Algebraic analysis of Trivium-like ciphers (Poster)

Sui-Guan Teo[1] **Kenneth Koon-Ho Wong**[1] **Harry Bartlett**[2]
Leonie Simpson[2] **Ed Dawson**[1]

[1]Institute for Future Environments
[2]Science and Engineering Faculty
Queensland University of Technology
2 George Street, Brisbane QLD 4000, Australia
{teosuiguan, kwong87}@gmail.com, {h.bartlett, lr.simpson, e.dawson}@qut.edu.au

Abstract

Trivium is a bit-based stream cipher in the final portfolio of the eSTREAM project. In this paper, we apply the algebraic attack approach of Berbain et al. to Trivium-like ciphers and perform new analyses on them. We demonstrate a new algebraic attack on Bivium-A. This attack requires less time and memory than previous techniques to recover Bivium-A's initial state. Though our attacks on Bivium-B, Trivium and Trivium-N are worse than exhaustive keysearch, the systems of equations which are constructed are smaller and less complex compared to previous algebraic analyses. We also answer an open question posed by Berbain et al. on the feasibility of applying their technique on Trivium-like ciphers. Factors which can affect the complexity of our attack on Trivium-like ciphers are discussed in detail. Analysis of Bivium-B and Trivium-N are omitted from this manuscript. The full version of this paper is available on the IACR ePrint Archive.

1 Introduction

Trivium (Canniére and Preneel, 2005) is a bit-based stream cipher selected in the final portfolio of the eSTREAM project (Robshaw, 2008). Trivium uses an 80-bit key and an 80-bit IV to initialise a 288-bit nonlinear feedback shift register (NLFSR). Each key-IV pair can be used to generate up to 2^{64} bits of keystream. Trivium's structural simplicity makes it a popular cipher to cryptanalyse, but to date, no attacks in the public literature are faster than exhaustive keysearch.

Algebraic attacks (Courtois and Meier, 2003) are commonly applied to stream ciphers based on shift registers. To attack Trivium, Raddum (2006) used an algebraic relabelling technique, where the state-update bits are represented using new variables, instead of nonlinear combinations of initial state bits (Courtois and Pieprzyk, 2002). This prevents equations of high degrees from being generated. For keystream generators which use a linear output function (as Trivium-like ciphers do), Berbain et. al. (2009) expressed new feedback bits of an NLFSR as linear combinations of keystream bits and internal state bits. By doing so, the equations representing the feedback bits of an NLFSR will always be linear. Berbain et al. claim that their technique can be extended to ciphers in which $q > 1$ bits of internal state are non-linearly updated at each step and q or more linear combinations of the state are output as keystream. However, whether these techniques can be extended to ciphers in which $q > 1$ bits of internal state are non-linearly updated, while only $q' < q$ linear combinations of the state-bits are output, has not been demonstrated and is posed as an open question (Berbain et. al., 2009).

1.1 Contributions of paper

Our algebraic analysis on Trivium-like ciphers, we provide an answer to Berbain et al.'s open question. Specifically, we use Bivium-A/B, Trivium and Trivium-N as case-studies, as $q' < q$ in all cases. We apply Berbain et al.'s method of representing the feedback bits as linear combinations of internal state bits and keystream bits.

To assist us in our analysis, we introduce a new variable j, which describes the number of registers the keystream generation function takes inputs from. We show that the value of j has a significant impact on the success of our algebraic attack on these ciphers. The values of q', q and j for the Trivium-like ciphers are given in Table 1. Some improvements achieved with this new method compared to existing one are presented. Additionally, we investigate the effect which varying the keystream function and feedback bit positions have on the complexity of our analysis on the Trivium family.

	Bivium-A	Bivium-B	Trivium	Trivium-N
q	2	2	3	3
q'	1	1	1	1
j	1	2	3	3

Table 1: Parameters for Trivium-like stream ciphers

2 Trivium and its variants

Trivium is commonly represented in the literature as being based on three non-autonomous binary NLFSRs: A, B, and C, of sizes 93, 84 and 111 bits respectively (Bernstein, 2006). We omit the description of the initialisation process for the cipher, as it has no impact on our analysis. The reader is referred to the specifications of Trivium (Canniére and Preneel, 2005) for a full treatment of its initialisation processes. Let A_i denote the stages for register A and $A_i(t)$ represent the contents of A_i at time t, for $0 \leq i \leq 92$. Similar notations are used for registers B and C. The state-update functions of Trivium are as

follows:

$$A_i(t+1) = \begin{cases} A_{24}(t) \oplus C_{45}(t) \\ \quad \oplus C_0(t) \oplus C_1(t)C_2(t) & i = 92, \\ A_{i+1}(t) & 0 \le i \le 91. \end{cases}$$

$$B_i(t+1) = \begin{cases} B_6(t) \oplus A_{27}(t) \\ \quad \oplus A_0(t) \oplus A_1(t)A_2(t) & i = 83, \\ B_{i+1}(t) & 0 \le i \le 82. \end{cases}$$

$$C_i(t+1) = \begin{cases} C_{24}(t) \oplus B_{15}(t) \\ \quad \oplus B_0(t) \oplus B_1(t)B_2(t) & i = 110, \\ C_{i+1}(t) & 0 \le i \le 109. \end{cases}$$

At time t, Trivium's output function generates a keystream bit as follows:

$$z(t) = A_{27}(t) \oplus A_0(t) \oplus B_{15}(t) \oplus B_0(t) \\ \oplus C_{45}(t) \oplus C_0(t) \ , \ t \ge 0$$

We extend this representation of Trivium and consider the keystream as a sequence related to the three underlying register sequences. The initial state sequences of A, B, C are $(A_0, A_1, \ldots, A_{92})$, $(B_0, B_1, \ldots, B_{83})$ and $(C_0, C_1, \ldots, C_{110})$ respectively. New sequence bits $A_{\alpha+92}$, $B_{\alpha+83}$ and $C_{\alpha+110}$ are produced after α iterations of Trivium's state-update function as follows:

$$A_{\alpha+92} = A_{\alpha+23} \oplus C_{\alpha+44} \oplus C_{\alpha-1} \oplus C_\alpha C_{\alpha+1} \quad (1)$$

$$B_{\alpha+83} = B_{\alpha+5} \oplus A_{\alpha+26} \oplus A_{\alpha-1} \oplus A_\alpha A_{\alpha+1} \quad (2)$$

$$C_{\alpha+110} = C_{\alpha+23} \oplus B_{\alpha+14} \oplus B_{\alpha-1} \oplus B_\alpha B_{\alpha+1} \quad (3)$$

The keystream bit can be expressed as a linear combination of sequence bits from A, B and C as follows:

$$z_{\alpha-1} = A_{\alpha+26} \oplus A_{\alpha-1} \\ \oplus B_{\alpha+14} \oplus B_{\alpha-1} \oplus C_{\alpha+44} \oplus C_{\alpha-1} \quad (4)$$

Note that the sequence-based approach is analogous to the relabelling approach of Raddum (2006). In this paper, we use these sequence equations in our algebraic analysis.

Bivium-A/B are reduced versions of Trivium utilising only two registers A, B with slightly modified feedback functions. The reader is referred to Raddum (2006) for their specificiations. In Bivium-A, the keystream bit is generated using

$$z_{\alpha-1} = B_{\alpha+14} \oplus B_{\alpha-1}$$

which is composed with stages from only one register. The key and IV sizes remain the same as Trivium.

2.1 Existing algebraic cryptanalysis

Raddum (2006) constructed a system of equations for Trivium consisting of 954 equations in 954 variables. Applying techniques from graph theory, it was estimated that the initial state of Trivium can be recovered in about 2^{164} operations. Several attempts have been made to solve Raddum's system of equation, (Simonetti et. al., 2008; Borghoff et. al., 2011; Schilling and Raddum, 2012a,b). Other attacks on Trivium include SAT-solvers (McDonald et. al., 2008) and cube attacks (Dinur and Shamir, 2009; Aumasson et. al., 2009; Forque and Vannet, 2013). However, none were better than exhaustive keysearch for the original Trivium proposal.

Raddum (2006) used a relabelling approach to recover the initial state of Bivium-A in about a day. He estimated that the complexity of recovering the initial state of Bivium-B will take about 2^{56} seconds. Eibach et. al. (2010) achieved $2^{39.12}$ for Bivium-B with some optimisations. Other algebraic analysis on Bivium-A/B use Boolean Satisfiability (SAT) solvers (Eibach et. al., 2008; McDonald et. al., 2008) to recover the initial state, which are also better than exhaustive keysearch.

3 New analysis of Trivium-like ciphers

In this section, we apply Berbain et al.'s approach to the analysis of Bivium-A and Trivium. Similar analyses have been applied to Bivium-B and Trivium-N, which are available in the full version of our paper.

3.1 New analysis of Bivium-A

Bivium-A's keystream bit $z_{\alpha-1}$ depends on two sequence bits produced by register B. The sequence bits of B after 83 iterations are unknown. Applying Berbain at al.'s technique to the equation we can determine the equation for calculating the sequence bit $B_{\alpha+83}$ for $\alpha \ge 1$:

$$B_{\alpha+83} = z_{\alpha+68} \oplus B_{\alpha+68}$$

We present a a divide-and-conquer approach to recover the initial state of Bivium-A. This involves first forming a system of equations to recover the contents of register B. Another system of equations is then formed using the now known contents of register B to recover the contents of register A. The first system of equations consists of the following equations:

$$z_{\alpha-1} \oplus B_{\alpha-1} \oplus B_{\alpha+14} = 0 \quad (5)$$

$$B_{\alpha+83} \oplus z_{\alpha+68} \oplus B_{\alpha+68} = 0 \quad (6)$$

where Equation 5 is a keystream equation for Bivium-A and Equation 6 the new equation representing Bivium-A's sequence bit $B_{\alpha+83}$ derived from our new analysis. For the first 69 iterations, we add two equations and one variable into the system of equations: one equation representing the keystream, and one equation and variable representing the sequence bit for register B. For the last 39 iterations, we only add the equations representing the sequence bits for B. This gives us a final system of equations after 108 iterations consisting of 192 variables in 177 equations. Solving this system of equations gives 2^{15} possible solutions.

For each of these 2^{15} possible solutions, we form the second system of equations to recover the initial state of A, substituting the sequence bits of B recovered in the first system of equations into the second system of equations. The second set of equations equations consists of:

$$A_{\alpha+92} \oplus B_{\alpha+14} \oplus B_{\alpha-1} \oplus B_\alpha B_{\alpha+1} \oplus A_{\alpha+23} = 0$$
$$B_{\alpha+83} \oplus A_{\alpha+26} \oplus A_{\alpha-1} \oplus A_\alpha A_{\alpha+1} \oplus B_{\alpha+5} = 0$$

After 108 iterations, we have a system of equations consisting of 201 variables in 216 equations. If the system can be solved, then the sequences of A and B are recovered. An attacker can then use this initial state to generate some keystream to check the validty of the recovered initial state.

3.1.1 Comparison of attacks

Details of the systems of equations formed for using Raddum's approach and our approach are shown in

Technique	Lin eqns	Quad eqns	Total eqns	K.S. (bits)	Vars	Exp sols
Raddum	177	222	399	177	399	1
Our approach:						
- Step 1	177	0	177	177	192	2^{15}
- Step 2	108	108	216	0	201	1

Column headings: Number of Linear and Quadratic Equations, Total Number of Equations, followed by Keystream bits required, Number of Variables, and Number of Expected Solutions

Table 2: Details for the system of equations in Bivium-A for both approaches.

Table 2. As we are solving two systems of equations separately, the complexity of our attack is likely to be less than that using Raddum's technique. We investigate this with experimental work. The results are shown in Table 3. The $F4$ implementation in Magma

No Guessing:	Time	Memory (MB)	Keystream (bits)
(Raddum, 2006)	DNF	87040	177
Our approach	13.9 hrs	135.56	177
Load 15 correct bits into register B:	**Time**	**Memory (MB)**	**Keystream (bits)**
(Raddum, 2006)	8.95 s	13.34	177
Our approach	3.29 s	13.31	177

Table 3: Time, memory, and data complexities for recovering initial state of Bivium-A

(Bosma et. al., 1997) was used. Note that this is equivalent to Gaussian elimination in Step 1 of our approach. We attempt to solve the system of equations in two ways: (1) without guessing any bits and (2) load 15 correct initial state bits of Register B.

Overall, our approach gave more favourable results under both scenarios. The full attack wth no guessing did not finish (DNF) using Raddum's attack after exhausting the assigned 85 GB of memory. Although McDonald et. al. (2008) solved the Bivium-A system in 16 seconds using SAT solvers, they reported 4660 hours to solve the same system using F4. It is difficult to directly compare the effectiveness of our approach against others (Raddum, 2006; McDonald et. al., 2008) as different hardware platforms and software implementations can have significant effects on the time and memory required to recover the initial state of Bivium-A.

3.2 New algebraic analysis on Trivium

Using Berbain et al.'s approach, we can determine the equation describing the sequence bits for registers A, B and C:

$$\begin{aligned} A_{\alpha+92} &= z_{\alpha+65} \oplus A_{\alpha+65} \\ &\quad \oplus B_{\alpha+80} \oplus B_{\alpha+65} \oplus C_{\alpha+110} \oplus C_{\alpha+65} \\ B_{\alpha+83} &= z_{\alpha+68} \oplus A_{\alpha+95} \oplus A_{\alpha+68} \\ &\quad \oplus B_{\alpha+68} \oplus C_{\alpha+112} \oplus C_{\alpha+67} \\ C_{\alpha+110} &= z_{\alpha+65} \oplus A_{\alpha+92} \oplus A_{\alpha+65} \\ &\quad \oplus B_{\alpha+80} \oplus B_{\alpha+65} \oplus C_{\alpha+65} \end{aligned}$$

However, we can not use all three sets of equations simultaneously since they are actually equivalent. For example, calculating B_{84} requires knowledge of the sequence bit A_{96}. The equation representing A_{96} is:

$$A_{96} = z_{69} \oplus A_{69} \oplus B_{84} \oplus B_{69} \oplus C_{114} \oplus C_{69} \qquad (7)$$

If we substitute Equation 7 into the equation representing the sequence bit B_{84}, we get the trivial equation $B_{84} = B_{84}$. This does not allow us to express B_{84} in terms of the sequence bits and keystream. Therefore, our technique only allows us to express the sequence bits for a single register in terms of internal state and keystream bits. Therefore, the divide-and-conquer approach used in our analysis of Bivium-A cannot be applied here.

In the following analysis, we express the sequence bits $A_{\alpha+93}$, $B_{\alpha+84}$, and $C_{\alpha+111}$ for $\alpha \geq 1$ using the sequence update function described in Section 2. This new system of equations starts off with 288 variables representing the inital state bits. For each of the first 66 iterations, we add three new variables and four new equations to the system of equations, consisting of:

$$\begin{aligned} A_{\alpha+92} \oplus z_{\alpha+65} \oplus A_{\alpha+65} \oplus B_{\alpha+80} \\ \oplus B_{\alpha+65} \oplus C_{\alpha+110} \oplus C_{\alpha+65} = 0 \\ B_{\alpha+83} \oplus B_{\alpha+5} \oplus A_{\alpha+26} \oplus A_{\alpha-1} \oplus A_\alpha A_{\alpha+1} = 0 \\ C_{\alpha+110} \oplus C_{\alpha+23} \oplus B_{\alpha+14} \oplus B_{\alpha-1} \oplus B_\alpha B_{\alpha+1} = 0 \\ z_{\alpha-1} \oplus A_{\alpha+26} \oplus A_{\alpha-1} \oplus B_{\alpha+14} \\ \oplus B_{\alpha-1} \oplus C_{\alpha+44} \oplus C_{\alpha-1} = 0 \end{aligned}$$

After 66 iterations, similar to Bivium-A, the keystream equations become redundant, and we can stop adding variables and equations into the system. This gives us a final system of equations consisting of 486 variables in 264 equations. The comparison of both systems of equations is shown in Table 4.

Technique	Lin eqn.	Quad eqn.	Total eqn.	K.S. (bits)	Vars	Exp sols
Raddum	288	666	954	288	954	1
Our approach	132	132	264	132	486	2^{222}

Column headings as per Table 2

Table 4: Details for the system of equations in Trivium

Our system of equations has less quadratic equations compared to Raddum's technique. Raddum's technique has 666 quadratic equations, compared to 132 quadratic equations in ours. The drawback of our system of equations however, is that our system of equations have a greater excess of variables over equations to Raddum's technique. In Raddum's technique, solving a system consisting of 954 variables in 954 equations should yield a unique solution. In contrast, solving our system of equations consisting of 486 variables in 264 equations using the F4 algorithm will yield 2^{222} possible solutions, which is worse than exhaustive keysearch. This will be discussed further in Section 4.2.

4 On Berbain et al.'s Open Question

In this section, we answer the open question posed by Berbain et. al. (2009) by showing how it may be possible to recover the initial states of some Trivium-like ciphers faster than exhaustive keysearch using our analysis. Two factors are considered:

- The relationship between j (the number of registers the keystream generation function takes as input to generate keystream) and q (the number of registers whose internal state is updated at each iteration).

- The largest index among the stages in a register used for the output function and the size of each register.

4.1 Relationship between j and q

In the case of Bivium-A, where $j < q$, we recovered the initial state of the cipher with a complexity less than Raddum's relabelling technique, which is already less than exhaustive key search. The complexity T_f of our new approach on Trivium-like ciphers with $j < q$ is

$$T_f = T_1 + (N_A \times T_2) \tag{8}$$

where T_1 and T_2 are the time required to solve the first and second system of equations respectively, and N_A is the number of solutions obtained in solving the first system of equations.

For Bivium-B, Trivium and Trivium-N, we have $j = q$. In this case, it is not possible to use a divide-and-conquer approach to recover the initial state of the keystream generators. A single system of equations is needed. However, using our approach to build this system of equations is problematic. Since a keystream equation is essentially being used to represent a sequence bit, additional keystream equations become unavailable after after a certain number of iterations. Adding further equations does not allow us to reduce the number of solutions obtained when the system of equations is solved and also adds to the complexity of solving these equations.

The complexity of recovering the initial state of Trivium-like ciphers where $j = q$ is $T_f = T_s + N_A$, where T_s is the time taken to solve the system of equations and N_A is the number of solutions obtained when the system of equations is solved.

4.2 Output functions, registers sizes, and N_A

The number of solutions N_A obtained when solving Trivium-like equation systems can be determined from the cipher specifications. Let S_R be the total size of the registers whose sequence bit(s) is not written in terms of keystream and internal state bits *and* whose stages are used during the construction of a system of equations for the cipher. When $j < q$, this system of equations is the first system of equations. When $j = q$, this system of equation is the sole system of equations obtained. For the register whose sequence bit is written in terms of keystream and internal state bits, let D_l denote the largest index among the stages used as input to the output function. The size of the set of solutions, N_A obtained when the system of equations is solved has been found to be:

$$N_A = 2^{S_R} \times 2^{D_l} \tag{9}$$

A summary of the results of our analyses of certain Trivium-like ciphers with regards to the number of equations, variables, solutions obtained, and their relationship with S_R and D_l, is given in Table 5.

Cipher			Step 1			Step 2		
	S_R	D_l	Eqn	Var	N_A	Eqn	Var	Soln
Bivium-A	0	15	177	192	2^{15}	216	201	1
Bivium-B	84	27	198	309	2^{111}	N.A	N.A	N.A
Trivium	195	27	264	486	2^{222}	N.A	N.A	N.A

Table 5: Details on systems of equations in our approaches for certain Trivium-like ciphers

Our new attack on Bivium-A can be prevented by making changes to the keystream output function. For example, suppose the output function of Bivium-A was, instead:

$$z_{\alpha-1} = B_{\alpha-1} \oplus B_{\alpha+82}$$

The sequence bit B_{84}, rewritten in terms of linear combinations of keystream and internal state bits, is now $B_{84} = z_1 \oplus B_1$. In this example, $D_l = 83$ and $S_R = 0$, so $N_A = 2^{S_R} \times 2^{D_l} = 2^0 \times 2^{83} = 2^{83}$, which is larger than the number of possible keys.

Conversely, assume D_l is a small value. We use Trivium to illustrate how this change reduces the number of solutions obtained when the system of equations was solved. Assume that the output function for Trivium is instead:

$$z_{\alpha-1} = A_\alpha \oplus A_{\alpha-1} \oplus B_{\alpha+14} \oplus B_{\alpha-1} \oplus C_{\alpha+44} \oplus C_{\alpha-1}$$

In this case, $D_l = 1$ and $S_R = 111 + 84 = 195$, which gives $N_A = 2^{S_R} \times 2^{D_l} = 2^{195} \times 2^1 = 2^{196}$, a decrease by a factor of 2^{26} in the number of possible solutions compared to 2^{222} obtained from our analysis of Trivium in Section 3.2.

The size of the registers used in the formation of the first system of equations can have also an effect on the number of solutions obtained for the first system of equations. For example, suppose that Trivium's three registers, A, B, and C had lengths 198, 45 and 45 bits, where the output function used is the same as the original Trivium proposal. For this particular case, $D_l = 27, S_R = 45 + 45$, and $N_A = 2^{90} \times 2^{27} = 2^{117}$, a 2^{105} factor decrease in the number of possible solutions compared to the original Trivium.

5 Conclusion

This paper analysed Trivium-like ciphers using the approach of Berbain et al. Our analysis answers their open question and shows that it is possible, in some cases, to extend their technique to keystream generators which update $q > 1$ bits of internal state at each iteration but only output $q' < q$ linear combinations of the state bits.

In particular, we demonstrated a new algebraic attack on Bivium-A. Our approach requires less time and memory than previous techniques We demonstrated that if $j < q$, it may be possible to mount a divide-and-conquer algebraic attack which is more efficient than exhaustive key search.

For Trivium-like ciphers, we showed that both the sizes of the registers used in the construction of the first system of equations and the selection of stages used as input to the output function can affect the complexity of our attack. For Bivium-A, changing the value of D_l can increase the complexity of the attack. However, in the case of Bivium-B, Trivium and Trivium-N, even if the value of D_l is small, the complexity of our algebraic attack is still worse than exhaustive keysearch as the value of S_R is larger than the keysize.

Acknowledgements

The authors would like to thank anonymous reviewers for their helpful comments. Computational resources and services used in this work were provided by the HPC and Research Support Unit at Queensland University of Technology, Brisbane, Australia.

References

Akers, S. B. (1978), Binary Decision Diagrams. *IEEE Transactions on Computers* **27**(6), 509–516.

Aumasson, J. P., Dinur, I., Meier, W. and Shamir, A. (2009), Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. *in* O. Dunkelman, ed, 'Fast Software Encryption (FSE 2009)'. Lecture Notes in Computer Science, Vol. 5665, Springer, pp. 1–22.

Berbain, C., Gilbert, H. and Joux, A. (2009), Algebraic and Correlation Attacks against Linearly Filtered Non Linear Feedback Shift Registers, *in* R.M. Avanzi, L. Keliher and F. Sica, eds, 'Selected Areas in Cryptography (SAC 2008)'. Lecture Notes in Computer Science, Vol. 5381, Springer. pp. 184–198.

Bernstein, D. , A reformulation of TRIVIUM. Submission to Phorum: ECRYPT forum. `http://www.ecrypt.eu.org/stream/phorum/read.php?1,448`

Borghoff, J., Knudsen, L.R. and Matusiewicz, K. (2010), Hill Climbing Algorithms and Trivium. *in* A. Biryukov, G. Gong and D. R. Stinson, eds, 'Selected Areas in Cryptography (SAC 2010)'. Lecture Notes in Computer Science, Vol. 6544, Springer, pp. 57–73.

Bosma, W., Cannon, J. J. and Playoust, C. (1997), The Magma algebra system. I. The user language. *Journal of Symbolic Computation* **24**(3-4), 235–265.

Canniére, C. D. and Preneel, B. , Trivium, *in* M.J.B. Robshaw, O. Billet, eds, 'New Stream Cipher Designs: The eSTREAM Finalists'. Lecture Notes in Computer Science, Vol. 4986, Springer, pp. 244–266.

Courtois, N. T. and Meier, W. (2003), Algebraic Attacks on Stream Ciphers with Linear Feedback, *in* E. Biham, ed, 'Advances in Cryptology — EUROCRYPT 2003'. Lecture Notes in Computer Science, Vol. 2656, Springer, pp. 345–359.

Courtois, N. T. and Pieprzyk, J. (2002), Cryptanalysis of Block Ciphers with Overdefined Systems of Equations, *in* Y. Zheng, ed, 'Advances in Cryptology — ASIACRPYT 2002'. Lecture Notes in Computer Science, Vol. 2501, Springer, pp. 267-287.

Dinur, I., Shamir, A. (2009), Cube Attacks on Tweakable Black Box Polynomials, *in* A. Joux, ed, 'Advances in Cryptology — EUROCRYPT 2009'. Lecture Notes in Computer Science, Vol. 5479, Springer, pp. 278–299.

Eibach, T., Pilz, E. and Völkel, G. (2008), Attacking Bivium Using SAT Solvers, *in* H.K. Büning, X. Zhao, eds, 'Theory and Applications of Satisfiability Testing — SAT 2008'. Lecture Notes in Computer Science, Vol. 4996, Springer, pp. 63–76.

Eibach, T., Pilz, E. and Völkel, G. (2010), Optimising Gröbner Bases on Bivium. *Mathematics in Computer Science* **3**(2), 159–172.

Robshaw, M. (2008), New Stream Cipher Designs. Lecture Notes in Computer Science, Vol. 4986, Springer.

Faugère, J. C. (1999), A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra **139**, 61–88.

Fouque, P. A., Vannet, T. (2013), Improving Key Recovery to 784 and 799 rounds of Trivium using Optimized Cube Attacks, *in* S. Moriai, ed, 'Fast Software Encryption (FSE 2013)'. To appear.

McDonald, C., Charnes, C., Pieprzyk, J. (2008), An Algebraic Analysis of Trivium Ciphers based on the Boolean Satisfiability Problem, *in* 'Fourth International Workshop on Boolean Functions: Cryptography and Applications', pp. 173-184.

Raddum, H. (2006), Cryptanalytic Results on Trivium. *eSTREAM, ECRYPT Stream Cipher Project*, Report 2006/039. `http://www.ecrypt.eu.org/stream/papersdir/2006/039.ps`

Schilling, T. E. and Raddum, H. (2012), Analysis of Trivium Using Compressed Right Hand Side Equations, *in* H. Kim, ed, 'Information Security and Cryptology (ICISC 2011)'. Lecture Notes in Computer Science, Vol. 7259, Springer, pp. 18–32.

Schilling, T. E. and Raddum, H. (2012), Solving Compressed Right Hand Side Equation Systems with Linear Absorption. *in* T. Helleseth, J. Jedwab, eds, 'Sequences and Their Applications (SETA 2012)'. Lecture Notes in Computer Science, Vol. 7280, Springer. pp. 291–302.

Simonetti, I., Perret, L., Faugère, J. C. (2008), Algebraic Attack Against Trivium, *in* 'First International Conference on Symbolic Computation and Cryptography, SCC 2008'. LMIB. pp. 95–102.