New Distinguishers for Reduced Round Trivium and Trivia-SC using Cube Testers

Anubhab Baksi, Subhamoy Maitra, Santanu Sarkar

New Distinguishers for Reduced Round Trivium and Trivia-SC using Cube Testers (Extended Abstract)

Anubhab Baksi[1], Subhamoy Maitra[1], Santanu Sarkar[2]

[1] Indian Statistical Institute, 203 B. T. Road, Kolkata 700 108, India
anubhab91@gmail.com, subho@isical.ac.in
[2] Indian Institute of Technology Madras, Sardar Patel Road, Chennai 600 036, India
sarkar.santanu.bir@gmail.com

Abstract. In this paper we experiment with cube testers on reduced round Trivium that can act as a distinguisher. Using heuristics, we obtain several distinguishers for Trivium running more than 800 rounds (maximum 829) with cube sizes not exceeding 27. In the process, we also exploit state biases that has not been explored before. Further, we apply our techniques to analyse Trivia-SC, a stream cipher proposed by modifying the parameters of Trivium and used as a building block for TriviA-ck (an AEAD scheme, which is submitted to the ongoing CAESAR competition). We obtain distinguishers till 900 rounds of Trivia-SC with a cube size of 21 only and our results refute certain claims made by the designers. These are the best results reported so far, though our work does not affect the security claims for the ciphers with full initialization rounds, namely 1152.

Keywords. Cryptanalysis, Cube Tester, Stream Cipher, Trivium, Trivia-SC.

1 Introduction

Cube attack, introduced by Dinur and Shamir [7,8], and cube tester, introduced by Aumasson et al. [9] have attracted serious attention recently. This is now being extensively used to cryptanalyse quite a few ciphers. For our explanations, let us consider the following model: Consider that the key, IV (Initialization Vector) and suitable padding bits are loaded in the initial state of a stream cipher. We denote this initial state by $S^{(0)}$ and obtain the state $S^{(r)}$ after the r-th step of evolution. In our model, we consider that the key bits are unknown; some of the IV bits (say, v many) are tried with all the $0, 1$ possibilities; rest of the IV bits are assigned to some specific value (we mostly experiment with the all-zero set-up); and the initial padding bits are fixed. Thus, for a single key, we try out 2^v IVs, and consider the GF(2) addition of all the distinct 2^v bits corresponding to each state location (denote it by $S_i^{(r)}$ for the bit at the state location i) or the output key-stream bit $f(S^{(r)})$ after t rounds of evolution during the Key Scheduling Algorithm (KSA). One may check that $S_i^{(r)}(k)$ and $f(S^{(r)}(k))$ are functions on

the secret key bits $k = k_1, k_2, \ldots, k_\ell$, where the secret key is ℓ-bit long. Now, over a random choice of keys (much less than 2^ℓ), if one can demonstrate that the behaviour of $f(S^{(r)}(k))$ can be distinguished from an ideal random number generator, then that is considered as a distinguisher (or, distinguishing attack) for that stream cipher. Since it may not be possible to obtain a distinguisher after the full rounds (say, τ) of evolution in KSA, one may try to study till t ($< \tau$) rounds. The choice of proper v locations from the IV bits is an important task and naturally identifying a smaller set to obtain a distinguisher for larger t is the prime challenge here. We study this situation for the stream ciphers Trivium and Trivia-SC in this paper and provide new results in this direction.

The Trivium [5] stream cipher, designed by De Cannière and Preneel, has a simple structure to understand and implement. It has been selected in the eStream [6] hardware portfolio. Even after more than a decade of effort by cryptanalysts, no attack could break the security conjectures made by the designers of Trivium. It has been studied extensively and there are several results with reduced initialization rounds of Trivium (see table 2).

TriviA-ck-v1 [2] is an AEAD (Authenticated Encryption with Associated Data) scheme, designed by Chakraborti & Nandi, and submitted to the ongoing "CAESAR - the Competition for Authenticated Encryption: Security, Applicability, and Robustness" project [4]. It uses the stream cipher Trivia-SC (sometimes refered to as SCTrivia or SC-Trivia, e.g., in [2, page 18], [3]) as a component. Trivia-SC is inspired from Trivium, but with a larger state size and also a larger key size. We succeed to find distinguishers for this cipher till 900 rounds that clearly refutes certain claims made by the designers.

1.1 Description of the Ciphers

Before proceeding further, let us describe the structures of Trivium and Trivia-SC. Table 1 briefly gives an overview.

Table 1: Overview of Trivia-SC and Trivium

Cipher	Key size	IV size	State size	Initialization rounds	# Bits involved in key-stream	# Operations in key-stream
Trivium	80	80	288	1152 ($= 4 \times 288$)	6	5 $\oplus$
Trivia-SC	128	128	384	1152 ($= 3 \times 384$)	8	6 $\oplus$, 1 $\wedge$

Throughout this document, and for both the ciphers, we use the convention of numbering both the key and IV bits as $1, 2, 3, \ldots$. Accordingly, we use $k_1, k_2, k_3, \ldots$ and $x_1, x_2, x_3, \ldots$ to represent the key and IV bits, respectively. We explain Trivium with a single array S and Trivia-SC with three arrays A, B, C, though each of these can be represented in any of the two manners.

We denote the the 288-bit register in Trivium by S, and its bits by $S_1, S_2, \ldots, S_{288}$ respectively. For both Trivium and Trivia-SC; the Key Loading Algorithm (KLA), by which the key and IV are initially loaded to the register; the

Key Scheduling Algorithm (KSA) routine which is carried out after the KLA and the Pseudo-Random Generation Algorithm (PRGA) routine, which generates the required number of key-streams are given in [1, section 1.1].

1.2 Existing Works & Our Contributions

Study of cube attacks on Trivium has received serious attention in literature. Table 2 summarizes the relevant state of the art results, including our own work. Note that, one part of the work done by Knellwolf, Meier, Naya-Plasencia [12, SAC'11] (3rd row of table 2) uses certain type of weak keys.

Table 2: Overview of cube attacks and cube testers on Trivium

Author(s): Type of the attack	Cube size : Round(s)
Dinur, Shamir [7] (Eurocrypt'09) [8] (Eprint'08): Cube attack	$12:672-685$; $23:735-747$; $29:767-774$
Aumasson, Dinur, Meier, Shamir [9] (FSE'09): Cube tester	$24:772$; $30:790$ (Distinguisher) $24:842$; $27:885$ (Non-randomness)
Stankovski [10] (Indocrypt'10): Cube tester	$44:806$ (Distinguisher) $45:1026$; $54:1078$ (Non-randomness)
Knellwolf, Meier, Naya-Plasencia [12] (SAC'11): Cube tester	$25:772, 782, 789, 798$ (Distinguisher) $25:868$ (2^{31} key space); $25:953, 961$ (2^{26} key space) (Non-randomness)
Fouque, Vannet [13] (FSE'13): Cube attack	$30:784$; $37:799$
Our work: Cube tester (Distinguisher)	$13:710$; $20:766, 792$; $21:777, 801$; $22:804, 810$; $27:822, 823, 826, 829$

In this paper, we look at the Cube tester, where we have control over the IVs only (we do not manipulate the secret key bits) and try to distinguish the key-stream generated from a perfectly random scenario. One may note that results with significantly larger cubes have been reported in [10]. However, those results are hard to emulate without high end facilities. Thus, further efforts have been made later in [12,13] to obtain relatively smaller cubes that can be checked quickly on commonly available laptop or desktop computers. In that direction, our efforts provide currently the best known results as evident from table 2. Our heuristics are built over the idea of [10]. We also optimize our software implementation (written in C language) to certain level to obtain faster execution.

Trivia-SC, being a recently proposed cipher, have not attracted much cryptanalysis yet (one distinguishing attack by Xu, Zhang and Feng [11]). The designers of this cipher claimed in [2, chapter 3.1, page 15] that it may not be possible to find cube-distinguishers on round 896 or beyond. However, with our heuristics, we could discover cubes of size 13 and 21 respectively to obtain distinguishers till 897 and 900 rounds. This clearly refutes the claim of the designers.

2 Finding the Cubes: Results on Trivium

As evident from the literature, there is not much concrete theory available to obtain a good cube, and most of the efforts are based on clever heuristics. The backbone of a cube tester is to obtain a (as minimal as possible) set of public variables, and experimentally check for the presence of bias for this set, if any, at some rounds over certain iterations.

At this point, let us refer to the notations that we have described in the introduction. Since we are considering reduced round model of the ciphers, we consider as if the key-stream bits are available from just after the completion of KLA. In this regard, we have already noted that $S_i^{(r)}(k)$ are the state bits and $f(S^{(r)}(k))$ are functions on the key bits only after the t-th round. For Trivium, by $S_i^{(0)}(k)$ we mean the state bits just after the KLA and $f(S^{(0)}(k))$ is the output key-stream bit at that point. Consider that U be the set of all IVs and we consider a subset V of U, with $v = |V|$. The IVs in V will be taken as the cube.

2.1 Maximum all zero [10]

Let us first describe the basic idea of GreedyBitSet heuristic proposed in [10]. We explain that in algorithm 3. Here the idea is to obtain the set of IV bits in the cube V so that $\Pr[f(S^{(r)}(k)) = 1] = 0$. As we will keep on updating the cubes, let us refer this as indexed by V (as and when required), i.e., we require $P[f_V(S^{(r)}(k)) = 1] = 0$ for all the rounds till r. Since it is being checked whether $P[f_V(S^{(r)}(k)) = 1] = 0$, i.e., whether the function $f_V(S^{(r)}(k))$ is identically zero, only a very few run with random keys will suffice with high confidence for this test. In [10], it is also described how in each step more than one bits can be added; this was proposed so that the local optima can be better avoided. From now on, we refer to GreedyBitSet as 'maximum all zero' approach.

Algorithm 3 GreedyBitSet

Input: Key k, set of IV bits U, desired cube size t
Output: Cube V of size t
1: $V \leftarrow \emptyset$
2: **for** $i \leftarrow 0$ to $t-1$ **do**
3: $V \leftarrow \text{GreedyAdd1Bit}(k, U, V)$
4: **end for**
5: **return** V

▷ GreedyAdd1Bit(k, U, V)
Input: k, U, cube V of size n
Output: Cube V of size $n+1$
1: $BestBit \leftarrow$ None; $max \leftarrow -1$
2: **for all** $x \in U \setminus V$ **do**
3: $V' \leftarrow V \cup \{x\}$
4: i is the maximum round up to & including which all $f_{V'}(S^{(r)}(k))$'s are zero
5: **if** $i > max$ **then**
6: $max \leftarrow i$; $BestBit \leftarrow x$
7: **end if**
8: **end for**
9: **return** $V \cup \{BestBit\}$

2.2 Our Strategies: Maximum last zero & Maximum frequency of zero

Instead of the 'maximum all zero' approach [10], we consider the 'maximum last zero' & 'maximum frequency of zero' approaches. That is, instead of considering the IV bit(s) for which we get all zero up to & including a particular round, we now consider the IV bit(s) for which we get the maximum last zero and maximum frequency of zero, respectively. A bit formally, line 4 in the above algorithm should be replaced by

1. "i is the last round for which the corresponding $f_{V'}(S^{(r)}(k)) = 0$", and
2. "i is the round for which the corresponding $f_{V'}(S^{(r)}(k)) = 0$ gives maximum zero count (counted up to some fixed round, say, R)",

respectively for maximum last zero and maximum frequrncy of zero. We explain these 3 heuristics with an example below.

Consider the function for the first 10 rounds. By 0 we mean that the function is identically zero, and we denote it by ? otherwise. Let $V = \emptyset$ and for the cube $\{x_1\}$, sequence of initial $f_{\{1\}}(S^{(r)}(k))$ is $\overbrace{0\cdots0}^{6}$?0??; whereas for $\{x_2\}$ (respectively, $\{x_3\}$), say the sequence of initial $f_{\{2\}}(S^{(r)}(k))$ is $\overbrace{0\cdots0}^{5}$????0 (respectively, $f_{\{3\}}(S^{(r)}(k))$ is $\overbrace{0\cdots0}^{5}$?00?0). While the strategy of [10] will choose the first one, we will choose the second one ('maximum last zero') and the third one ('maximum frequency of zero'). Similar to [10], in our strategy too, one may consider more than one IV bits can be included at a time for getting rid of local optima. For R (up to & including which round we will be counting the frequency), in 'maximum frequency of zero', we take 1152, the full KSA round for the ciphers.

We also consider a few other issues for obtaining *good* cubes with the following ideas:

1. Once we get a cube $V = \{x_1, x_2, \ldots, x_v\}$, we also consider the cubes of size $v+1$ as $\{x_1, x_2, x_3, \ldots, x_v, x_{v+1}\}$, where x_{v+1} is chosen randomly from $\in U \setminus V$ and check its usefulness as a *good* cube.
2. Following [10,12], we sometimes consider the IV bits in V with a gap of three. This performs well for Trivium, though for Trivia-SC this is not that effective.
3. We also study GreedyBitSet heuristic along with our own heuristics. In all our experiments, we note that our heuristics outperform GreedyBitSet. Also, it does not prove that GreedyBitSet is indeed weaker to our heuristics, since some randomness is always associated and initially we test with very small runs.
4. While nothing is said about tie cases in [10], we notice that one particular tie case candidate performs much better than others. Although, it is not possible for us to check all tie case candidates with our computational power, we try to

cover as many as possible. We notice that the 'maximum last zero' heuristic gives maximum number of tie cases in comparison to other two heuristics.

5. While finding the cubes, we mostly study 3+3+3+3+2+2+2+2+1+1+... strategy, i.e., starting from an empty cube, we first set 3 IV bits at a time; then with that 3-cube, we insert another 3 IV bits; etc.

As for the convention, we denote by $\star$, $*$, $\dagger$, $\ddagger$ the cubes we obtain by using maximum last zero approach, maximum frequency of zero approach, the idea of random bit insertion, and the idea of cubes of a gap of 3, respectively.

Based on these ideas we try out many cubes with small number of iterations. Once we obtain a cube that seems promising, we go for longer runs for exact estimation of probabilities to obtain distinguishers after significant number of rounds. It is needless to mention that our method do not use any automated tool. We run a lot of programs and also need to inspect quite a large amount of output generated by ourselves before we could obtain a promisingly *good* cube.

2.3 Experimental Results

Now we describe the statistical criteria needed to detect a bias. Consider that we have two key-stream bits, one is generated from an ideally random source and the other one is the key-stream bits of a stream cipher; where the probability of ocurrance of some event is p and that of the same is $p(1+q)$, respectively, where q is significantly smaller than p. Then one can distinguish between these two in $\mathcal{O}\left(\frac{1}{pq^2}\right)$ trials, i.e., one needs to have this many key-stream bits available. This underlines one component of the complexity. Thus, if $p = \frac{1}{2}$ and $q = \pm\frac{1}{2^u}$, then the complexity is $\mathcal{O}\left(2^{2u+1}\right)$. The other component here is 2^v, where v is the cube size. That is the total complexity to obtain a cube distinguisher is 2^{v+2u+1}. Now, it can be shown that, for $\frac{1}{pq^2}$ trials the success probability of distinguishing is approximately 69%; whereas it increases to more than 99.9% with $\frac{39}{pq^2}$ trials. However, for complexity figures, we consider $\frac{1}{pq^2}$ trials as it provides success probability significantly more than 50%.

With this background, let us present some interesting experimental results in table 3 (Trivium) and table 5 (Trivia-SC). For a short-hand notation, we write $\mathcal{P}^{(r)}$ in place of $\Pr[f(S^{(r)}(k)) = 1]$. In the first colums of the tables 3 & 5, we present two rounds, r and r_0; where r gives the maximum round up to which a bias (given in column 2) can be detected with complexity given in last column; and r_0 gives the maximum round up to which an exact bias of zero (i.e., $\mathcal{P}^{(r_0)} = 0$) can be detected in 2^{v+1} complexity. One may note that the starting index for the IV bits is 1 and the key-stream couting for Trivia-SC starts from 0. We mark by those IV indices by bold fonts, which are inserted to a previous cube to yield a new cube.

One very special distinguisher in table 3 is in the first row, this cube is originally obtained from a 12 cube given in [7, table 1, row 15 - excluding the header], which gives a key-recovery attack on Trivium at round 685. We extended the given cube by inserting 23, which gives a clear distinguisher till round 710.

Table 3: Cubes obtained for Trivium giving bias in key-stream

r (r_0)	$\mathcal{P}^{(r)}$	Cube indices	(#)	Complexity
710 (686)	0.462977	1, 3, 10, 12, 14, **23**, 38, 45, 48, 50, 69, 75, 79	(13)†	$2^{21.511}$
766 (747)	0.496747	2, 4, 6, 8, 11, 13, 15, 17, 19, 26, 28, 30, 32, 37, 55, 58, 67, 73, 76, 79	(20)∗	$2^{35.528}$
777 (747)	0.491104	2, 4, 6, 8, 11, 13, 15, 17, 19, 26, 28, 30, 32, **34**, 37, 55, 58, 67, 73, 76, 79	(21)∗	$2^{33.625}$
792 (736)	0.495842	2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35, 38, 41, 47, 50, 68, 74, 77, 80	(20)‡	$2^{34.820}$
801 (741)	0.497301	2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35, 38, 41, **44**, 47, 50, 68, 74, 77, 80	(21)‡	$2^{37.067}$
804 (766)	0.497907	2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, **56**, 68, 74, 77, 80	(22)‡	$2^{38.800}$
822 (778)	0.495747	2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, **53**, 56, **59**, **62**, **65**,	(27)‡	$2^{41.755}$
823	0.496973	68, **71**, 74, 77, 80		$2^{42.736}$

In table 3 (last 4 rows), the IV bits are differing by 3 or multiples of 3. The 27-cube (last row) gives a bias for maximal round, this is the full cube of gap 3. Note that in section 2.4, we show how one can obtain a distinguisher for Trivium at 829-th round using the 27-cube mentioned here.

2.4 Exploiting Biases Observed at the State Bits of Trivium

One may note that the key-stream bit of Trivium is the XOR of six state bits. Thus, if one considers:

i. each of those six state bits are biased (this can be experimentally checked in this scenario), and
ii. each of the state bits are independent (this is actually not true; but in a stream cipher design it is expected that after quite a few rounds of initialization, the individual state bits should not have much correlation),

then one may use the Piling-up lemma [14, chapter 3.3.1]. Consider that X_1, X_2, $\ldots$, X_q are independent binary random variables. Let, $\Pr(X_i = 1) = \frac{1}{2}(1 - \epsilon_i)$, where $\epsilon_i \geq 0\ \forall i = 1, 2, \ldots, q$. Here, according to the notations of Trivium, $q = 6$ and

$$X_1 \leftarrow S_{66}, X_2 \leftarrow S_{93}, X_3 \leftarrow S_{162}, X_4 \leftarrow S_{177}, X_5 \leftarrow S_{243}, X_6 \leftarrow S_{288}.$$

Let $X = X_1 \oplus X_2 \oplus \cdots \oplus X_6$. This provides the key-stream bits in reduced round Trivium. Then $\Pr(X = 1) = \frac{1}{2}(1 - \prod_{i=1}^{6} \epsilon_i)$. One may easily note that $\epsilon = \prod_{i=1}^{6} \epsilon_i$ is much smaller than each of the ϵ_i's and thus it requires much less complexity to experimentally detect higher individual bias ϵ_i rather than the composite bias ϵ.

We record interesting results with our 27-cube (last row of table 3). Table 4 shows necessary calculations with the 27 cube, the cases of round 820, 822, 826 & 829 are given here. We can detect even the most difficult state bias (in case of ϵ_5, where the actual probabilities are 0.495017 and 0.497464 for round 826 and 829 respectively) with complexity for each cube $2^{14.298}$ and $2^{16.246}$ for round 826 and for round 829 respectively (i.e., total detection complexities for this state are given by $2^{14.298} \times 2^{27} = 2^{41.298}$ and $2^{16.246} \times 2^{27} = 2^{43.246}$ respectively, which are within our scope).

We also like to point out that with such cubes, there are certain state bits where we can obtain bias till further rounds. As for an example, at the 881-th round, the input corresponding to $X_6 \leftarrow S_{288}$ remains biased to a significant extent.

Table 4: Obtaining the bias of the key-stream bit from the biases of the state bits

Round	ϵ_1	ϵ_2	ϵ_3	ϵ_4	ϵ_5	ϵ_6	Observed (ϵ)	Estimated ($\hat{\epsilon}$)
820	0.128770	1.000000	0.962410	1.000000	0.021830	1.000000	0.010078	0.002705
822	0.137094	0.999174	0.944876	1.000000	0.018380	1.000000	0.009132	0.002379
826	0.036620	0.982358	0.791138	1.000000	0.009966	0.995456	0.000678	0.000282
829	0.019288	0.890656	0.741232	1.000000	0.005072	0.946106	0.000200	0.000061

One may also note from table 4 that, ϵ is much higher than $\hat{\epsilon}$. So, the detection complexity for the biases will be indeed less than which is governed by last columns of table 4. The piling-up lemma is not obeyed exactly, and this is not unusual; since the state bits are not actually independent. However, our analysis shows that one can assume independence with a very low margin of error.

3 Results on Trivia-SC

For Trivia-SC, the cubes and related results are presented in table 5. One challenging task here is to find a distinguisher for more than 896 rounds in the initialization as the designers commented that they could not discover any non-randomness beyond that. We show here that non-randomness can indeed be discovered till round 900 with cube size as small as 21.

Table 5: Cubes for Trivia-SC giving bias in key-stream

r (r_0)	$\mathcal{P}^{(r)}$	Cube indices	(#)	Complexity
802 (801)	0.433241	1, 2, 3, 5, 9, 24, 54, 69, 84	(9)†	$2^{15.810}$
897 (882)	0.246883	1, 2, 3, 7, 16, 22, 32, 37, 46, 52, 67, 82, 97	(13)†	$2^{15.964}$
890 (889)	0.475783	9, 10, 11, 15, 24, 25, 26, 30, 45, 60, 75, 90, 105, 106, 121, 126	(16)⋆	$2^{25.736}$
900 (884)	0.418750	4, 10, 13, 16, 19, 25, 34, 40, 46, 49, 55, 70, 79, 85, 91, 94, 100, 115, 121, 125, 127	(21)∗	$2^{27.243}$
900 (885)	0.415210	4, 10, 13, 16, 19, 25, 34, 40, **44**, 46, 49, 55, 70, 79, 85, 91, 94, 100, 115, 121, 125, 127	(22)∗	$2^{28.120}$

Further, we show that exploiting very small cubes of size 9 only, one can obtain distinguishers for more than 800 rounds in Trivia-SC. This indicates that under this kind of analysis, Trivia-SC is indeed weaker than Trivium.

We would like to point out that the state bias related analysis, as in section 2.4, cannot directly be applied on Trivia-SC because of the AND operation.

3.1 One Suggested Key-stream Expression for Trivia-SC

Here we show that, even with retaining everything as it is, the security of Trivia-SC can be greatly improved just by changing the key-steam expression of algo-

rithm. The idea here is to use one such expression for generating key-stream which has better cryptographic properties than that one chosen by the designers. We use a rotation symmetric Boolean function on 5 variables as given in [15, section 3.2.1]. The algebraic normal form of the function is: $x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_1 \wedge x_2 \oplus x_2 \wedge x_3 \oplus x_3 \wedge x_4 \oplus x_4 \wedge x_5 \oplus x_5 \wedge x_1 \oplus x_1 \wedge x_2 \wedge x_3 \oplus x_2 \wedge x_3 \wedge x_4 \oplus x_3 \wedge x_4 \wedge x_5 \oplus x_4 \wedge x_5 \wedge x_1 \oplus x_5 \wedge x_1 \wedge x_2$; we substitute x_1, x_2, ..., x_5 by B_{66}, A_{102}, C_{147}, B_{105}, A_{132} respectively; and XOR the resulting expression with $A_{66} \oplus B_{69} \oplus C_{66}$ (to make use of all the 8 state bits, as in the original design). Thus, we get our suggested key-stream expression as:

$(A_{66} \oplus B_{69} \oplus C_{66}) \oplus [B_{66} \oplus A_{102} \oplus C_{147} \oplus B_{105} \oplus A_{132} \oplus B_{66} \wedge A_{102} \oplus A_{102} \wedge C_{147} \oplus C_{147} \wedge B_{105} \oplus B_{105} \wedge A_{132} \oplus A_{132} \wedge B_{66} \oplus B_{66} \wedge A_{102} \wedge C_{147} \oplus A_{102} \wedge C_{147} \wedge B_{105} \oplus C_{147} \wedge B_{105} \wedge A_{132} \oplus B_{105} \wedge A_{132} \wedge B_{66} \oplus A_{132} \wedge B_{66} \wedge A_{102}]$.

The part inside $[\cdot]$ has the best possible cryptographic properties (in terms of nonlinearity, resiliency, algebraic degree and autocorrelation values) among all 5 variable Boolean functions. We run our heuristics on this proposed design too. The best cubes we obtain this way are given in table 6. One may note that for the same size of the cubes, the distinguishers are for much smaller rounds; e.g., we can not go beyond round 819 here.

Table 6: Cubes for our suggested variant of Trivia-SC giving bias in key-stream

r (r_0)	$\mathcal{P}^{(r)}$	Cube indices	(#)	Complexity
746 (745)	0.474401	5, 14, 44, 59, 80, 89, 119, 121, 125	$(9)\star$	$2^{18.576}$
749 (703)	0.489759	2, 5, 11, 20, 26, 41, 80, 86, 122	$(9)*$	$2^{21.219}$
781 (780)	0.142615	5, **8**, 14, **29**, **35**, 44, 59, **68**, **74**, 80, 89, **103**, **107**, 119, 121, 125	$(16)\star$	$2^{17.969}$
813 (768)	0.495023	2, 5, **7**, 11, 20, **22**, 26, **35**, 41, **56**, **71**, 80, 86, **92**, **101**, 122	$(16)*$	$2^{30.301}$
819 (804)	0.075385	2, 5, 7, 11, **14**, 20, 22, 26, 35, 41, **50**, 56, **69**, 71, 80, **83**, 86, 92, **95**, 101, 122	$(21)*$	$2^{22.472}$

Future AEAD designers, who will be relying on a stream cipher, may use this idea to make their design even better. One may argue that incorporating a different Boolean function may increase the gate count in hardware implementation. This argument has merit for a stream cipher design in constrained environment. However, for an AEAD scheme, there are additional components involved and thus, the increase in gate count with a 5-variable function will be minimal, if at all.

4 Conclusion

In this paper, we discuss some new heuristics, and also their usefulness to find bias after more than 800 (namely, 823) rounds of Trivium with small sized cubes. To the best of our knowledge, biases till such higher rounds could not be reached earlier with such small cubes, where size of the cubes do not exceed 27. In another direction, we also provide a novel idea for exploiting the state biases to estimate bias in key-stream till 829 rounds of Trivium. In fact, such state biases could be observed till 881 rounds. It is expected that further extensions can be made over

our heuristics to cryptanalyse more rounds. We also study Trivia-SC that is a part of the AEAD scheme TriviA-ck. We provide empirical evidences to show that Trivia-SC is weaker than Trivium and thus propose certain modifications in key-stream generation.

Acknowledgement. The first author likes to acknowledge the support of Co-EC, Indian Statistical Institute, Kolkata towards this research.

References

1. A. Baksi, S. Maitra, S. Sarkar. New Distinguishers for Reduced Round Trivium and Trivia-SC using Cube Testers (Extended Abstract). Available at `http://eprint.iacr.org/2015/223.pdf`.
2. A. Chakraborti, M. Nandi. TriviA-ck-v1. Available at `http://competitions.cr.yp.to/round1/triviackv1.pdf`.
3. A. Chakraborti, M. Nandi. Important Features and Flexibilities of TriviA. Presentation at DIAC 2014, available at `http://2014.diac.cr.yp.to/slides/nandi-trivia.pdf`.
4. CAESAR: Competition for authenticated encryption: Security, applicability, and robustness. Available at `http://competitions.cr.yp.to/caesar.html`.
5. C. De Cannière, B. Preneel. Trivium. Available at `http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf`.
6. eSTREAM: the ECRYPT Stream Cipher Project. Available at `http://www.ecrypt.eu.org/stream/`.
7. I. Dinur, A. Shamir. Cube Attacks on Tweakable Black Box Polynomials. In Eurocrypt 2009, LNCS, Vol. 5479, pp. 278-299, 2009.
8. I. Dinur, A. Shamir. Cube Attacks on Tweakable Black Box Polynomials. In Eprint, 2008. Available at `http://eprint.iacr.org/2008/385.pdf`.
9. J-P. Aumasson, I. Dinur, W. Meier, A. Shamir. Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In FSE 2009, LNCS, Vol. 5665, pp. 1–22, 2009.
10. P. Stankovski. Greedy Distinguishers and Nonrandomness Detectors. In Indocrypt 2010, LNCS, Vol. 6498, pp. 210–226, 2010.
11. C. Xu, B. Zhang, D. Feng. Linear Cryptanalysis of FASER128/256 and TriviA-ck. In Indocrypt 2014, LNCS, Vol. 8885, pp. 237–254, 2014.
12. S. Knellwolf, W. Meier, M. Naya-Plasencia. Conditional Differential Cryptanalysis of Trivium and KATAN. In SAC 2011, LNCS, Vol. 7118, pp. 200–212, 2011.
13. P.-A. Fouque, T. Vannet. Improving Key Recovery to 784 and 799 Rounds of Trivium Using Optimized Cube Attacks. In FSE 2013, LNCS, Vol. 8424, pp. 502–517, 2013.
14. D. R. Stinson. Cryptography Theory and Practice. Chapman & Hall/CRC. Third Edition, 2006.
15. P. Stănică, S. Maitra. Rotation symmetric Boolean functions—Count and cryptographic properties. In Discrete Applied Mathematics 156(10): 1567-1580, 2008.