Guess and Determine Attack on Bivium

Neda Rohani*, Zainab Noferesti*, Javad Mohajeri**
and Mohammad Reza Aref*

Abstract—Bivium is a simplified version of Trivium, a hardware profile finalist of the eSTREAM project. Bivium has an internal state size of 177 bits and a key length of 80 bits. In this paper, a guess and determine attack on this cipher is introduced. In the proposed method, the best linear approximations for the updating functions are first defined. Then by using these calculated approximations, a system of linear equations is built. By guessing 30 bits of internal state, the system is solved and all the other 147 remaining bits are determined. The complexity of the attack is O (2^{30}), which is an improvement to the previous guess and determine attack with a complexity of order O($2^{52.3}$).

Keywords—Bivium, Guess and Determine Attack, Stream Ciphers, Linear Approximations, Entropy

1. INTRODUCTION

Cryptography is the practice and study of hiding information. It uses two main styles of encryption: symmetrical and asymmetrical. Symmetric algorithms use the same key for encryption as they do for decryption, while asymmetric algorithms use two different keys for encryption and decryption.

Symmetric cryptography schemes are generally categorized as being either stream ciphers or block ciphers. Stream ciphers generate a very long keystream and use it to encrypt a single bit, byte or word at a time. A block cipher encrypts one block of data at a time using the same key on each block. In general, a plaintext block will always encrypt to the same cipher text in a block cipher, whereas the same plaintext will encrypt to a different cipher text in a stream cipher using the same key.

A stream cipher is a symmetric key cipher where plaintext bits are combined with a pseudo-random cipher bit stream by a specific function. Stream ciphers typically execute at a higher speed than block ciphers and have lower hardware complexity.

A stream cipher generates successive elements of the keystream based on an internal state. This state is updated in two different ways: if the alterations of the state are independent of the plaintext or cipher text messages, the cipher is defined as a synchronous stream cipher. Otherwise the algorithm will be a self-synchronizing stream cipher in which the state is updated according to the previous cipher text digits. While stream ciphers do not propagate transmission errors, they are, by their nature, periodic so the keystream will eventually be repeated.

※ This work was partially supported by the Iran National Science Foundation (INSF) - cryptography chair and by the Iran Telecommunications Research Center (ITRC).
Manuscript received August 6, 2010; accepted September 2, 2010.
Corresponding Author: Neda Rohani
* Information System and Security Lab (ISSL), Dept. of Electrical Engineering, Sharif University of Technology, Tehran, Iran (n_rohani@ee.sharif.edu, znoferesti@ee.sharif.edu, aref@sharif.edu)
** Electronics Research Center, Sharif University of Technology, Tehran, Iran (mohajer@sharif.ir)

 Copyright ⓒ 2011 KIPS (ISSN 1976-913X)

In 2004, the European Network of Excellence in Cryptology (ECRYPT) launched a project on stream ciphers, eSTREAM [1], with a focus on introducing practical stream ciphers with an acceptable level of security. The main evaluation criteria were likely to be long-term security, efficiency (performance), flexibility and market requirements.

The project was launched in two profiles: software and hardware. Software profile candidates were meant to be stream ciphers for software applications with high throughput requirements. Hardware profile candidates were to be designed for hardware applications with restricted resources such as limited storage, gate count, or power consumption.

Bivium [2] is a simplified version of Trivium [3], a synchronous stream cipher submitted to this project as a hardware profile candidate. Trivium has been selected as one of the portfolio finalists. Bivium has less internal state variables. Security, speed and simplicity are three important characteristics of its design. The previous guess and determine attack on Bivium was performed by McDonald, Charnes, Pieprzyk [4], with a complexity of order O ($2^{52.3}$).

Due to Bivium's low nonlinearity and existence of linear approximations with good bias, we were able to perform a guess and determine attack on this cipher with a complexity of order O (2^{30}).

This paper is organized as following: In the next section a description of the Bivium stream cipher is given. The proposed guess and determine attack on Bivium is described in section 3 and section 4 provides a comparison between related works, and finally section 5 concludes this paper.

2. CIPHER SPECIFICATION

The internal state of Bivium consists of 177 bits initialized by an 80-bit key and an 80-bit IV during an initialization phase. In every step, two bits are updated according to nonlinear update functions and the others are updated as in a linear shift register.

Throughout this document '+' and '.' operators will denote addition and multiplication over GF (2), respectively.

2.1 Keystream Generation

Denoting the state variables by $s_t^1, s_t^2, ..., s_t^{177}$, the keystream generation is described according to the following pseudo-code [3]:

for i = 1 to N do

$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$

$z_i \leftarrow t_1 + t_2$

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

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{69}$

$(s_1, s_2, ..., s_{93}) \leftarrow (t_2, s_1, ..., s_{92})$

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

end for

We will denote $(s_t^1, s_t^2, ..., s_t^{93})$ by $(a_t^1, a_t^2, ..., a_t^{93})$ and $(s_t^{94}, s_t^{95}, ..., s_t^{177})$ by $(b_t^1, b_t^2, ..., b_t^{84})$. The ci-

Keystream Generation Pseudo-code

$a_{t+1}^1 \leftarrow b_t^{69} \oplus b_t^{84} \oplus a_t^{69} \oplus b_t^{82}.b_t^{83}$ (1)

$b_{t+1}^1 \leftarrow a_t^{66} \oplus a_t^{93} \oplus b_t^{78} \oplus a_t^{91}.a_t^{92}$ (2)

for i=92:-1:1

$a_{t+1}^{i+1} = a_t^i$

end

for i=83:-1:1

$b_{t+1}^{i+1} = b_t^i$

end

$z_t \leftarrow a_t^{66} \oplus a_t^{93} \oplus b_t^{69} \oplus b_t^{84}$ (3)

Fig. 1. Bivium Keystream Generation Algorithm

pher is updated according to the pseudo-code shown in Fig. 1.

2.2 Initialization

In the initialization phase, the cipher is clocked 4×177 times without generating any output. This process can be summarized by the pseudo-code shown in Fig. 2.

The key generation process is illustrated in Fig. 3 [5], where Bivium is illustrated as a truncated version of the Trivium cipher.

Initialization Algorithm

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots, 0)$

for t = 1 to 4 × 177

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

$t_2 \leftarrow s_{69} + s_{162} + s_{175}.s_{176} + s_{177}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_2, s_1, \ldots, s_{92})$

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

end for

Fig. 2. Initialization Algorithm

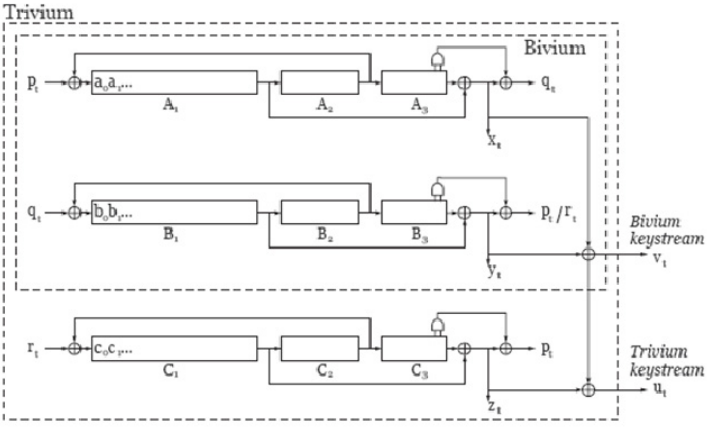

Fig. 3. Bivium and Trivium stream ciphers [5]

3. GUESS AND DETERMINE ATTACK

In this section, the guess and determine attack on Bivium is described. Guess and determine attack [6] is a class of structural attacks which have been applied to some stream cipher algorithms such as Sober [7], Snow [8] and Polar Bear [9]. This attack consists of several steps. Firstly, the attacker should guess a part of the internal state. Next, the remaining part of the state is determined according to the guessed values and the relations of the cipher. Finally, by running the cipher which is loaded by the determined state, the produced sequence is compared with the observed one. The equivalences between the two sequences mean that the guessed values are correct; otherwise the adversary should guess other values and repeat the steps until the two sequences become equal.

In the following, a guess and determine attack on Bivium is introduced. The idea mentioned in [10] is used as a key point in the proposed method.

3.1 Guess and Determine Attack on Bivium

According to the updating functions (1) and (2), after each time the cipher is clocked, two new variables a_{t+1}^1 and b_{t+1}^1 are produced. Moreover, two nonlinear equations (1) and (2) and one linear equation (3) are generated.

As it was mentioned before, the internal state has 177 bits. So if the cipher is clocked for k times we will have a system of k linear and 2×k nonlinear equations with 177+2×k unknowns. If the output function (3) is considered, it can be concluded that if the cipher is clocked 66 more times, there will be 66 extra linear equations related to the output functions which add no more unknowns. In other words, these 66 equations are linear relations of the 177 internal state bits. So there will be k+66 linear equations.

According to the equations (1) and (2), the updating relations are nonlinear with a degree of 2. As the truth table in Fig. 4 verifies, the multiplication of two bits can be substituted with zero, with a probability of ¾.

Therefore reducing the nonlinear updating relations to:

$$a_{t+1}^1 = b_t^{69} + b_t^{84} + a_t^{69} \tag{4}$$

$$b_{t+1}^1 = a_t^{66} + a_t^{93} + b_t^{78} \tag{5}$$

The entropy [11] of both equations (4) and (5) is equal to 0.811:

$$H(p) = -plogp - (1-p)log(1-p) = -0.75log0.75 - 0.25log0.25 \tag{6}$$

a	b	a&b
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 4. Truth Table for multiplication of two bits

So the relations (4) and (5) give 2×(1-H(p)) = 2×0.189 bits of information to the attacker.

If we replace the nonlinear terms with zero, the updating functions will become completely linear. So, the number of linear equations will be 3×k+66. Replacing the nonlinear terms with zero and using the information given by its entropy, 0.189×2k bits are determined. By subtracting the information obtained from linearization of the equations, the number of unknowns will be 177+2×k-0.189×2k. For solving the system, the number of equations should be greater than the number of the unknowns:

$$3k + 66 > 177 + 2k - 2 \times 0.189k \rightarrow k > 80.55 \tag{7}$$

In the proposed method, we assume that the updating functions are linear. As these equations are used k times, the probability that all these 2×k equations are linear, is:

$$p = 0.75^{2k}$$

In order to verify our assumptions, we should try 1/p bits of keystream sequence. So, the data complexity is equal to $O(1/p)$. Since the security level of Bivium is 80-bits, the complexity should be smaller than $O(2^{80})$.

$$2klog_2(4/3) < 80 \rightarrow k < 96.37 \tag{8}$$

If we compare the two bounds for k found in (7) and (8), it can be concluded that $80.55 < k < 96.37$. If we decrease the number of unknowns, data complexity will be reduced. If we guess n bits the number of unknowns will be 177+1.622×k-n. For solving the new system:

$$3k + 66 > 177 + 2k - 0.189 \times 2k - n \rightarrow k > 80.55 - n \tag{9}$$

Different numbers can be chosen for n. The number n defines the number of guessed bits. Since the security level of Bivium is 80-bits, n should be less than 80. The time complexity is equal to the complexity of guessed bits $O(2^n)$. If we choose n=30, k will be 50.55. According to (9), the value of n affects data complexity, and therefore, there is a tradeoff between values of k and n. By comparing previous attacks, which will be introduced briefly in the next section, we choose n=30 in order to propose a better attack.

Based on the value of n, the complexity of our attack is O (2^{30}). For applying our attack, we need O ($2^{41.96}$) bits of keystream. The computation complexity can be reduced by increasing data complexity. Since the base 2 logarithm of this complexity is less than the key length, our attack can be claimed to be a successful one.

4. RELATED WORKS

In [2] Borghoff, Knudsen and Stolpe proposed a new approach to solve the system of equations for internal state recovery of Bivium using combinatorial optimization with an estimated time complexity of $2^{64.5}$ seconds. In [12], Raddum proposed an algebraic attack on Bivium using Minisat for solving the system of nonlinear equations, with time complexity of 2^{56} seconds.

Table 1. Attacks on Bivium

Analyzers	Type of Attack	Complexity
Borghof,Knudse, Stolpe	State Recovery Attack	$O(2^{64.5})$
Raddum	Algebraic Attack	$O(2^{56})$
McDonald, Charnes, Pieprzyk	Guess and Determine Attack	$O(2^{52.3})$
Maximov, Biryukov	State Recovery Attack	$O(2^{51})$
Maximov, Biryukov	Distinguishing Attack	$O(2^{32})$
Noferesti, Rohani, Mohajeri, Aref	Distinguishing Attack	$O(2^{30.79})$
This Paper	Guess and Determine Attack	$O(2^{30})$

McDonald, Charnes and Pieprzyk [4] introduced a type of guess and determine attack on Bivium with a complexity of approximately O ($2^{52.3}$). Given a minimal amount of keystream, MiniSat determined the remaining unknown state, leading to complete key recovery. The authors convert the problem of solving a system of nonlinear equations over GF(2) into a corresponding SAT-problem. In [5], Maximov and Biryukov performed a state recovery attack on Bivium with a complexity of order O (2^{51}). They showed that Bivium's internal state can be recovered given the keystream. They also introduced a distinguishing attack with a complexity of order O (2^{32}) [5]. For applying their method, linear statistical methods are applied. In [13], Noferesti et al. performed a distinguishing attack with a complexity of order O ($2^{30.79}$). The current attack is a guess and determine attack with a complexity of O (2^{30}) which is the best among all. A summary is given in Table 1.

5. CONCLUSION

In this paper, we concentrated on a guess and determine attack applied to the Bivium stream cipher, which is a simplified version of Trivium, one of the hardware profile finalists of the eSTREAM project.

The attack is based on approximating the nonlinear update functions of the cipher with linear relations and solving a system of equations. The complexity of the attack is O (2^{30}). It seems replacing the updating functions with ones which have linear approximations with lower probabilities, may strengthen the algorithm against this attack.

REFERENCES

[1] eSTREAM: eSTREAM – The ECRYPT Stream Cipher Project: , http://www.ecrypt.eu.org/stream/

[2] J. Borghoff, L. R. Knudsen, M. Stolpe, *"Bivium as a Mixed-Integer Linear Programming Problem"*, Cryptography and Coding, Lecture Notes in Computer Science, Vol.5921, Springer, 2009, pp.133-152.

[3] C. De Canniere, B. Preneel, *"TRIVIUM – a stream cipher construction inspired by block cipher design principles"*, new stream cipher designs: the eSTREAM finalists, Lecture Notes in Computer Science, Vol.4986, Springer, 2008, pp.244-266.

[4] C. McDonald, C. Charnes, J. Pieprzyk, *"Attacking Bivium with MiniSat"*, Cryptology ePrint Archive, Report 2007/040, 2007.

[5] A. Maximov, A. Biryukov, *"Two Trivial Attacks on Trivium"*, Selected Areas in Cryptography, Lecture Notes in Computer Science, Vol.4876, Springer, 2007, pp.36-55.

[6] H. Ahmadi, T. Eghlidos, *"Heuristic Guess-and-Determine Attacks on Stream Ciphers"*, IET Journal in Information Security, Vol.3, 2009, pp.66-73.
[7] P. Hawkes, G. Rose, *The t-class of SOBER stream ciphers*, Technical report, QUALCOMM Australia, Suite 410, Birkenhead Point, DrummoyneNSW 2137, Australia, 1999.
[8] P. Hawkes, G. Rose, *"Guess and Determine Attacks on SNOW"*, Selected Areas in Cryptography, Lecture Notes in Computer Science, vol. 2595, Springer, 2002, pp.37-46.
[9] J. Hastad, M. Naslund, *"The Stream Cipher Polar Bear"*, eSTREAM, ECRYPT Stream Cipher Project Report 2005/021, 2005, http://www.ecrypt.eu.org/stream/, accessed June, 2008.
[10] *S. Babbage, "Some Thoughts on Trivium"*, eSTREAM, ECRYPT Stream Cipher. Project, Report 2007/007 (2007),http://www.ecrypt.eu.org/stream.
[11] D. *Denning, Cryptography and Data Security*, Addison-Wesley, May, 1982.
[12] H.Raddum, *"Cryptanalytic Results on Trivium"*, eSTREAM, ECRYPT Stream Cipher Project, Report 2006/039, 2006. http://www.ecrypt.eu.org/stream.
[13] Z. Noferesti, N. Rohani, J. Mohajeri, M. Aref, *"Distinguishing Attack on Bivium", 10th IEEE International Conference on Computer and Information Technology*, UK, 2010, pp.1075-1078.

Neda Rohani

She received a B. S. degree in electrical engineering, from Sharif University of Technology, in 2008 and an M.S. degree in secure communications engineering from Sharif University of Technology, in 2010. She is currently a researcher at the Information System and Security Lab of Sharif University of technology. Her fields of interest include symmetric cryptography, Bioinformatics and wireless security.

Zainab Noferesti

She received a B. S. degree in electrical engineering, from Sharif University of Technology, in 2007 and an M.S. degree in secure communications engineering from Sharif University of Technology, in 2010. She is currently a researcher at the Information System and Security Lab of Sharif University of technology. Her fields of interest include symmetric cryptography, data networks and wireless communication.

Javad Mohajeri

He received a B. S. degree from the Isfahan University in 1986 and an M. S. degree from Sharif University of Technology in 1989, both in mathematics. Since 1990 he has been a faculty member at Electronics Research Center of Sharif University of Technology. His research interests include cryptography and data security. He is author/co-author of over 50 research articles in refereed Journals/ Conferences. He is one of the founding members of the Iranian Society of Cryptology.

Mohammad Reza Aref

He received a B.S. degree in electronics engineering from Tehran University, and an M.S. degree and Ph.D. degree in electrical and communication engineering from Stanford University in 1976 and 1980 respectively. He was a faculty member of the Isfahan University of Technology from 1982 to 1995; He has been a Professor of Electrical Engineering at Sharif University of Technology since 1995 and has published more than 160 technical papers in Communication and Information Theory and Cryptography in international journals and conference proceedings.