COZMO - A New Lightweight Stream Cipher

Krishnendu Rarhi, Rhea Bonnerji, Simanta Sarkar and Pratyusa Mukherjee

April 24, 2018

COZMO - A New Lightweight Stream Cipher

Rhea Bonnerji[0000-0002-5825-8800], Simanta Sarkar [0000-0002-4210-2764],
Krishnendu Rarhi[0000-0002-5794-215X], Abhishek Bhattacharya

School of Information Technology, Institute of Engineering &
Management, Kolkata

Abstract. This paper deals with the merger of the two lightweight stream ciphers – A5/1 and Trivium. The idea is to make the key stream generation more secure and to remove the attacks of the individual algorithms. The bits generated by the Trivium cipher (output) will act as the input of the A5/1 cipher. The output bits of the Trivium cipher will fill the registers used in the A5/1 cipher. The three registers will then be connected to generate an output, which will be our required key stream.

Keywords: Lightweight stream cipher, A5/1, Trivium

1 Introduction

Cryptography is very important in today's times, as we want confidential information to remain confidential. We do not want people other than the intended receivers to be able to access our information and if accessed to be able to be able to understand it. Cryptography is the method of converting information to a non-understandable form and then being able to convert it to the understandable form by the recipient. Lightweight stream ciphers are used to reach high levels of security using only a small computing power. Stream encryption is the encryption of each letter one by one followed by the changing the encryption key after each letter. Lightweight stream ciphers have the advantage of having low cost hardware implementations as they have high throughput and low complexity. The idea is to come up with a lightweight stream cipher algorithm using existing cipher algorithms that is secure to the attacks of the originally used cipher algorithms. Also, to generate a more secure keystream. Here, we are using Trivium and A5/1 algorithm and making changes to suit our needs. This paper has results of statistical tests and proves that our algorithm can be considered strong.

1.1 Understanding the working of Trivium:

Trivium generates up to 2^{64} bits of key stream from an 80-bit secret key and an 80-bit initialization vector (IV). First, the internal state of the cipher is initialised using the key and the IV. Then the state is updated repeatedly to generate the key stream bits. It consists of 3 interconnected non-linear feedback shift registers. The length of the registers are 93, 84 and 111 bits, respectively. Initialisation requires 1152 steps of the clocking before key stream is generated. There are XOR gates to XOR the outputs we get from the various registers which is again fed back to the first one. The 91 and 92 bits are ANDed and used to give the feedback to the 94th bit. The 175 and 176 bits are ANDed and used to give the feedback to the 178th bit. The

286 and 287 bits are ANDed and used to give feedback to the 288th bit. In addition, the 288th bit is giving back the feedback to the first bit. [1]

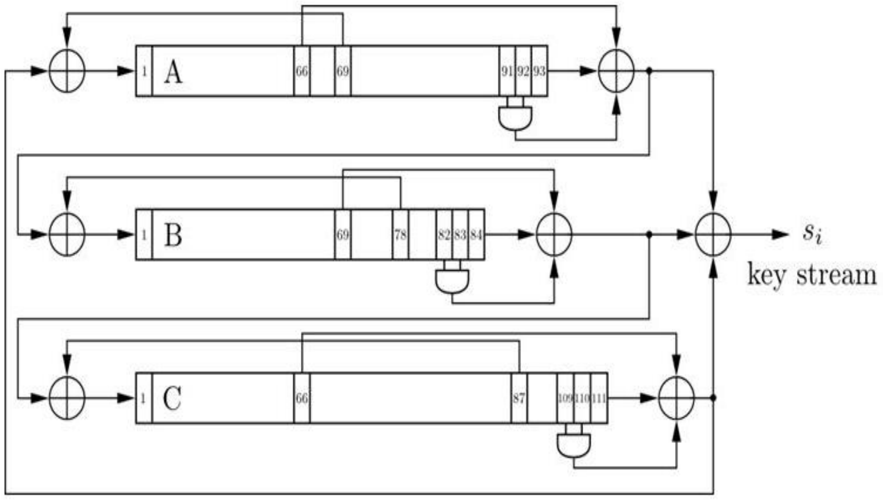

Fig. 1. Trivium

Mathematical structure

If we denote the internal state, bits of the Trivium Model algorithm at time t as

z(t)=(z1(t),z2(t), ... ,$z3_{n3}$ (t)), then the internal bits from time t to time t+ 1 can be expressed as follows

$$z(t + 1) = A \, . \, z(t) + b(t) \qquad (1)$$

Where A = $(a_{ij})3_{n3 \text{ x }} 3_{n3}$ is the state-transition matrix of the algorithm with size 3_{n3} x 3_{n3}

$$a_{ij} = \begin{cases} 1, & i = 1, j = 3u_2, 3u_5, 3n_2 \\ 1, & i = 3n_1 + 1, j = 3u_1, 3u_4 \\ 1, & i = 3n_2 + 1, j = 3u_3, 3u_6 \\ 1, & 1 < i <= 3n_3, j = I - 1 \\ 0, & \text{otherwise} \end{cases} \qquad (2)$$

b(t)=$(b_i(t))3_{n3}$ is the nonlinear segment of the algorithm, which is treated as the vectors of bits

$$b_i(t) = \begin{cases} z3_{n3-2}(t) \, . \, z3_{n3-1}(t), & i = 1 \\ z3_{n1-2}(t) \, . \, z3_{n1-1}(t), & i = 3n_1 + 1 \\ z3_{n2-2}(t) \, . \, z3_{n2-1}(t), & i = 3n_2 + 1 \\ 0, & \text{otherwise} \end{cases} \qquad (3)$$

1.2 Understanding the working of A5/1:

A5/1 uses 3 shift registers of 19, 22 and 23 bits respectively. We have a 64-bit key and we will load it in the three registers. Then we'll do some process to generate as many bits as we want to and we'll use those bits as the key stream and XOR it with the plain text to encrypt and the cipher text to decrypt. There is a majority vote function, which will take 3 bits and find the majority of them. The three shift registers are loaded with the 64-bit key. There are three special positions in the registers – 8, 10 and 10 respectively. Take the majority of these three bits and check which of the registers are in majority. If register 1 is in majority then take the 13th, 16th, 17th and 18th bit and XOR them. If register 2 is in majority then XOR the bits in the 20th and 21st position. If register 3 is in majority then XOR the bits in the 7th, 20th, 21st and 22nd position. Only shift the XORed bits to the first position of the registers belonging in majority. The register, which is not in majority, will remain untouched. Then we have to XOR the bits in the last position of the registers and that is our required key stream bit. Repeating this process repeatedly will give us our key stream. [2]

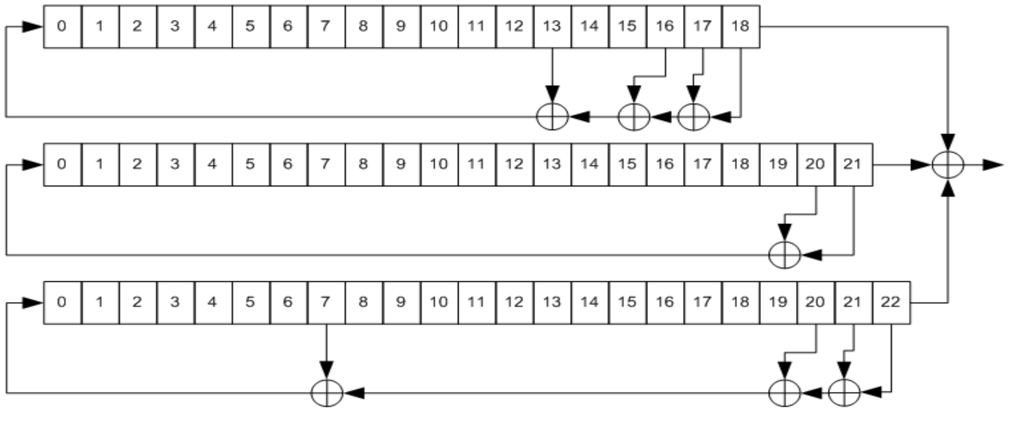

Fig. 2. A5/1

2 Proposed Methodology

The idea is to make the key stream even more secure by merging the two algorithms together. The Trivium cipher is initialised with a key and initialization vector and the A5/1 cipher is initialised with all 0's. First, we will be generating a key stream using the Trivium algorithm. The generated key stream bits will be used as the input for the A5/1 cipher. There will be a clocking of 1216 (1152+64) cycles before the key stream starts being generated. Let the three registers be called register A, B and C respectively. First we XOR the selected position bits in all the registers. For the first register A, the selected positions are the 13th, 16th, 17th, and 18th bits. For the second register B, the selected positions are the 20th and 21st bits. For the third register C, the selected positions are the 7th, 20th, 21st and 22nd bits. Next, we check the majority function and find the two registers that are in majority. Next, we shift the bit in the last position to the first position by using right shift by one place only for the registers in majority. Once that is done, we will use the XOR bits. If register A is in majority and B is not then discard the XORed bit from register A. Since register C is in majority, the XORed bit of register B will replace the bit in the first position of register C. Register A that is also in majority will have its first bit replaced by the XORed bit of the register C that is XORed with the bit generated by Trivium. The final key stream is the XOR of the three bits in the last positions of the three registers.

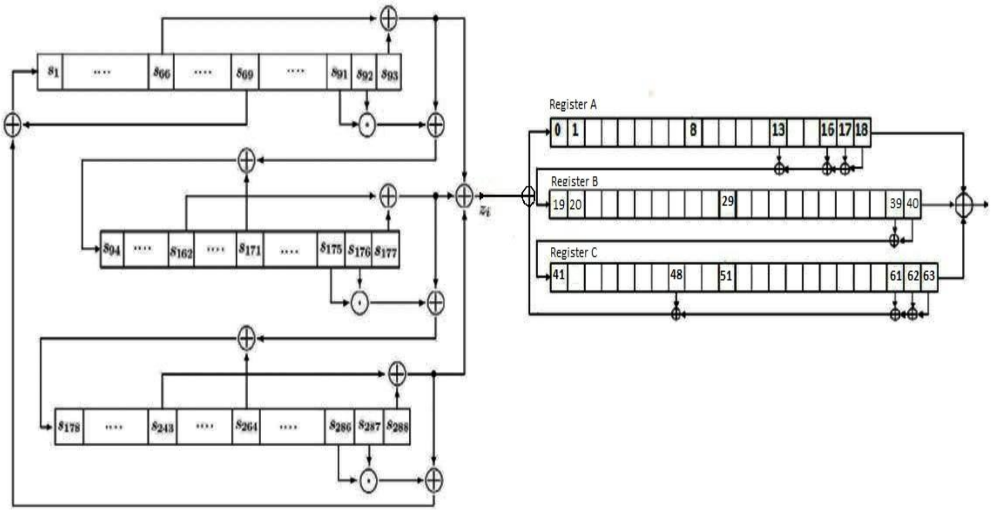

Fig. 3. COZMO

Pseudo Code:

```
Key = (K1, . . . , K80)
IV = (IV1, . . . , IV80)
(s1, s2, . . . , s93) ← (K1, . . . , K80, 0, . . . , 0)
(s94, s95, . . . , s177) ← (IV1, . . . , IV80, 0, . . . ,
0)
(s178, s279, . . . , s288) ← (0, . . . , 0, 1, 1, 1)
Register A:
    (r0, r2, . . . , r18) ← (0,0, . . . , 0)
Register B:
    (r19, r20, . . . , r40) ← (0,0, . . . , 0)
Register C:
        (r41, r43, . . . , r63) ← (0,0, . . . , 0)
```

Operations:

⊕ : bit-wise exclusive OR

& : bit-wise AND

Variables:

z_i : the i^{th} bit generated by trivium.

l : r_8 bit of register A

M : r_{29} bit of register B

N : r_{51} bit of register C

t_i : The keystream bit generated at the i^{th} step.

Function:

maj(L, M, N) = (L&M) ⊕ (M&N) ⊕ (L&N)

A complete description is given by the following pseudo-code:

```
For i = 1 to N do
        $t_i \leftarrow r_{18} \oplus r_{40} \oplus r_{63}$
        p1 ← $r_{13} \oplus r_{16} \oplus r_{17} \oplus r_{18}$
        p2 ← $r_{39} \oplus r_{40}$
        p3 ← $r_{48} \oplus r_{61} \oplus r_{62} \oplus r_{63} \oplus z_i$
        If $s_8$ equal to maj(L, M, N) Then
                (r0,r1, r2, . . . , r18) ← (p3, r0, . . . , r17)
        ElseIf s29 equal to maj(L, M, N) Then
                (r19,r20, . . . , r40) ← (p1, r19, . . . , r39)
        ElseIf $s_{52}$ equal to maj(L, M, N) Then
                (r41,r42, . . . , r63) ← (p2, r41, . . . , r62)
        EndIf
End for
```

Attacks:

Correlation attack:

This attack is usually used on ciphers whose output is a combination of LFSRs and Boolean functions. This is possible when there is correlation between bit generated by the combination of bits used in the Boolean function and the output bit of the LFSRs. Our algorithm, however, has been tested and is secure from the correlation attack.

3 Results and Discussion

As we know statistical analysis is a very important analysis for determining the acceptance or rejection of the hypothesis, the following tests were conducted to test the randomness property of the key bit generator. We performed basic statistical tests like frequency, longest run of one's in a block, linear complexity, entropy, etc. on our hypothesis. The result of the tests are shown in the tables below.

Statistical Test	p- value	Success/failure
Frequency	0.534146	Success
Cumulative Sums	P1-0.122325 P2-0.350485	Success
Approximate Entropy	0.004301	Success
Linear Complexity	0.213309	Success
Serial	P1-0.004301 P2- 0.017912	Success
Longest Run of Ones	0.350485	Success
Runs	0.911413	Success

Table 1: The Results of Statistical Analysis for A5/1

Statistical Test	p- value	Success/failure
Frequency	0.534146	Success
Cumulative Sums	P1- 0.213309 P2- 0.350485	Success
Approximate Entropy	0.534146	Success
Linear Complexity	0.035174	Success
Serial	P1- 0.911413 P2- 0.534146	Success
Longest Run of Ones	0.739918	Success
Runs	0.350485	Success

Table 2: The Results of Statistical Analysis for Trivium

Statistical Test	p- value	Success/failure
Frequency	0.911413	Success
Cumulative Sums	P1-0.991468 P2-0.739918	Success
Approximate Entropy	0.350485	Success
Linear Complexity	0.066882	Success

Serial	P1-0.213309 P2-0.534146	Success
Longest Run of Ones	0.534146	Success
Runs	0.122325	Success

Table 3: The Results of Statistical Analysis for COZMO

4 Conclusion

We wanted to propose a new lightweight stream cipher because of their higher speed, efficiency and its easy implementations on applications, which require plain texts of unknown length. In this paper, we merged and made some changes to the algorithms of two already existing lightweight stream ciphers – Trivium and A5/1 and proposed a new lightweight stream cipher with good key randomness as key randomness is a major issue is stream ciphers. The generated key stream is more secure compared to the key streams generated by the individual ciphers. Trivium is already known to be a secure cipher. Any modifications made on it naturally increase the chances to the generated key stream to be more secure. Our algorithm also produces sequences, which are stronger in statistical properties. The implementation of this algorithm is also very feasible and easy to put to use.

5 References

1. C. De Canni`ere and B. Preneel, “Trivium – A Stream Cipher Construction Inspired by Block Cipher Design Principles,” 2006/021. (2006)
2. Timo Gendrullis, Martin Novotny, and Andy Rupp, “A Real-World Attack Breaking A5/1 within Hours” /2008/147, (2008).
3. Diffie, W. y M.E.Hellman. "New directions in cryptography", IEEE Transactions on Information Theory 22 (1976), pp. 644-654.
4. Raddum, H.“Cryptanalytic Results on Trivium”, eSTREAM, ECRYPT Stream Cipher Project, Report 2006/039, 2006.
5. De Canniere C and Preneel B. TRIVIUM specifications. eSTREAM, ECRYPT stream cipher project. Report no. 2005/030, April 2005
6. Fischer, W., Gammel, B.M., Kniffler, O., Velten, J.: Differential power analysis of stream ciphers. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 257–270. Springer, Heidelberg (2006)
7. Lennart Diedrich, Patrick Jattke, Lulzim Murati, Matthias Senker, and Alexander Wiesmaier, “Comparisons of Lightweight Stream Ciphers: MICLEY 2.0, WG-8, Grain and Trivium”
8. Maximov, A. and Biryukov, A. “Two Trivial Attacks on Trivium”, Selected Areas in Cryptography, Lecture Notes in Computer Science, Vol.4876, Springer, 2007.
9. Jia, Y., Yupu, H., Wang, F., Wang, H.: Correlation power analysis of Trivium. Secur. Commun. Netw. **5**(5), 479–484 (2012)
10. Biham E and Dunkelman O. Cryptanalysis of the A5/1 GSM stream cipher. In: Roy B and Okamoto E (eds) Progress in cryptology INDOCRYPT 2000. Berlin: Springer, 2000, pp.43–51.

11. Maximov, A., T. Johansson, and S. Babbage, An Improved Correlation Attack on A5/1, in Selected Areas in Cryptography, H. Handschuh and M. Hasan, Editors. 2005, Springer Berlin / Heidelberg. p. 1-18.
12. Meier, W. and O. Staffelbach, Nonlinearity Criteria for Cryptographic Functions, in Advances in Cryptology — EUROCRYPT '89, J.-J. Quisquater and J. Vandewalle, Editors. 1990, Springer Berlin / Heidelberg. p. 549-562.
13. Barkan, E., E. Biham, and N. Keller, Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. Journal of Cryptology, 2008. 21(3): p. 392-429.
14. Gendrullis, T., M. Novotný, and A. Rupp, A RealWorld Attack Breaking A5/1 within Hours, in Cryptographic Hardware and Embedded Systems – CHES 2008, E. Oswald and P. Rohatgi, Editors. 2008, Springer Berlin / Heidelberg. p. 266-282.
15. Barkan, E., E. Biham, and N. Keller, Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication, in Advances in Cryptology - CRYPTO 2003. 2003, Springer Berlin / Heidelberg. p. 600-616.
16. . J. Lano, N. Mentens, B. Preneel, and I. Verbauwhede, "Power Analysis of Synchronous Stream Ciphers with Resynchronization Mechanism," in ECRYPT Workshop, SASC – The State of the Art of Stream Ciphers, pp. 327–333, 2004.