Two Trivial Attacks on Trivium

Alexander Maximov and Alex Biryukov

Laboratory of Algorithmics, Cryptology and Security
University of Luxembourg
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
movax@mail.ru, alex.cryptan@gmail.com

Abstract. Trivium is a stream cipher designed in 2005 by C. De Cannière and B. Preneel for the European project eSTREAM. It has an internal state of 288 bits and the key of length 80 bits. Although the design has a simple and elegant structure, no attack on it has been found yet.

In this paper a family of Trivium-like designs is studied. We propose a set of techniques for methodological cryptanalysis of these structures in general, including state recovering and linear distinguishing attacks. In particular, we study the original Trivium and present a state recovering attack with time complexity around $c2^{83.5}$, which is 2^{30} faster than the best previous result. Our attack clearly shows that Trivium has a very thin safety margin and that in its current form it can not be used with longer 128-bit keys.

Finally, we identify interesting open problems and propose a new design Trivium/128, which resists all of our attacks proposed in this paper. It also accepts a 128 bit secret key due to the improved security level.

1 Introduction

Additive stream ciphers are an important class of data encryption primitives, in which the process of encryption simulates the one-time-pad. The core of any stream cipher is its *pseudo-random keystream generator* (PRKG). It is initialized with a *secret key* K, and an *initial value* (IV). Afterwards, it produces a long *pseudo-random sequence* called *keystream* $\mathbf{u}$. In the encryption procedure, the ciphertext $\mathbf{c}$ is then obtained by a bitwise xor of the message $\mathbf{m}$ and the keystream $\mathbf{u}$, i.e., $\mathbf{c} = \mathbf{m} \oplus \mathbf{u}$.

Many stream ciphers are currently used in various aspects of our life. To mention some of them, they are: RC4 [Sma03] (is used on the Internet), E_0 [Blu03] (in Bluetooth), A5/1 [BGW99] (in GSM communication), and others. However, it has been shown that these primitives are susceptible to various kinds of weaknesses and attacks [FM00, MS01, LV04, LMV05, BSW00, MJB04]. In 1999 the European project NESSIE was launched [NES99] and among other encryption and signature primitives it attempted to select stream ciphers for its final portfolio. However after a few rounds of evaluation and cryptanalysis, most of the

C. Adams, A. Miri, and M. Wiener (Eds.): SAC 2007, LNCS 4876, pp. 36–55, 2007.

proposals were broken[1]. As a result the board of the project NESSIE could not select any of the stream cipher proposals for its final portfolio.

The recent European project ECRYPT [ECR05] has started in 2004 within the Sixth Framework Programme (FP6). It announced a new call for stream cipher proposals, for its subproject eSTREAM. In the first phase 34 proposals were received, but only a few of them got the status of "focused" algorithms in the second phase. In the *hardware portfolio* only four new designs are in focus, they are: Trivium [CP05], Grain [HJM05], Mickey [BD05], and Phelix [WSLM05].

In this paper we analyze one of these designs – Trivium. The stream cipher Trivium was proposed in 2005 for the project eSTREAM by C. De Canniére and B. Preneel [CP05]. It has an internal state of 288 bits and the key of 80 bits. Though the cipher was designed for hardware implementation it is also very fast in software, which makes it one of the most attractive candidates of the competition. The structure of the cipher is elegant and simple, and it follows clearly described design principles. After the design was announced many cryptographers tried to analyze it. However, only two results on Trivium are known so far.

The first known result is actually given on the eSTREAM discussion forum [eDF05] where the complexity to recover the internal state from given keystream is argued to be 2^{135}. The second result is a paper from H. Raddum [Rad06], where a new algorithm for solving nonlinear systems of equations is proposed and applied on Trivium. The attack complexity found was 2^{164}. Two reduced versions of this design, Bivium -A and -B, were proposed in that paper as well. The first reduced version was broken "in about one day", whereas the second version required time of around 2^{56} *seconds*.

In this paper we consider the design of Trivium in general, and as examples we consider two instances: the original design of Trivium and a reduced version Bivium, the one given in [Rad06] under the name Bivium-B. We propose a set of techniques to analyse this class of stream ciphers, and show how its internal state can be recovered given the keystream. The complexity of this attack determines the upper bound for the security level of the cipher. Its complexities for Trivium and Bivium are found to be $c \cdot 2^{83.5}$ and $c \cdot 2^{36.1}$, respectively, where c is the complexity of solving a sparse system of linear equations (192 for Trivium and 118 for Bivium). It means that, for example, the secret key cannot be increased to 128 bits in a straightforward way unless the design in general is changed. This time complexity is much better than in [eDF05] and [Rad06], and is the best known result on Trivium so far.

In the second attack linear statistical methods are applied. We show how a distinguisher can be built, and propose a linear distinguishing attack on Bivium with less than 2^{32} operations in total. This attack was implemented and in practice works even slightly better than expected.

We also show how cryptanalysis of Trivium can be related to another general problem. For example, if one would know how to solve a highly structured system

[1] There was a discussion at NESSIE on whether a distinguishing attack of very high complexity qualifies as a break of a cipher.

of 576 quadratic equations on 576 unknowns efficiently, he would be able find the full secret state of the cipher. On the other hand, putting a designer hat on, we propose several simple ideas which could help strengthen a TRIVIUM-like design. Following those we propose a tweaked design TRIVIUM/128, which is a slight modification of TRIVIUM, but is believed to have a larger security margin, and thus can be used with a larger 128 bit secret key.

This paper is organized as follows. In Section 2 we define the structures of TRIVIUM and BIVIUM. Afterwards, in Section 3, we give methods for a *state recovering attack*, and propose a set of attack scenarios for both TRIVIUM and BIVIUM. In Section 4 we propose a general attack scenario on the whole family of TRIVIUM-like stream ciphers. A linear distinguishing attack is given in Section 5. We identify a few interesting open problems, and propose an improved design TRIVIUM/128 in Section 6 (and in Appendix A). The paper ends with the summary of our results and conclusions.

1.1 Notation

In this paper we accept the following notation. A single bit will commonly be denoted by $x_i^{(t)}$, where i is an index of a variable, and t is the time instance. Bold symbols $\mathbf{u}$ represent a stream or a vector of bit-oriented data $u_1, u_2, \ldots$. Let us also define *triple-clock* of a cipher as just three consecutive clocks of it.

2 Bivium and Trivium

In Figure 1 two classes of stream ciphers are shown, namely, BIVIUM and TRIVIUM.

The number of basic components is two or three, respectively. Each basic component (a register) consist of *three blocks, each of size divisible by 3*. An instance of this class is a *specification vector* with the blocks' sizes specified, i.e.,

$$\begin{aligned} \text{BIVIUM} &\Rightarrow \text{BI}(A_1, A_2, A_3; B_1, B_2, B_3), \\ \text{TRIVIUM} &\Rightarrow \text{TRI}(A_1, A_2, A_3; B_1, B_2, B_3; C_1, C_2, C_3). \end{aligned} \tag{1}$$

Notation on the registers is summarized in Table 1.

The exact algorithm of TRIVIUM is given in Table 2.

At any time t, the keystream bits of BIVIUM and TRIVIUM are derived as $u_t = x_t + y_t$, and $v_t = x_t + y_t + z_t$, respectively. In this paper two examples from

Table 1. The structure of the internal state's registers

Reg	total length	cells denoted	the AND gate	In:Out	Res
R_A	$A = A_1 + A_2 + A_3$	$a_0^{(t)}, \ldots, a_{A-1}^{(t)}$	$a_{A-3}^{(t)} \cdot a_{A-2}^{(t)}$	$p_t : q_t$	x_t
R_B	$B = B_1 + B_2 + B_3$	$b_0^{(t)}, \ldots, b_{B-1}^{(t)}$	$b_{B-3}^{(t)} \cdot b_{B-2}^{(t)}$	$q_t : p_t / r_t$	y_t
R_C	$C = C_1 + C_2 + C_3$	$c_0^{(t)}, \ldots, c_{C-1}^{(t)}$	$c_{C-3}^{(t)} \cdot c_{C-2}^{(t)}$	$r_t : p_t$	z_t

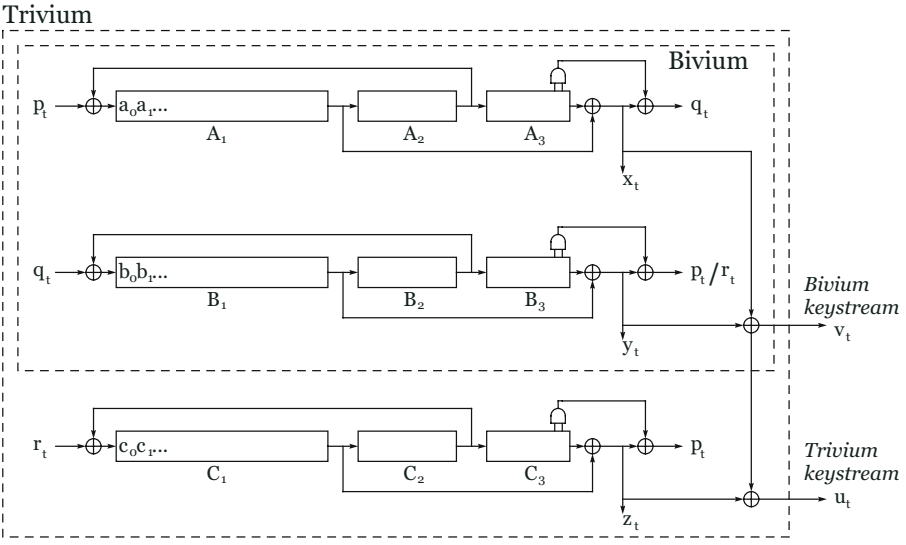

Fig. 1. Bivium and Trivium classes of stream ciphers

Table 2. TRIVIUM stream cipher

Initialisation Procedure(Key, IV)
Repeat until enough of keystream is produced

$$a_0^{(t+1)} = a_{A_1+A_2-1}^{(t)} \oplus c_{C-1}^{(t)} \oplus c_{C_1-1}^{(t)} \oplus c_{C-3}^{(t)} \cdot c_{C-2}^{(t)}$$
$$b_0^{(t+1)} = b_{B_1+B_2-1}^{(t)} \oplus a_{A-1}^{(t)} \oplus a_{A_1-1}^{(t)} \oplus a_{A-3}^{(t)} \cdot a_{A-2}^{(t)}$$
$$c_0^{(t+1)} = c_{C_1+C_2-1}^{(t)} \oplus b_{B-1}^{(t)} \oplus b_{B_1-1}^{(t)} \oplus b_{B-3}^{(t)} \cdot b_{B-2}^{(t)}$$
$$a_i^{(t+1)} = a_{i-1}^{(t)}, \quad \forall i \in [1 : A-1]$$
$$b_j^{(t+1)} = b_{j-1}^{(t)}, \quad \forall j \in [1 : B-1]$$
$$c_k^{(t+1)} = c_{k-1}^{(t)}, \quad \forall k \in [1 : C-1]$$
$$u_t = a_{A-1}^{(t)} \oplus a_{A_1-1}^{(t)} \oplus b_{B-1}^{(t)} \oplus b_{B_1-1}^{(t)} \oplus c_{C-1}^{(t)} \oplus c_{C_1-1}^{(t)}$$

Table 3. Two instances' specifications, TRIVIUM and BIVIUM

Description	Specification	$A : B : C$	Size, θ
TRIVIUM [CP05]	TRI$(66, 3, 24; 69, 9, 6; 66, 21, 24)$	$93 : 84 : 111$	288
BIVIUM [Rad06]	BI$(66, 3, 24; 69, 9, 6)$	$93 : 84 : -$	177

this class of stream ciphers are considered in detail, the specification of which is given in Table 3. These correspond to TRIVIUM and BIVIUM as described in [CP05, Rad06].

For simplicity in further derivations let us introduce three subsets:

$$\begin{aligned} \mathcal{T}_0^{(t)} &= \{a_{3i+0}^{(t)}\} \cup \{b_{3j+0}^{(t)}\} \cup \{c_{3k+0}^{(t)}\} \\ \mathcal{T}_1^{(t)} &= \{a_{3i+1}^{(t)}\} \cup \{b_{3j+1}^{(t)}\} \cup \{c_{3k+1}^{(t)}\} \\ \mathcal{T}_2^{(t)} &= \{a_{3i+2}^{(t)}\} \cup \{b_{3j+2}^{(t)}\} \cup \{c_{3k+2}^{(t)}\} \end{aligned} \quad \text{where} \quad \begin{aligned} i &= 0, 1, \ldots, A/3-1, \\ j &= 0, 1, \ldots, B/3-1, \\ k &= 0, 1, \ldots, C/3-1. \end{aligned} \tag{2}$$

3 First Analysis: State Recovering

In this attack, given a keystream $\mathbf{u}$ of some length n an attacker wants to recover the internal state of the cipher. Since the cipher has invertible state-update function this also leads to a *key recovery attack*. A classical time-memory trade-off technique based on the birthday paradox gives the upper bound for such an attack of $O(2^{\theta/2})$ known keystream, and memory, where θ is the size of the internal state. The importance of the *state recovering analysis* is that it gives the upper bound for the length of the secret key K. When the design of Trivium appeared, several researchers raised the question: *Whether the secret key can be increased from 80 bits till, for example, 128 bits, thus, improving the security level?* In this section we will give the precise answer.

3.1 Guessing $\mathcal{T}_0^{(t)}$ at Some Time t

One of the main observations is that all blocks of the cipher are divisible by 3. Moreover, the transition of the internal state at time t to time $t+1$ is a linear transformation of the subset $\mathcal{T}_{t \bmod 3}^{(t)}$, plus a minor one bit disturbance from the adjacent two subsets. Therefore, the attack scenario can consist of the following phases.

Phase I: Guess the state $\mathcal{T}_0^{(t)}$ at some time t,
Phase II: Having the state $\mathcal{T}_0^{(t)}$ guessed correctly, recover the rest of the bits.

Since the second phase depends on the first phase, the total complexity of the attack $C_{\mathtt{tot}}$ is

$$C_{\mathtt{tot}} = C_{\text{Phase I}} \cdot C_{\text{Phase II}}. \tag{3}$$

Phase I could be done by an exhaustive search of the true state $\mathcal{T}_0^{(t)}$ at some time t. The time complexity of this search is $O(2^{\theta/3})$, and the keystream length required is $O(1)$. However, this complexity can be reduced if we note that the first $d = \min\{A_1, B_1, C_1\}/3$ forward triple-clocks we receive d linear equations on the bits of $\mathcal{T}_0^{(t)}$. By this way the total time complexity is reduced down to

$$O(2^{(\theta - \min\{A_1, B_1, C_1\})/3}).$$

For Trivium and Bivium these complexities are 2^{74} and 2^{37}, respectively. In the following subsections we discuss other ideas on what can be done to make the total complexity of an attack smaller.

3.2 Guessing Outcomes for Specific AND Gates

To receive more linear equations for the two phases, one can consider a set of specific AND gates:

$$\begin{aligned} &a_{A-3}^{(t+3i)} \cdot a_{A-2}^{(t+3i)}, \quad i = 0, 1, \ldots, g_a - 1, \\ &b_{B-3}^{(t+3j)} \cdot b_{B-2}^{(t+3j)}, \quad j = 0, 1, \ldots, g_b - 1, \\ &c_{C-3}^{(t+3k)} \cdot c_{C-2}^{(t+3k)}, \quad k = 0, 1, \ldots, g_c - 1, \end{aligned} \tag{4}$$

where g_a, g_b, g_c are some chosen parameters. Whenever the outcomes of these gates are guessed, the number of linear equations that one can derive for the first phase is

$$d' = \min\{g_a + \frac{B_1}{3}, g_b + \frac{C_1}{3}, g_c + \frac{A_1}{3}\}.$$

The most probable guess would be that all these gates produce zeros, since $\Pr\{x\&y = 0\} = 0.75$, and we simply search in the keystream for the place where this is satisfied. The expected length of the keystream in this case is around $(0.75)^{-(g_a+g_b+g_c)}$. However, if we allow some of the gates to produce ones, the length of the keystream can be reduced significantly.

Let G gates out of $g_a + g_b + g_c$ AND gates produce zeros, and the remaining H gates produce ones. The total probability of this event is

$$p_g = 0.75^G 0.25^H.$$

Note that we can allocate H ones among $G+H$ positions in $\binom{G+H}{H}$ ways. Therefore, the keystream is needed to be of length approximately $O\left(1/\left[p_g \cdot \binom{G+H}{H}\right]\right)$.

3.3 Guessing Sums of Specific AND Gates

The right guess of the specific AND gates from the previous subsection allowed us to increase the number of linear equations for the first phase till d'. However, the remaining bits of $\mathcal{T}_0^{(t)}$ should be guessed with probability 1/2. The total probability of the remaining guess could, however, be reduced if the available keystream can be increased. Below we show another trade-off between the keystream and the complexity of the remaining guessing part.

After d' triple-clocks, we start receiving nonlinear equations, where the linear part consists of the bits from $\mathcal{T}_0^{(t)}$, and the nonlinear part is the sum of w AND gates, for some small w. Since the outcome of each of them is biased, then their sum is biased as well. Let p_w be the probability that the sum of w gates is zero, which is derived as

$$p_w = \sum_{i=0}^{\lfloor w/2 \rfloor} \binom{w}{2i} 0.75^{w-2i} 0.25^{2i} \quad (> 0.5), \tag{5}$$

or, via its recursive formula as $p_{w+1} = 0.75p_w + 0.25(1 - p_w)$, with $p_0 = 1$. Let l_w be the number of nonlinear equations with the sum of w AND gates. The time

complexity to recover l_w bits is $p_w^{l_w}$, instead of 0.5^{l_w}, but, however, it requires to increase the length of the keystream by the ratio $p_w^{-l_w}$. The total probability of such an event is

$$q = \sum_{w=1}^{\infty} p_w^{l_w}.$$

This approach is reasonable to use for small ws, say for $w \in \{1, 2, 3, 4\}$, since for large ws the probability p_w is close to 0.5 and, therefore, it does not give a big gain versus a truly random guess, but rather increases the length of the keystream rapidly.

3.4 Collecting System of Equations for Remaining Unknowns

Assume that the state of $\mathcal{T}_0^{(t)}$ and the outcomes of specific $G + H$ AND gates are guessed and derived correctly. To recover the remaining 2/3 of the state we need to collect a number of equations on $\mathcal{T}_1^{(t)}$ and $\mathcal{T}_2^{(t)}$, enough to derive the exact solution.

At any time t, if the values $a_{A-3}^{(t)}, b_{B-3}^{(t)}, c_{C-3}^{(t)}$ are known, then two consecutive clocks of the cipher are linear. Because of our specific guess, we know that d' triple-clocks the system is linear. In one triple-clock two linear equations on the remaining unknowns of the internal state are received. The first nonlinearity will not affect on the degree of receiving equations immediately, but rather with some delay. The first nonlinear equations will be of degree 2, and then of degree 3, and so on. Also note that each of H guesses also give us two equations of degree 1 of the form $x_i = 1$ and $x_{i+1} = 1$, and each of the G guesses give us another equation of degree 2 of the form $x_i x_j = 0$. The structure of this cipher is such that backward clocks increase the degree of equations rapidly[2]. Therefore, only a few equations of low degree can be collected by backward clocking.

Let the number of equations of degrees 1 and 2 that can be collected be n_1 and n_2, respectively. Whenever all the parameters are fixed, a particular scenario can be described.

3.5 Attack Scenarios for Trivium and Bivium

In this subsection we accumulate techniques given in the previous subsections, and propose a set of attack scenarios for Trivium and Bivium in Table 4[3]. Moreover, a brief algorithm of the scenario T1 is presented in Table 5.

In all scenarios above the constant c is the time required for the second phase, where the remaining bits are recovered, and it is different for different scenarios.

T0 and B0 are trivial scenarios for Trivium and Bivium, where no outcomes of any AND gates are guessed. However, the number of linear equations

[2] Trivium is designed to maximize parallelism in forward direction. This allows hardware designers to choose trade-off between speed and chip-size.

[3] The keystream length given in the table is an average number of positions at the keystream from where we need to study a window of around $A + B + C$ consecutive bits.

Table 4. Attack scenarios

Scenario T0		Descr. = Tri			$l_1{:}l_2{:}l_3{:}l_4$ = 0:0:0:0			Ph.II unknowns=192	
$g_a{:}g_b{:}g_c$	$G{:}H$	r	d'	q	p_g	n_1	n_2	time	keystream
0:0:0	0:0	1	22	1	1	100	61	$c \cdot 2^{74.0}$	$O(1)$
Scenario T1		Descr. = Tri			$l_1{:}l_2{:}l_3{:}l_4$ = 5:5:4:1			Ph.II unknowns=192	
$g_a{:}g_b{:}g_c$	$G{:}H$	r	d'	q	p_g	n_1	n_2	time	keystream
46:37:42	125:0	1	59	$2^{-9.7}$	$2^{-51.9}$	192	178	$c \cdot 2^{83.5}$	$2^{61.5}$
Scenario T2									
42:33:38	113:4	$2^{22.6}$	55	$2^{-9.7}$	$2^{-53.2}$	192	162	$c \cdot 2^{88.9}$	$2^{40.3}$
Scenario T3		Descr. = Tri			$l_1{:}l_2{:}l_3{:}l_4$ = 0:0:5:4			Ph.II unknowns=192	
$g_a{:}g_b{:}g_c$	$G{:}H$	r	d'	q	p_g	n_1	n_2	time	keystream
29:30:30	89:0	1	52	$2^{-7.8}$	$2^{-36.9}$	158	152	$c \cdot 2^{79.7}$	$2^{44.7}$
Scenario B0		Descr. = Bi			$l_1{:}l_2{:}l_3{:}l_4$ = 0:0:0:0			Ph.II unknowns=118	
$g_a{:}g_b{:}g_c$	$G{:}H$	r	d'	q	p_g	n_1	n_2	time	keystream
0:0:—	0:0	1	22	1	1	100	61	$c \cdot 2^{37.0}$	$O(1)$
Scenario B1									
9:5:—	14:0	1	27	1	$2^{-5.8}$	118	67	$c \cdot 2^{37.8}$	$2^{5.8}$

is not enough to recover the remaining bits using simple Gaussian elimination. Therefore, equations of a higher degree need to be collected and used. These scenarios have the least possible time and keystream complexities, and are the lower bounds.

In T1 and B1 we show optimal, on our view, choice of parameters such that the second phase has enough linear equations and the time complexity is minimal. However, along with linear equations we also have many equations of degree 2, which we are not using at all. Note that the attack complexities presented here are much lower than those given in [Rad06].

In T2 we show how the trade-off between the length of the keystream and time works. For a small increase of time we can reduce keystream significantly.

In T3 we receive a system of equations of degree ≤ 2 on 192 variables. This system is quite over-defined (more than 50%), and it might be possible to have an efficient algorithm for solving such a system.

However, the results given in these scenarios can be improved significantly if a *pre-* or/and a *post-* statistical tests can be applied efficiently. The goal of such a test is to reduce the constant c. For these approaches see Appendices A and B.

Another possibility to reduce the constant c can be achieved via efficient solving a system of sparse linear equations (in cases of T1, T2, B1), or through the use of high degree equations (in cases of T3, B0). Finding such an algorithm is a hard problem, and we leave it as an open problem, identified in a more general form in Section 6.

3.6 Our Results vs. Exhaustive Search

We have shown that Bivium can be broken in time around $c \cdot 2^{37}$, which determines a really low bound for the security level. This example was taken into

Table 5. Attack scenario T1 for TRIVIUM in brief

Given: $\mathbf{u} = u_1, u_2$ – the keystream of TRIVIUM of length $2^{61.5}$
Attack Scenario T1*:*

1. For every $t = 0, 1, 2, \ldots, \lceil 2^{61.5} \rceil$ assume that $a_{90}^{(t+3i)} a_{91}^{(t+3i)} = 0, b_{81}^{(t+3j)} b_{82}^{(t+3j)} = 0, c_{108}^{(t+3k)} c_{109}^{(t+3k)} = 0$, for $i = [0 : 45], j = [0 : 36], k = [0 : 41]$.
2. Collect 59 linear equations on $\mathcal{T}_0$ with probability 1, and 15 more linear equations with the total probability $2^{-9.7}$, see Subsection 3.3.
3. For every guess of the remaining 22 bits from $\mathcal{T}_0$, derive the state of $\mathcal{T}_0$ using the linear equations collected in step 2.
4. Collect 192 linear equations on $\mathcal{T}_1$ and $\mathcal{T}_2$, clocking the cipher forward, under the assumption that the guess above was correct.
5. Recover the state of $\mathcal{T}_1$ and $\mathcal{T}_2$ by any linear technique (e.g., Gaussian elimination) in fixed time, and verify the solution in time $O(1)$.
6. Repeat the loops in steps 1 and 3 until the right internal state is found.

account in order to make a comparison of the techniques versus the ones used in the paper [Rad06], where the best attack on this design has been found to be of the complexity around $c \cdot 2^{56}$ *seconds*.

Although the security level of TRIVIUM is 2^{80}, we believe that an exhaustive search will require much more time, $\gamma 2^{80}$, where γ is the initialization time of the cipher that includes 1152 clocks to be done before the first keystream bits are produced. Because of different implementation issues can be applied, including parallelism, an average time required for one clock of the cipher can vary. However, we believe that a conservative value for the coefficient γ is around 2^{10}, and an exhaustive search would require around 2^{10+80} operations. This means that such scenarios, such as T1, T3, are competitive in terms of the time complexity, and at least are very close to the exhaustive search, if not faster.

Obviously, in this particular design the security level cannot be improved by simply increasing the size of the key – our attack will definitely be faster than an exhaustive search in that case. Therefore, in order to increase the security level the design of TRIVIUM should be changed, for example, the size of the state could be increased. This would result in a longer initialization time and a larger hardware footprint.

4 General Attack Scenario

Let us investigate a general structure of TRIVIUM-like stream ciphers with the following properties[4].

– It has k nonlinear shift registers $S = (S_1, S_2, \ldots, S_k)$, with the corresponding lengths $L = (L_1, L_2, \ldots, L_k)$. The bits of each register S_i are $S_i[1], \ldots, S_i[L_i]$;

[4] In this section a slightly changed manner of the indexation for the vectors is used, starting to count the indices from 1, instead of 0.

– Each register is divided into blocks, all divisible by d;
– There are k AND gates, and they are placed as $\texttt{AND}(S_i[L_i-2], S_i[L_i-1])$ like in TRIVIUM.

Let us denote this structure as $\text{TRIVGEN}(k, d, L)$. For simplicity, Let the vector of the lengths L be sorted such that $L_i \leq L_{i+1}, \forall i$. In this section we study a scenario of a *general state recovering attack* on the whole class of the TRIVIUM-like family of stream ciphers. The total size of the internal state S is

$$l = \sum_{i=1}^{k} L_i. \tag{6}$$

The d subsets of the total state S are defined as

$$\mathcal{T}_i = \left\{ \begin{cases} \forall j = 1, \ldots, k \\ \forall t = 0, \ldots, \frac{L_j}{d} - 1 \end{cases} : S_j[td+i] \right\}, \quad \forall i = 1, \ldots, d. \tag{7}$$

4.1 Phase I: Deriving $\mathcal{T}_1$ and $\mathcal{T}_2$

Let us first explain how $|\mathcal{T}_1|$ bits of the first subset $\mathcal{T}_1$ can be derived. We simply assume (or guess) that during consecutive $d \cdot |\mathcal{T}_1|$ clocks of TRIVGEN enough linear equations on $\mathcal{T}_1$ are collected. One observes every d's output bit at the keystream and writes up equations one by one, where every new AND term is approximated by zero.

Assume Δ is the number of linear equations that can be received from the keystream without any approximations. Let for the remaining $|\mathcal{T}_1| - \Delta$ equations the number of G AND terms have to be approximated. The values of Δ and G are easy to calculate once an exact instance of the design is given. The number of such guesses G is upper bounded as

$$G \leq k \left(\frac{l}{d} - \Delta \right). \tag{8}$$

The same procedure can be done for $\mathcal{T}_2$. Note that these two parts share the same values of Δ and G.

Instead of approximating $2G$ AND gates with probability 0.75 each, when deriving linear equations on $\mathcal{T}_1$ and $\mathcal{T}_2$, one can use the fact that these gates are not independent. At some time i two gates, one for the equation on $\mathcal{T}_1$ and one on $\mathcal{T}_2$, share one variable, i.e., if the first gate is $\texttt{AND}(a, b)$, then the second is $\texttt{AND}(b, c)$. The probability that both gates produce zeros is $5/8$. Therefore, the total probability of the required guess is

$$p_I = (5/8)^G. \tag{9}$$

4.2 Phase II: Calculating $\mathcal{T}_3 \ldots \mathcal{T}_d$

In the first phase two subsets $\mathcal{T}_1$ and $\mathcal{T}_2$ are received. Additionally, by guessing every AND term we also guaranteed that during the first l clocks only linear

transformations over the two subsets are applied. It means that an outcome of every AND gate that is connected to $\mathcal{T}_3$ is known. Thus, required number of linear equations on $\mathcal{T}_3$ can be collected, and then Gaussian elimination is applied.

After $\mathcal{T}_3$ is determined, we can start with a similar procedure to derive $\mathcal{T}_4$, and so on. When the final subset $\mathcal{T}_d$ is derived, one can use the guesses from the first phase to check if they are in a conflict with the recovered state or not.

The total complexity of this part is

$$c_{II} \approx O(d \cdot (l/d)^{2.808}), \tag{10}$$

if the *Strassen's algorithm* for computing solution of a linear system is used. In this complexity we also included time for similar computation of the first phase. This attack scenario requires around p_I^{-1} of the keystream, and c_{II}/p_I of time.

4.3 Example: Trivium-6

Let us consider the following construction

$$\text{Trivium-6} \Rightarrow \text{Tri}(66, 6, 24;\ 72, 12, 6;\ 66, 24, 24). \tag{11}$$

This is a slightly modified Trivium stream cipher with the internal state of size 300 bits, and all building blocks are divisible by 6 (still divisible by 3, but also by 2). The design of Trivium-6 belongs to the class $\text{TrivGen}(3, 6, (90, 96, 114))$.

I.e., we have $k = 3$, $d = 6$, $l = 300$, and $\mathcal{T}_1, \ldots, \mathcal{T}_6$ are defined as in (7), each of size $|\mathcal{T}_i| = 50$ $(= l/d)$. One can easily check that

$$\begin{aligned} \Delta &= \min\{66, 72, 66\}/6 + 1 = 12, \\ G &= \left[(|\mathcal{T}_1| - \Delta) - (\frac{66}{6} - 11)\right] + \left[(|\mathcal{T}_1| - \Delta) - (\frac{72}{6} - 11)\right] \\ &\quad + \left[(|\mathcal{T}_1| - \Delta) - (\frac{66}{6} - 11)\right] = 38 + 37 + 38 = 113, \end{aligned} \tag{12}$$

where 11 equations of Δ are received from $11 \cdot 6$ forward clocking, and one from backward clockings. Thus,

$$p_I \approx 2^{-76.6}, \quad c_{II} \approx 2^{18.4}. \tag{13}$$

It means that the total time complexity is 2^{95} for this example, which is smaller than for Trivium, although the internal state is larger. Note also that the keystream complexity can significantly be reduces in a similar manner as in Section 3.2 for a small penalty in time.

5 Second Analysis: Statistical Tests

Linear cryptanalysis is one of the most powerful tools for analysis of stream ciphers. In this section we find a way of sampling from the keystream such that their distribution is biased. By this mean we build a linear distinguisher for the cipher.

5.1 Standard Approximation Technique

Let the variables of $\mathcal{T}_0$ be denoted as $\{w_0, w_1, \ldots, w_{95}\}$. Then, *assuming that all* AND *terms are zeros*, we receive a system of linear equations of rank 93 (instead of 96). It means that we can sample from the stream as follows

$$\sum_{i \in \mathcal{I}_k} w_i = N_k, \quad \forall k \in \{93, 94, 95, 96\}, \tag{14}$$

where

$$\begin{aligned}
\mathcal{I}_{93} &= \{0, 1, 4, 6, 8, 9, 12, 13, 14, 17, 19, 20, 23, 25, 27, 30, 31, 34, 35, 38, 39, 41, 43, 44, \\
&\quad 67, 68, 70, 72, 73, 76, 77, 80, 81, 84, 85, 88, 89, 92, 93\}; \\
\mathcal{I}_{94} &= \{0, 2, 4, 5, 6, 7, 8, 10, 12, 15, 17, 18, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 34, 36, \\
&\quad 38, 40, 41, 42, 43, 45, 67, 69, 70, 71, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94\}; \\
\mathcal{I}_{95} &= \{0, 3, 4, 5, 7, 11, 12, 14, 16, 17, 18, 22, 23, 24, 26, 28, 29, 30, 33, 34, 37, 38, 42, \\
&\quad 46, 67, 71, 75, 76, 79, 80, 83, 84, 87, 88, 91, 92, 95\}; \\
\mathcal{I}_{96} &= \{0, 5, 9, 14, 15, 18, 20, 24, 29, 41, 44, 47, 67, 70, 73, 96\}.
\end{aligned} \tag{15}$$

The noise variable N_k is a sum of a set of random AND gates. Therefore, the bias and the complexity of a distinguisher can be summarized in Table 6.

Table 6. Linear distinguishers for TRIVIUM and its attack complexities

k	# of AND gates in N_k	bias ϵ	attack complexity
93	108	2^{-108}	2^{216}
94	126	2^{-126}	2^{252}
95	112	2^{-112}	2^{224}
96	72	2^{-72}	2^{144}

Table 7. Linear distinguishers for BIVIUM and their attack complexities

k	# ANDs	time	$\mathcal{I}_k$
57	49	2^{98}	$\{0, 2, 4, 5, 6, 7, 8, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 33, 34,$ $35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57\}$
58	49	2^{98}	$\{1, 3, 5, 6, 7, 8, 9, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 34, 35,$ $36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58\}$
59	16	2^{32}	$\{0, 5, 9, 10, 14, 33, 36, 59\}$

Obviously, we could also mix these four equations to receive other 8 linear combinations that are different in principal from the found four. However, we could not achieve complexity lower than 2^{144}.

For BIVIUM, the rank appeared to be 57 (instead of 59), and similar resulting Table 7 is as follows.

Table 8. A linear distinguishing attack on BIVIUM in detail

Given: $\mathbf{v} = v_1, v_2$ – the keystream of BIVIUM of length 2^{32}
Init: $P[2] = \mathbf{0}$ – a binary distribution, not normalized
A linear distinguishing attack on BIVIUM*:*
1. For every $t = 1, 2, \ldots, 2^{32}$ calculate

$$s = v_t + v_{t+15} + v_{t+27} + v_{t+30} + v_{t+42} + v_{t+99} + v_{t+108} + v_{t+177},$$

and attune the distribution as $P[s] + +$.
2. After the loop is finished, calculate the distance

$$\xi = P[0]/2^{32} - 0.5.$$

3. Make the final decision

$$\delta(\xi) = \begin{cases} \mathbf{v} \text{ is from BIVIUM}, & \text{if } \xi > 2^{-16}/2, \\ \mathbf{v} \text{ is Random}, & \text{if } \xi \leq 2^{-16}/2. \end{cases}$$

I.e., BIVIUM can be distinguished from random in time complexity 2^{32}, which is much faster than all previously known attacks on it. Since the complexity of the attack is feasible, we could run the simulation of the attack on BIVIUM, which confirmed the found theoretical bias. This attack is shown in brief in Table 8.

5.2 Another Way of Approximation

In the previous section all AND terms were approximated as zero. However, another sort of approximation is possible, such as

$$\texttt{AND}(x, y) = \tau_x x + \tau_y y + n,$$

where τ_x, τ_y are chosen coefficients, and n is the noise variable with the bias $\epsilon = 2^{-1}$. Whenever approximations for every AND gate are appropriately chosen, there must exist a biased linear equation on a shorter window of the keystream than that in the previous subsection. Our goal is to reduce the number of noise variables in the final expression for sampling. Unfortunately, the search for appropriate coefficients, which give us a strongly biased expression for sampling, is a hard task. Moreover, the probability that we can find an expression with the number of gates less than 72 is low. In our simulations we could find several biased equations on a shorter window, but the number of approximations were larger than 72. This issue is an interesting open problem.

5.3 Multidimensional Approximation

In Subsection 5.1 we gave a set of linear relations for a biased sampling from the keystream. The best equations for TRIVIUM and BIVIUM require 72 and 16 approximations of AND gates, respectively. However, these samples are not

independent, and some of the noises appear in several samples at different time instances. Therefore, the attack complexity can be improved by considering several samples jointly. I.e., we suggest to test a multidimensional approximation where one sample comes from a joint distribution.

Unfortunately, this did not give us a significant improvement. We considered three samples jointly, and the bias of that noise was $2^{-15.4}$, which is larger than 2^{-16}, but does not differ significantly.

6 Open Problems

Below we would like to identify several interesting open problems that we found while working on Trivium.

OP-1 Let the complete internal state of Trivium be 288 unknowns. When clocking forward, we receive several equations on these unknowns. Every time when a new `AND` gate appears, we will make a substitution, introducing a new variable into the system. After exactly $k = 288$ clocking, we will receive k linear equations on $3k$ unknowns, and also $2k$ equations of degree 2 (substitutions). All terms of degree 2 will look like $a_0a_1, a_1a_2, a_2a_3, \ldots$, and the same for b's and c's. After partial Gaussian elimination we can remain with $2k$ nonlinear equations on $2k$ unknowns[5].

Let X be a variable, an integer number of length $2k$ bits (=576), representing the solution. Then, the problem of breaking Trivium can, after a slight modification, be interpreted as solving the following equation in $\mathbb{Z}_{2^{2k}}$.

$$X\&(X \ll 1) \oplus M \cdot X = V, \tag{16}$$

where M is a known and fixed Boolean matrix, and V is a known vector, constructed from the keystream. Our task is to find at lest one solution of the equation (guaranteed to exist).

OP-2 The set $\mathcal{T}_0$ is a set of 96 unknowns. We know that each guess of $\mathcal{T}_0$ allows us to construct a system of linear equations on the remaining sets $\mathcal{T}_1$ and $\mathcal{T}_2$. However, we believe that after a partial Gaussian elimination that matrix will look similar to

$$\begin{pmatrix} \mathcal{T}_1 \\ \mathcal{T}_2 \end{pmatrix} \cdot \begin{pmatrix} I & W_1 \\ W_2 & I \end{pmatrix} = V, \tag{17}$$

where W_1 and W_2 are sub-matrices dependent on the guess of $\mathcal{T}_0$, and V is a known vector. Since we made a set of guesses that particular $\mathtt{AND}(\mathcal{T}_1, \mathcal{T}_2)$ gates are zeros, we would like then to "extract" somehow only one pair of bits from this system. Afterwards, we can make a test whether their

[5] The idea of writing up equations with specific substitutions was first proposed by Steve Babbage at SASC-06.

product is zero or not, and then in a case of a wrong result, skip the calculations of the remaining bits.

If it would be possible, then this technique would allow to reduce the constant c in the time complexity of the attack significantly.

OP-3 Finally an interesting open problem is how to strengthen Trivium, while keeping its elegance, simplicity and degree of parallelism. We propose one possible solution to this problem in Appendix A.

7 Results and Conclusions

In this paper we have studied various methods for analysis of Trivium-like stream ciphers. Below we give a comparison Table 9 of the known attacks on two instances, original Trivium and a reduced version called Bivium.

Table 9. Resulting comparison of various attacks

Case	State	Complexity	Exhaustive search	State Recovering Attack		Distinguishing Attack	
				previous	new attack	previous	new attack
Trivium	288 bits	time	$\gamma 2^{80}$ $\gamma \approx 2^{10}$	$\delta \cdot 2^{135}$ [eDF05] 2^{164} [Rad06]	$c \cdot 2^{83.5}$ $c \approx 2^{16}$	2^{144} [CP05]	—
		keystream	$O(1)$	$O(1)$	$2^{61.5}$	2^{144}	—
Bivium	177 bits	time	$\gamma 2^{80}$	2^{56} sec. [Rad06]	$c \cdot 2^{36.1}$ $c \approx 2^{14}$	—	2^{32} verified
		keystream	$O(1)$	$O(1)$	$2^{11.7}$	—	2^{32}
Triv-6	300 bits	time	—	—	$c \cdot 2^{76.6}$ $c \approx 2^{18.4}$	—	—
		keystream	—	—	$2^{76.6}$	—	—

A brief summary for the algorithm of the state recovering attack on Trivium is given in Table 5, and a distinguishing attack on Bivium is presented in Table 8.

With the key of 80 bits Trivium seems to be secure. However, contrary to what one could expect from its almost 300 bit state, there is no security margin. This also means that one cannot use 128 bit keys and IVs with the current design. For this purpose, either the internal state has to be increased or some other redesign should take place. Moreover, we have clearly shown on the example of Trivium-6 that one has to be very carefully when introducing the property of d-divisibility of the construction blocks' lengths.

In this paper we have proposed a modified design Trivium/128, which we believe is more secure than the original Trivium. In hardware, its speed of encryption is the same as in Trivium, but the footprint is slightly larger due to the 3 additional `AND` terms. For the same reason, in software it is also slightly slower. However, its security level seems to be much better.

References

[BD05] Babbage, S., Dodd, M.: Mickey-128 (2005), http://www.ecrypt.eu.org/stream/ciphers/mickey128/mickey128.pdf

[BGW99] Briceno, M., Goldberg, I., Wagner, D.: A pedagogical implementation of A5/1 (1999) (accessed August 18, 2003), available at http://jya.com/a51-pi.htm

[Blu03] SIG Bluetooth. Bluetooth specification (2003) (accessed August 18, 2003), available at http://www.bluetooth.com

[BSW00] Biryukov, A., Shamir, A., Wagner, D.: Real time cryptanalysis of A5/1 on a PC. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 1–13. Springer, Heidelberg (2001)

[CP05] De Canniére, C., Preneel, B.: Trivium – a stream cipher construction inspired by block cipher design principles. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/030 (2005-04-29) (2005), http://www.ecrypt.eu.org/stream

[ECR05] eSTREAM: ECRYPT Stream Cipher Project, IST-2002-507932 (2005) (accessed September 29, 2005), available at http://www.ecrypt.eu.org/stream/

[eDF05] eSTREAM Discussion Forum. A reformulation of TRIVIUM created on 02/24/06 12:52PM (2005), http://www.ecrypt.eu.org/stream/phorum/read.php?1,448

[FM00] Fluhrer, S.R., McGrew, D.A.: Statistical analysis of the alleged RC4 keystream generator. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 19–30. Springer, Heidelberg (2001)

[HJM05] Hell, M., Johansson, T., Meier, W.: Grain V.1. — a stream cipher for constrained environments (2005), http://www.it.lth.se/grain/grainV1.pdf

[LMV05] Lu, Y., Meier, W., Vaudenay, S.: The conditional correlation attack: A practical attack on Bluetooth encryption. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 97–117. Springer, Heidelberg (2005)

[LV04] Lu, Y., Vaudenay, S.: Cryptanalysis of Bluetooth keystream generator two-level E_0. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, Springer, Heidelberg (2004)

[MJB04] Maximov, A., Johansson, T., Babbage, S.: An improved correlation attack on A5/1. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 1–18. Springer, Heidelberg (2004)

[MS01] Mantin, I., Shamir, A.: Practical attack on broadcast RC4. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 152–164. Springer, Heidelberg (2002)

[NES99] NESSIE: New European Schemes for Signatures, Integrity, and Encryption (1999) (accessed August 18, 2003), available at http://www.cryptonessie.org

[Rad06] Raddum, H.: Cryptanalytic results on TRIVIUM. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/039 (2006), http://www.ecrypt.eu.org/stream

[Sma03] Smart, N.: Cryptography: An Introduction. McGraw-Hill Education, New York (2003)

[WSLM05] Whiting, D., Schneier, B., Lucks, S., Muller, F.: Phelix - fast encryption and authentication in a single cryptographic primitive. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/020 (2005-04-29) (2005), http://www.ecrypt.eu.org/stream

Appendix A: Modification of Trivium: Trivium/128

In this section we present several modifications of the original Trivium design which improve its security against our attacks and which allow to use Trivium with 128-bit keys.

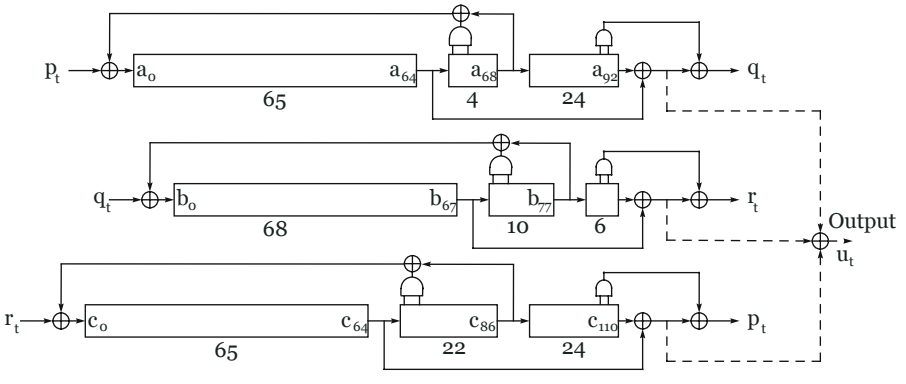

Fig. 2. A 128-bit improved design Trivium/128

Suggestions for possible improvements of Trivium are as follows.

I-1 Although clocking forward allows us to receive many linear or low degree equations, the backward clocking does not. The backward evaluation of Trivium seem to be "well-protected", since the outcome of the AND gates are connected *forward*, thus, supporting a huge avalanche effect of noise propagation when clocking backward. We suggest to introduce 3 additional gates, but connected *backward*, in order to support a similar effect when clocking forward. To keep the parallelism property (64 clocks at once), the distance between the taps and the outcome pins of the new gates should be not less than 64;

I-2 In all our attacks we used the property that the building blocks of Trivium are divisible by 3. We think that if one can remove this property, the attack could be more complex. However, this could create a risk of an existence of a good distinguisher.

According to the suggestions above we propose a modification Trivium/128, which is similar to Trivium, but possibly more secure. Trivium/128 is presented in Figure 2. We keep the size of the internal state to be 288 bits, as well

as the sizes of the nonlinear registers, 94, 84, and 111 bits, respectively. In each register the size of the first block is decreased by 1 bit, and the second block is increased by 1 bit. This would destroy the "3-divisibility" property. Moreover, in each register we introduce an additional AND gate, the inputs of which are the *first* and the *third* taps of the second block. For example, in the register A the tap positions are a_{65} and a_{67}. The new gate will make the complexity of equations to grow faster than in the original TRIVIUM, keeping the parallelism property of the cipher (ideally we should jump just a few bits back, but this would destroy parallelism). If only 32-bit parallelism suffices, then the new AND gates could jump in the middle of the first blocks in the registers. The propagation of the noise would be twice faster then. Yet another option which would have very fast growth of non-linearity would be to move the AND gates to the beginning of the long register (ex. tap positions a_1, a_2). We keep the same initialization procedure as in TRIVIUM, but with a 128 bit secret key to be loaded instead.

We believe that this tweak of the original design would resist our attacks and, possibly stay resistant against distinguishing attacks as well. We are currently checking Trivium-like designs in order to find one with best security/performance tradeoff.

Appendix B: Statistical Pre-Test for the Phase I

In the scenarios above the constant c within time complexity denotes the time needed for solving a system of equations in the second phase. Although the equations are sparse, this constant can still be large. When the number of variables is 192, we assume that this constant is approximately lower bounded as $c \approx 2^{16}$.

One idea to reduce the total time complexity is to consider only those "windows" in the stream where the probability for the guess of the AND gates is larger than in a random case.

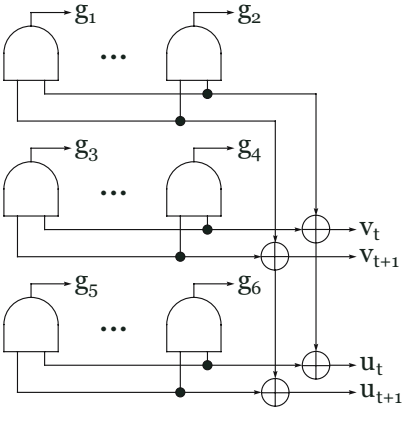

Fig. 3. Statistical pre-test

Table 10. Keystream influence for the pre-test technique

(u_t, u_{t+1})	Pr{the sum of AND gates is zero}	
(v_t, v_{t+1})	in Trivium	in Bivium
$(0,0)$	0.53125	0.625
$(0,1)$	0.5	0.5
$(1,0)$	0.5	0.5
$(1,1)$	0.5	0.5

Let us observe an output pair (u_t, u_{t+1}) (or (v_t, v_{t+1})) at some time t and $t+1$, each component of which is the sum of 6 (respectively, 4) bits of the state from $\mathcal{T}_1^{(t)}$ and $\mathcal{T}_2^{(t)}$, as shown in Figure 3. The question here is: What is the probability that the sum of six (four) AND gates is zero, given the observed pair? We can use this criteria to cut undesired cases, since the sum of the gates must be zero when all of them are zeros as well. Below, in Table 10, we give these probabilities in accordance.

I.e., when the keystream in a specified "window" is a zero sequence, then the probability of our guess, a set of specific AND gates is zero, is larger than otherwise. However, this approach would require a much longer keystream, and the gain in time complexity is not significant. More complicated tests can also be developed.

Appendix C: Statistical Post-test of the Phase I

Another approach is to make a test after the first 1/3rd of the state is guessed and derived. Let us introduce a decision rule for the test

$$\delta(\mathcal{T}_0^{(t)}) = \begin{cases} \text{Accept}, & \mathcal{T}_0^{(t)} \text{ passes the test,} \\ \text{Reject}, & \text{otherwise.} \end{cases} \tag{18}$$

Associated with the decision rule δ there are two error probabilities.

$$\begin{aligned} \alpha &= \Pr\{\delta(\mathcal{T}_0^{(t)}) = \text{Reject} | \text{the guess } \mathcal{T}_0^{(t)} \text{ is correct}\}, \\ \beta &= \Pr\{\delta(\mathcal{T}_0^{(t)}) = \text{Accept} | \text{the guess } \mathcal{T}_0^{(t)} \text{ is wrong}\}. \end{aligned} \tag{19}$$

Thus, the time complexity can be reduced from $c \cdot Q$ down to $\beta \cdot c \cdot Q$. However, the success of the attack will be $P_{\mathtt{succ}} = 1 - \alpha$. If the test is statistically strong, then α and β are small, lowering the time complexity significantly.

One such a test could be as follows. At a time t the sequence of d' triple-clocks allows us to receive d' linear equations on the bits of $\mathcal{T}_0^{(t)}$. However, if we continue clocking, we will then receive a sequence of biased samples. The bias decreases rapidly as long as the number of random AND terms in the equation for the noise variable grows.

Unfortunately, for Trivium there is no valuable gain, but for Bivium the gain is more visible. Consider the scenario B1. After the first phase the following triple-clocks give us the following samples.

AND gates in the noise, $i =$	1	2	3	4	...
Number of samples, $l_i =$	5	4	1	13	∞

Let us denote the first 23 samples (24=5+4+1+13) as $\mathbf{s}^{23} = s_0, s_1, \ldots, s_{22}$, and the decision rule for our test be

$$\delta(\mathbf{s}^{23}) = \begin{cases} \text{Accept}, & \text{if } H_w(\mathbf{s}^{23}) \geq \sigma_0, \\ \text{Reject}, & \text{otherwise}, \end{cases}$$

where $0 \leq \sigma_0 \leq 23$ is some appropriately chosen *decision threshold*. The error probabilities are then as follows.

$$\begin{aligned} \alpha &= \sum_{\substack{\forall t_w : 0 \leq t_w \leq l_w, w = 1 \ldots 4 \\ t_1 + t_2 + t_3 + t_4 < \sigma_0}} \prod_{w=1}^{4} \binom{l_w}{t_1} p_w^{t_w} (1 - p_w)^{l_w - t_w}, \\ \beta &= 2^{-23} \sum_{t=\sigma_0}^{23} \binom{23}{t}, \end{aligned} \tag{20}$$

where the probabilities p_w are calculated via (5). Additional information is extracted from the fact that the distribution of α is "shifted" with regard to the distribution of β, and, therefore, the gain can be achieved. In Table 11 these probabilities are given for several values of the threshold σ_0.

Table 11. Error probabilities for the post-test technique

σ_0	0	7	11	12	14	18	23
α	~ 0	0.0038	0.1585	0.2964	0.6275	0.9839	~ 1
β	~ 1	0.9826	0.6612	0.5000	0.2024	0.0053	~ 0

I.e., if we choose $\sigma_0 = 18$ in B1, then the time complexity will be $c \cdot 2^{30.2}$, instead of $2^{37.8}$. The length of the keystream remains the same. However, the success probability of this attack is $P_{\mathtt{succ}} = 0.0161$, which is low.

The situation with the success rate can be improved if the attack will be repeated $1/P_{\mathtt{succ}}$ times. Thus, we have the overall time complexity around $2^{5.9} \cdot 2^{30.2} = 2^{36.1}$, but the keystream is also increased till $2^{11.7}$. We could trade-off a better time complexity with the length of the keystream, and the overall success probability is around 1.

Searching for a proper statistical test is a challenge and is not an easy task.