. If we do not add any new variable we obtain a system which uses only 288 unknowns. At the clock i we get one equation, which expresses the bit z_i of the keystream as a non-linear function of the variables $s_1, \ldots, s_{288}$. Obviously the degree of equations increase. In fact, for example, in case of Trivium we have that for the first 66 clocks the equations are linear, then for $67 \leq i \leq 148$ they have degree two and for $149 \leq i \leq 214$ the degree becomes three and so on. The equations we can obtain with this method are as follows:

$$\begin{aligned}
z_1 &= s_{66} + s_{93} + s_{162} + s_{177} + s_{243} + s_{288} \\
z_2 &= s_{65} + s_{92} + s_{161} + s_{176} + s_{242} + s_{287} \\
&\vdots \\
z_{66} &= s_1 + s_{28} + s_{97} + s_{112} + s_{178} + s_{223} \\
z_{67} &= s_{175} \cdot s_{176} + s_{286} \cdot s_{287} + s_{27} + s_{69} + s_{96} + s_{111} + s_{162} + s_{177} + s_{222} + \\
&\quad s_{243} + s_{264} + s_{288} \\
&\vdots \\
z_{148} &= s_{13} \cdot s_{14} + s_{28} \cdot s_{29} + s_{79} \cdot s_{80} + s_{91} \cdot s_{92} + s_{94} \cdot s_{95} + s_{139} \cdot s_{140} + \\
&\quad s_{160} \cdot s_{161} + s_{205} \cdot s_{206} + s_{232} \cdot s_{233} + s_3 + s_{30} + s_{54} + s_{66} + s_{81} + \\
&\quad s_{93} + s_{96} + s_{108} + s_{126} + s_{141} + s_{147} + s_{159} + s_{162} + s_{171} + s_{183} + \\
&\quad s_{189} + s_{207} + s_{228} + s_{234} + s_{249} \\
z_{149} &= s_{91} \cdot s_{92} \cdot s_{94} + s_{12} \cdot s_{13} + s_{27} \cdot s_{28} + s_{78} \cdot s_{79} + s_{90} \cdot s_{91} + s_{66} \cdot s_{94} + \\
&\quad s_{93} \cdot s_{94} + s_{138} \cdot s_{139} + s_{159} \cdot s_{160} + s_{94} \cdot s_{171} + s_{204} \cdot s_{205} + s_{231} \cdot s_{232} + \\
&\quad s_2 + s_{29} + s_{53} + s_{65} + s_{80} + s_{92} + s_{95} + s_{107} + s_{125} + s_{140} + s_{146} + \\
&\quad s_{158} + s_{161} + s_{170} + s_{182} + s_{188} + s_{206} + s_{227} + s_{233} + s_{248} \\
&\vdots
\end{aligned}$$

5. Attacks with Gröbner basis

Our approach is using Gröbner bases computation to attack Bivium and Trivium. Computations have been done with the algorithm F_4 [5] implemented in Magma.

Polynomials in Gröbner basis that we computed, are linear, and it is simple recover values of unknown.
We use both approaches described in Section 4. Gröbner basis computation for Trivium did not finish in any case, so we do not have any experimental results.
Let I be the system obtained without adding any variable for Bivium. So $I_B \subset \mathbb{F}_2[s_1, \ldots, s_{177}]$. We can obtain 320 equations. The complexity results are:

- Bivium A
  - If we do not guess any variable, the computation finishes in 68732.180 seconds
  - If we guess just 5 "suitable" variables, the computation finishes in 40.819 seconds
- Bivium B
  - If we do not guess any variable, the computation does not finish
  - If we guess just 56 variables, the computation finishes in 483.640 seconds

We will explain later, what "suitable" means.
It is evident that the light difference between the structures of Bivium A and B, generates a big difference in computation time.

Let J_n be the system obtained from Bivium adding two new variables for any clock, where n is the number of known keystream bits. So J_n is a system in $2n + 177$ variables, with $3n$ equations. In this case we obtain better complexity results with respect the previous approach for the variant A, but it seems to be worse for the variant B. Infact we have that:

- Bivium A
  - If we do not guess any variable and $n = 2000$ the computation finishes in 400.810 seconds
  - If we guess just 2 "suitable" variables and $n = 800$, the computation finishes in 17.930 seconds
- Bivium B
  - If we do not guess any variable, the computation does not finish
  - If we guess 56 variables and $n = 2000$, the computation finishes in 1006.259 seconds

5.1. "Suitable" variables

Our idea is that, if we understand the structure of systems I and J_n, we can choose apposite variables to guess. In fact, we try to guess variables for decreasing the degree of equations in I and for having as many linear equation as possible in J_n. We try two different strategies:

1. we guess alternate variables. Infact, in the algorithm for Bivium we have that the equations are not linear for the product $s_{91} \cdot s_{92}$ and for $s_{175} \cdot s_{176}$. This choice seems to work better for the ideal I;

2. we guess consecutive variables between s_{91} and s_{176}, which come out more often than the other variables in term of degree greater than 1. This approach works better for the ideal J_n.

References

[1] H. Raddum, I. Semaev. *New Technique for Solving Sparse Equation Systems*. Internal ECRYPT webpage, `http://www.cosic.esat.kuleuven.be/ecrypt/intern/STVL`, January 2006.

[2] H. Raddum. *Cryptanalytic Results on Trivium*. eSTREAM, ECRYPT Stream Cipher Project, `http://www.ecrypt.eu.org/stream`, 2006

[3] C. McDonald, C. Chernes, J. Pieprzyk. *Attacking Bivium with MiniSat*. `http://eprint.iacr.org/2007/129`, 2007.

[4] MiniSat 2.0. `http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/MiniSat.html`

[5] J. C. Faugére. *A new efficient algorithm for computing Gröbner bases* (F_4), J. Pure Appl. Algebra, 1999, vol. 139, pag. 61–88,

[6] A. Maximov, A. Biryukov. *Two trivial attacks on Trivium*, Cryptology ePrint Archive, Report 2007/021, 2001

[7] eSTREAM: ECRYPT Stream Cipher Project. `http://www.ecrypt.eu.org/stream/`

Ilaria Simonetti
Department of Mathematics
University of Milano
Italy

Ludovic Perret
UPMC Paris Universitas 06
LIP6/ INRIA Paris-Rocquencourt
SALSA
Paris, France

Jean Charles Faugère
UPMC Paris Universitas 06
LIP6/ INRIA Paris-Rocquencourt
SALSA
Paris, France