On the Design of Trivium

Yun Tian, Gongliang Chen, Jianhua Li
School of Information Security Engineering,
Shanghai Jiaotong University, China
ruth_tian@sjtu.edu.cn, chengl@sjtu.edu.cn, lijh888@sjtu.edu.cn

Abstract. eSTREAM called for new stream ciphers designed for niche areas such as exceptional performance in software and hardware where resources are restricted. This project provides an open platform to discuss these ciphers. Trivium is one of the promising new ciphers submitted to it. Until now, no attack has been successfully applied to it. This paper illustrates new design principles of stream ciphers based on the structure of Trivium and introduces the definition of k-order primitive polynomials. New designs of Trivium are also given according to the principles in this paper.

Key words: eSTREAM, Trivium, design principles of stream ciphers

§1 Introduction

The ECRYPT Stream Cipher Project[1], abbreviated eSTREAM, is a multi-year effort to identify new stream ciphers potentially suitable for widespread adoption. In late 2004 eSTREAM announced a call for new stream cipher proposals and no less than 34 different stream cipher proposals were submitted in two performance profiles, software oriented and hardware oriented. Now the project has selected a portfolio of 7 promising new stream ciphers.

A stream cipher is a symmetric encryption algorithm which takes a stream of plaintext, a secret key and an IV as input and then operates the plaintext with key stream generated by the key and IV, typically bit by bit. The general framework of a stream cipher is shown in Figure 1.

The secret key and IV are of fixed length and used to initialize the state of key stream generator, the key part of a stream cipher. Now there are many strategies and ways to design a secure and efficient key stream generator, including LFSRs, FCSRs, NLFSRs, T-functions and so on. For many years, the stream ciphers used are kept secret and lack open discussion of security and attacks, which may lead to fatal failure when the algorithm is leaked. For example, misuse of RC4 in Wired Equivalent Privacy (WEP) protocol decreased the security of the protocol.

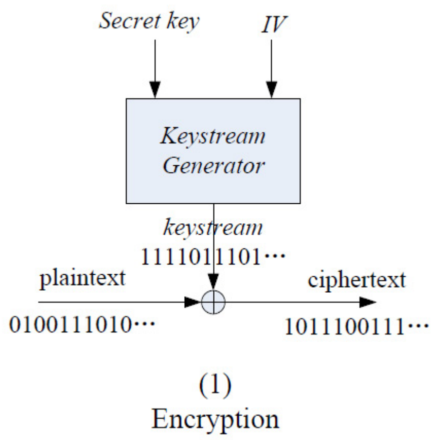


Figure 1 A Stream Cipher

Although block ciphers seem to be perfectly adequate for use in nearly all areas, stream ciphers are still desirable in a few niche areas, which is pointed out by Adi Shamir at the first ECRYPT State of the Art of Stream Ciphers workshop in October 2004[5]. These niche areas were identified as: 1) Exceptional encryption performance in software, where the luxury of additional hardware is not available to speed up encryption; 2) Any reasonable kind of encryption performance in hardware environments where the available resources such as gate count or power might be heavily restricted. The extreme example of this is provided by simple RFID tags. eSTREAM provides such a platform that scientists and researchers can exchange the design strategies of stream ciphers and find possible attacks on stream ciphers submitted to the project. Trivium[2] is one of the promising new stream ciphers in the hardware oriented profile. It has got high scores in evaluation due to its good performance and high security. Its simplicity and clarity perfectly demonstrate a new way to design secure stream ciphers. The idea of design of Trivium can be found in [3].

This paper focuses on the new design principles of stream ciphers and tries to find out better constructions and more application of Trivium. These principles are given based on the structure of Trivium. Section 2 describes the stream cipher Trivium and Section 3 discusses its security and efficiency. Section 4 introduces Trivium-model stream ciphers and k-order primitive polynomials in order to illustrate the new design principles. Section 5 shows new constructions of Trivium based on the principles in Section 4 and Section 6 draws the conclusion.

§1.1 Notation

$(s_m, \ldots, s_n)$	the internal state of $(m-n+1)$ bits
s_i	the i^{th} state or bit in the shift registers
z_t	the bit of keystream generated at time t
$+$	addition over $GF(2)$, i.e. XOR
$\cdot$	multiplication over $GF(2)$, i.e. AND
$\{a, b, c\}$	notation of one round in Trivium by its active bits: a^{th} bit, b^{th} bit and c^{th} bit

§2 Trivium

Trivium is designed to generate up to 2^{64} bits of key stream from an 80-bit secret key and an 80-bit initial value (IV). The process consists of two phases: first the internal state of the cipher is initialized using the key and the IV, then the state is repeatedly updated and used to generate key stream bits. There are 288 bits in the internal state.

A complete description of the generation keystream phase is given by the following simple pseudo-code:

for $i = 1$ to N do

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

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

$t_3 \leftarrow s_{243} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

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

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$

$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

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

$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

Figure 2 shows the structure of the algorithm.

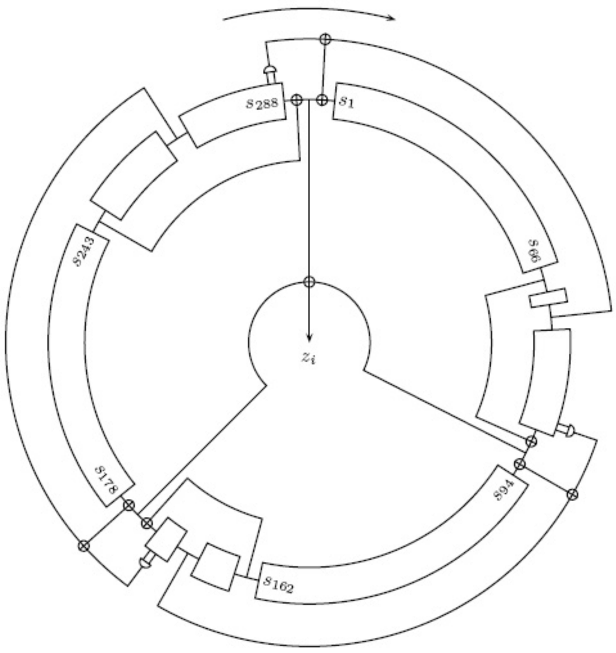


Figure 2 Structure of Trivium

The initialization phase operates exactly the same as the keystream generation phase except that it doesn't generate keystream. The state is rotated over 4 full cycles after the loading of key and IV. 4 full cycles means $4 * 288 = 1152$ clock cycles.

Key and IV are loaded as following:

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_3, \ldots, K_{80}, 0, \ldots, 0)$

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

$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$

§3 Security and Efficiency of Trivium

Trivium is designed to be both secure and efficient. The following sections discuss the two properties of Trivium.

§3.1 Security

[3] classifies attacks against stream ciphers into two categories as follows:

- **Key recovery attacks**, the goal of which is to recovery part or the whole key by observing the key stream.

- **Distinguishing attacks**, the goal of which is to detect that the key stream bits are not completely unpredictable.

It is obvious that the objective of distinguishing attacks is weaker than that of key recovery attacks. So it is easier to apply and harder to protect against. Design of Trivium focuses particularly on linear correlations and the main design objective is to keep the largest correlations below safe bounds. Other important properties, such as a sufficiently long period, are considered afterwards.

Until now, no attack has been successfully applied to Trivium. One attempt is to find out attacks on reduced version of Trivium and try to extend them to the original cipher. In [12], two attacks on Trivium are studied. These attacks are state recovering and statistical tests. Although the analysis applied to Bivium (a reduced version of Trivium from 3 rounds to 2 rounds) is quite successful, the results on Trivium are not good since the attacks are no faster than exhaustive search. [6] applies the SAT solver attack and the BDD attack on Bivium and optimizes them. The attacks are of good speed, but the extension of the attacks to Trivium was left as an open question. Another reduced version of Trivium, 2-round Trivium, is proposed in [15]. Matsui's linear cryptanalysis is applied to this 2-round Trivium and a linear approximation with bias 2^{-31} is given.

Another attempt is to exploit the initialization part of Trivium. Results of statistical analysis of Trivium can be found in [14], which include key/key stream correlation test and IV/key stream correlation test. [8] demonstrates a key recovery attack on reduced initialization version of Trivium by chosen IV statistical analysis. But evidence is given that the analysis is not applicable on Trivium with full IV initialization. [4] develops a new technique, called cube attack, which is a major improvement over several previously published attacks of the same type and applies it to Trivium with a reduced number of initialization round. The speed of this attack is faster than that in [8]. Evaluation on implementation of Trivium for low-power application in RFID system is given in [7] with comparison to AES-128. Synthesis result of Trivium is better than that of AES-128 except for the large amount of clock cycles needed in the initialization phase. [12] analyzes the completeness property of the initialization function and proposes a new input to the initialization of Trivium that has better diffusion properties.

§3.2 Efficiency

[3] gives a good measure, number of key stream bits generated per cycle per gate, for the efficiency of a stream cipher in hardware applications. In the later test and evaluation, hardware implementation of Trivium in FPGAs and ASICs shows that it is faster and more efficient than other candidates in eSTREAM and AES-CTR (see [9-11], [13]). For example, [9] points out that Trivium outperforms other eSTREAM candidates considered in the paper in terms of the two most important optimization criteria, minimum area

and maximum throughput to area ratio, by a factor of at least two.

§4 Design Principles of Trivium-model Stream Ciphers

Trivium is successfully designed to meet the need in the niche area where resources of hardware environment such as gate count or power might be heavily restricted. Besides efficiency, Trivium also resist all the known attacks. Based on the low linear correlations and high efficiency of Trivium, this paper tries to find out design principles of secure stream ciphers.

Let's focus on the operation of state and the pseudo-code is given bellow:

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

$t_2 \leftarrow s_{162} + s_{177} + s_{175} \cdot s_{176} + s_{264}$

$t_3 \leftarrow s_{243} + s_{288} + s_{286} \cdot s_{287} + s_{69}$

Index numbers of active bits in this part are rewritten in Table 1.

$3u_1$	$3u_2$	$3n_1$	$3u_3$	$3u_4$	$3n_2$	$3u_5$	$3u_6$	$3n_3$
$66 = 3 \cdot 22$	$69 = 3 \cdot 23$	$93 = 3 \cdot 31$	$162 = 3 \cdot 54$	$171 = 3 \cdot 57$	$177 = 3 \cdot 59$	$243 = 3 \cdot 81$	$264 = 3 \cdot 88$	$288 = 3 \cdot 96$

Table 1 Numbers of original Trvium

There are 93 bits in the first round of Trivium, $(177-93) = 84$ bits in the second round and $(288-177) = 111$ bits in the third round. Here, the first round is noted by its active bits as $\{66, 69, 93\}$. Thus , the second round can be written as $\{162, 171, 177\}$, and $\{243, 264, 288\}$ for the third round. It is not difficult to find that each number in Table 1 has a factor of 3.

§4.1 Description of Trivium-model Stream Ciphers

In order to discuss the design of Trivium, the algorithm can be generalized into a model. By changing the exact index numbers of active bits into variables, we can get a generalized algorithm of Trivium. These variables can be seen as the parameter of this Trivium-model stream cipher. Trivium-model stream cipher can be noted as

$$\{3u_1, 3u_2, 3n_1\},\ \{3u_3, 3u_4, 3n_2\},\ \{3u_5, 3u_6, 3n_3\},$$

where u_i $(i = 1, \cdots, 6)$, n_i $(i = 1, 2, 3)$ are parameters and $u_1 < u_2 < n_1 < u_3 < u_4 < n_2 < u_5 < u_6 < n_3$. n_1, n_2, n_3 also represent the degree of the corresponding characteristic polynomials. We will discuss more details about this in the following sections.

Pseudo-code of the original cipher is changed to a generalized form as following.

for $\quad i = 1$ to N do

$$
\begin{aligned}
&t_1 \leftarrow s_{3u_1} + s_{3n_1}\\
&t_2 \leftarrow s_{3u_3} + s_{3n_2}\\
&t_3 \leftarrow s_{3u_5} + s_{3n_3}\\[4pt]
&z_i \leftarrow t_1 + t_2 + t_3\\[4pt]
&t_1 \leftarrow t_1 + s_{3n_1-2} \cdot s_{3n_1-1} + s_{3u_4}\\
&t_2 \leftarrow t_2 + s_{3n_2-2} \cdot s_{3n_2-1} + s_{3u_6}\\
&t_3 \leftarrow t_3 + s_{3n_3-2} \cdot s_{3n_3-1} + s_{3u_2}\\[4pt]
&(s_1, s_2, \ldots, s_{3n_1}) \leftarrow (t_3, s_1, \ldots, s_{3n_1-1})\\
&(s_{3n_1+1}, s_{3n_1+2}, \ldots, s_{3n_2}) \leftarrow (t_1, s_{3n_1+1}, \ldots, s_{3n_2-1})\\
&(s_{3n_2+1}, s_{3n_2+2}, \ldots, s_{3n_3}) \leftarrow (t_2, s_{3n_2+1}, \ldots, s_{3n_3-1})
\end{aligned}
$$

This is the general model of Trivium. The structure of this model is given by Figure 3.

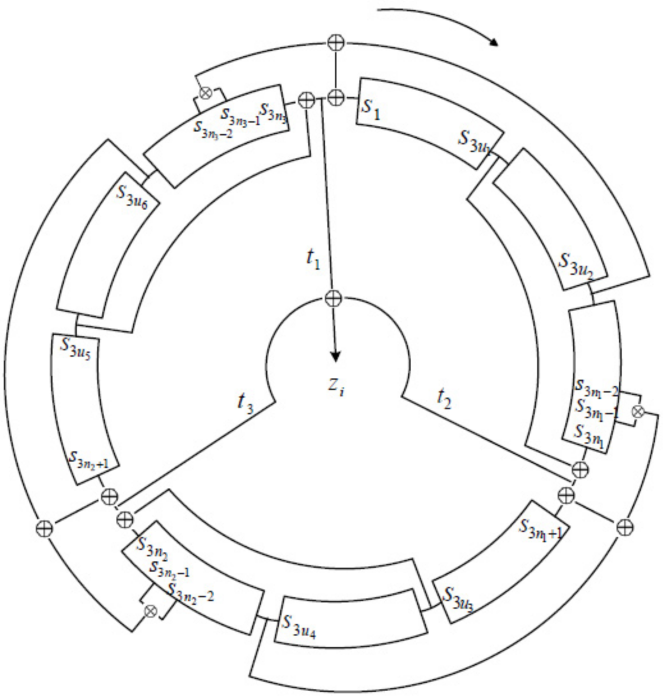


Figure 3 Structure of a Trivium-model stream cipher

A Trivium-model stream cipher consists of 3 rounds and each round operates nearly the same as the other two. In order to get design strategies of Trivium-model stream ciphers, the following discussion breaks the cipher into smaller parts, reconstructs these parts and gives an extension.

§4.1.1 Univium: A 1-round Trivium-model Stream Cipher

Univium is a 1-round Trivium-model stream cipher. It can be noted as

$$\{3u_1, 3u_2, 3n_1\},$$

where u_1, u_2, n_1 are parameters, and $u_1 < u_2 < n_1$. n_1 also denotes the degree of the characteristic polynomial of Univium.

Pseudo-code of this cipher is given below:

for $\quad i = 1$ to N do

$\quad t_1 \leftarrow s_{3u_1} + s_{3n_1}$

$\quad z_i \leftarrow t_1$

$\quad t_1 \leftarrow t_1 + s_{3n_1-2} \cdot s_{3n_1-1} + s_{3u_2}$

$\quad (s_1, s_2, \ldots, s_{3n_1}) \leftarrow (t_3, s_1, \ldots, s_{3n_1-1})$

Figure 4(a) shows the internal states of Univium. If the sequences of internal state are of uniform distribution, $Prob[s_{m_3-2} \cdot s_{m_3-1} = 0] = 0.75$. This means that the non-linear part of the algorithm can be neglected with a probability of 0.75. Figure 4(b) shows the internal states of Univium with non-linear part omitted and all the parameters divided by 3. Design principles discussed below are based on this model.

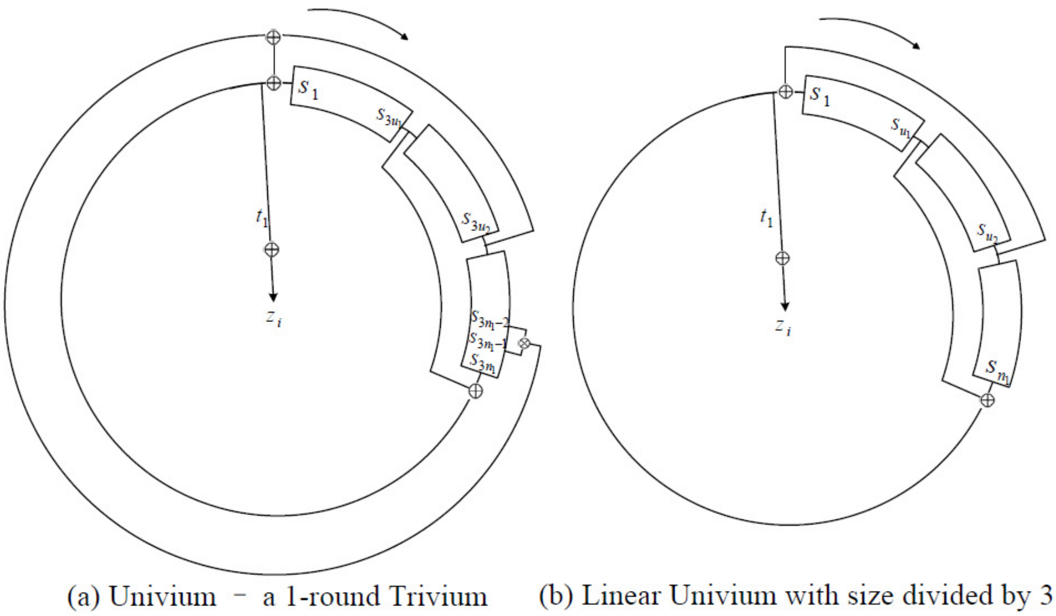


Figure 4 Univium

§4.1.2 Bivium: A 2-round Trivium-model Stream Cipher

Bivium is a 2-round Trivium-model stream cipher. It can be noted as

$$\{3u_1, 3u_2, 3n_1\},\ \{3u_3, 3u_4, 3n_2\},$$

where u_i $(i = 1, \cdots, 4)$, n_1, n_2 are parameters and $u_1 < u_2 < n_1 < u_3 < u_4 < n_2$. n_2 also denotes the degree of the characteristic polynomial of Bivium.

Pseudo-code of this cipher is given below:

for $\quad i = 1$ to N do

$$t_1 \leftarrow s_{3u_1} + s_{3n_1}$$
$$t_2 \leftarrow s_{3u_3} + s_{3n_2}$$
$$z_i \leftarrow t_1 + t_2$$
$$t_1 \leftarrow t_1 + s_{3n_1-2} \cdot s_{3n_1-1} + s_{3u_4}$$
$$t_2 \leftarrow t_2 + s_{3n_2-2} \cdot s_{3n_2-1} + s_{3u_2}$$
$$(s_1, s_2, \ldots, s_{3n_1}) \leftarrow (t_2, s_1, \ldots, s_{3n_1-1})$$
$$(s_{3n_1+1}, s_{3n_1+2}, \ldots, s_{3n_2}) \leftarrow (t_1, s_{3n_1+1}, \ldots, s_{3n_2-1})$$

Figure 5(a) shows the internal states of Bivium. If the sequences of internal state are of uniform distribution, the non-linear part of the algorithm can be neglected with a probability of 0.75^2. Figure 5(b) shows the internal states of Bivium with non-linear part omitted and all the parameters divided by 3. Design principles discussed below are based on this model.

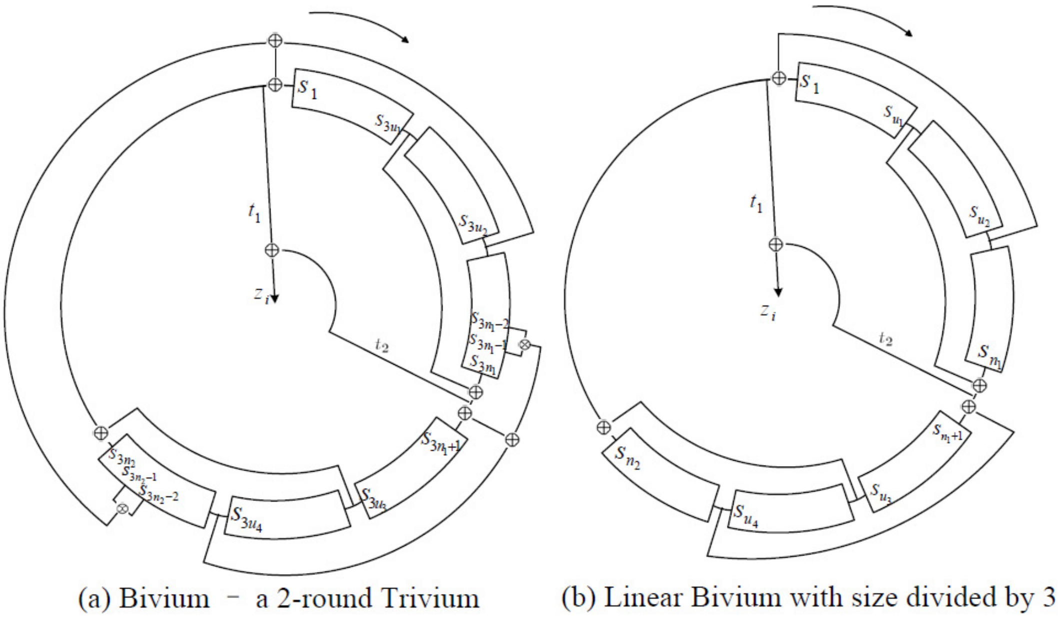


Figure 5 Bivium

§4.1.3 Trivium: A 3-round Trivium-model Stream Cipher

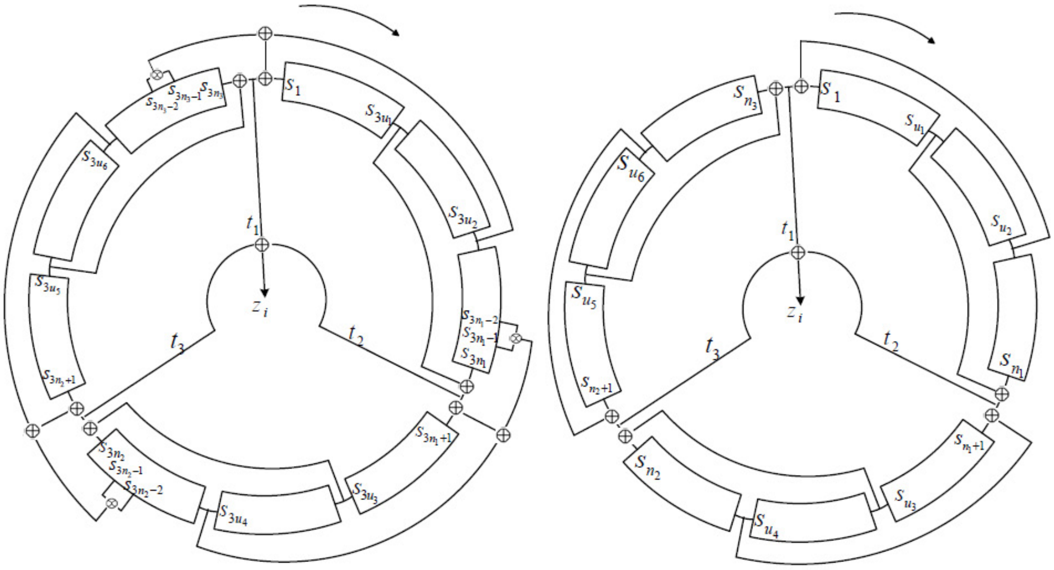
Here, we still use 'Trivium' to denote a 3-round Trivium-model stream cipher. A 3-round Trivium-model stream cipher can be noted as

$$\{3u_1, 3u_2, 3n_1\},\ \{3u_3, 3u_4, 3n_2\},\ \{3u_5, 3u_6, 3n_3\},$$

where u_i $(i = 1, \ldots, 6)$, n_i $(i = 1, 2, 3)$ are parameters and $u_1 < u_2 < n_1 < u_3 < u_4 < n_2 < u_5 < u_6 < n_3$. n_3 also denotes the degree of the characteristic polynomial of Trivium.

Pseudo-code of this cipher is the same as that in Section 4.1.

Figure 6(a) shows the internal states of Trivium. If the sequences of internal state are of uniform distribution, the non-linear part of the algorithm can be neglected with a probability of 0.75^3. Figure 6(b) shows the internal states of Trivium with non-linear part omitted and all the parameters divided by 3. Design principles discussed below are based on this model.



(a) Trivium – Generalized Trivium (b) Linear Trivium with size divided by 3

Figure 6 Trivium

§4.1.4 Extension: k-round Trivium-model Stream Ciphers

After the description of Univium, Bivium and Trivium, it is not difficult to extend the Trivium-model stream cipher to arbitrary rounds. Each round still contains feedback part, feed forward part and non-linear part.

§4.2 k-order Primitive Polynomial

In order to discuss criterion to determine the parameters of the stream ciphers described in the above sections, a new definition, k-order primitive polynomial, is introduced here. First, we would like to review the definition of primitive polynomial.

Definition 4.2.1 A polynomial $f(x)$ with coefficients in $GF(p) = Z/pZ$ is a *primitive polynomial* if it has a root α in $GF(p^m)$ such tha t $\{0, 1, \alpha, \alpha^2, \alpha^3, \cdots, \alpha^{p^m-2}\}$ is the entire field $GF(p^m)$, and moreover, $f(x)$ is the smallest degree polynomial having α as root.

Based on definition 4.2.1 and let $p = 2$, an extensive definition, k-order primitive polynomials, is given below.

Definition 4.2.2 Given $f(x) = \sum_{i=0}^{n} a_i x^i$, $n > k$, $a_i \in GF(2)$, $i = 0, 1, \cdots, n$, $f(x)$ is called a *k-order primitive polynomial* if $f(x) = (x+1)^k \cdot g(x)$, where $g(x)$ is a primitive polynomial.

Remark Definition 4.2.2 is an extension of definition 4.2.1 because 0-order primitive polynomial is a primitive polynomial by definition 4.2.1.

Example 4.2.1 $f_1(x) = x^{28} + x^5 + x^2 + 1 = (x+1)(x^{27} + x^{26} + x^{25} + x^{24} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x + 1)$ is a 1-order primitive polynomial.

Example 4.2.2 $f_2(x) = x^{31} + x^9 + x^8 + 1 = (x+1)^2(x^{11} + x^{10} + x^6 + x^5 + x^3 + x + 1)(x^7 + x + 1)(x^{11} + x^{10} + x^7 + x^3 + 1)$ is not a 1-order primitive polynomial.

Example 4.2.3 $f_3(x) = x^{96} + x^{73} + x^{70} + x^{67} + x^{47} + x^{44} + x^{41} + x^{29} + x^{24} + x^{20} + x^{18} + x^{15} + x^{14} + x^9 + x^5 + 1 = (x+1)^3(x^{93} + x^{92} + x^{89} + x^{88} + x^{85} + x^{84} + x^{81} + x^{80} + x^{77} + x^{76} + x^{73} + x^{72} + x^{70} + x^{68} + x^{67} + x^{44} + x^{43} + x^{41} + x^{39} + x^{38} + x^{35} + x^{34} + x^{31} + x^{30} + x^{27} + x^{25} + x^{23} + x^{20} + x^{19} + x^{17} + x^{14} + x^{13} + x^{12} + x^9 + x^8 + x^6 + x^4 + x + 1)$ is a 3-order primitive polynomial.

§4.3 Design Principles of Trivium-model Stream Ciphers

Design principles introduced in this section mainly relates to the linear part of Trivium-model stream ciphers. The objective of these principles is to determine parameters in the algorithms described above, i.e. Univium, Bivium, Trivium and so on. k-order primitive polynomials are used to illustrate the principles.

If we want to construct a Trivium-model stream cipher, we can first determine the rounds of the cipher. Here, we suppose that we still need a 3-round cipher, i.e. Trivium. It is noted as

$$\{3u_1, 3u_2, 3n_1\},\ \{3u_3, 3u_4, 3n_2\},\ \{3u_5, 3u_6, 3n_3\},$$

where u_i $(i = 1, \ldots, 6)$, n_i $(i = 1, 2, 3)$ are parameters and $u_1 < u_2 < n_1 < u_3 < u_4 < n_2 < u_5 < u_6 < n_3$.

Next, we determine the unknown parameters under the following principles.

Principles Construct a 3-round Trivium, denoted by $\{3u_1, 3u_2, 3n_1\}, \{3u_3, 3u_4, 3n_2\}, \{3u_5, 3u_6, 3n_3\}$, such that

1) The characteristic polynomial of $\{u_1, u_2, n_1\}$ is a 1-order primitive polynomial;

2) The characteristic polynomial of $\{u_1, u_2, n_1\},\ \{u_3, u_4, n_2\}$ is a 2-order primitive polynomial;

3) The characteristic polynomial of $\{u_1, u_2, n_1\},\ \{u_3, u_4, n_2\},\ \{u_5, u_6, n_3\}$ is a 3-order primitive polynomial.

According to the principles, we can omit the non-linear parts (AND operations) and use Figure 4(b), 5(b), 6(b) to determine parameters u_i $(i = 1, \ldots, 6)$ and n_i $(i = 1, 2, 3)$.

If we want to construct a k-round Trivium-model stream cipher, then the characteristic polynomial of the i^{th}-round $(i = 1, 2, \cdots, k)$ of the linear part should be an i-order primitive polynomial.

§5 Improvement on Trivium

Based on the design principles given above, we would like to give better parameters than those of the original Trivium. In this paper, we only consider the condition under which the characteristic polynomials are calculated from the state S_1, i.e. the first internal bit. Setting different internal state bit as starting point may lead to different polynomials and results.

We choose the parameters divided by 3 according to the description in Section 4. Trivium has 288-bit internal state, so we set $n_3 = 288/3 = 96$. It can be noted as

$$\{3 \times 22, 3 \times 23, 3 \times 31\},\ \{3 \times 54, 3 \times 57, 3 \times 59\},\ \{3 \times 81, 3 \times 88, 3 \times 96\}.$$

We found that the characteristic polynomial of the original Trivium is 3-order primitive, but the characteristic polynomial of the corresponding Univium is not 1-order primitive. Neither is that of the corresponding Bivium.

After lots of research, we found out some suitable parameters according to the discussion above. Sections below describes new constructions of Trivium-model stream ciphers.

§5.1 Improvement On the Original Trivium

This algorithm is the improvement on original Trivium. n_1, n_2, n_3 are the same as the original algorithm and u_i $(i = 1, \cdots, 6)$ are adjusted so as to satisfy the principles discussed in Section 4.3. Thus, not only the characteristic polynomial of the whole algorithm is a 3-order primitive polynomial, but also those of the Univium and Bivium are 1-order and 2-order primitive polynomials respectively. This improved construction is noted as

$$\{3 \times 10, 3 \times 22, 3 \times 31\},\ \{3 \times 36, 3 \times 48, 3 \times 59\},\ \{3 \times 65, 3 \times 85, 3 \times 96\}.$$

The complete description of the generation keystream phase of this algorithm is given by the following simple pseudo-code:

for $i = 1$ to N do

$t_1 \leftarrow s_{30} + s_{93}$
$t_2 \leftarrow s_{108} + s_{177}$
$t_3 \leftarrow s_{195} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{144}$
$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{255}$
$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{66}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

§5.2 A New Trivium of 384 Bits

Based on the principles in Section 4.3, we can also construct a new Trivium-model stream cipher with internal states of 384 bits. This Trivium-model stream cipher can be noted as

$$\{3 \times 10, 3 \times 22, 3 \times 31\},\ \{3 \times 36, 3 \times 48, 3 \times 59\},\ \{3 \times 65, 3 \times 72, 3 \times 128\}.$$

The characteristic polynomial of $\{10, 22, 31\}$ is a 1-order primitive polynomial. That of $\{10, 22, 31\}$, $\{36, 48, 59\}$ is a 2-order primitive polynomial. That of $\{10, 22, 31\}$, $\{36, 48, 59\}$, $\{65, 72, 128\}$ is a 3-order primitive polynomial.

The complete description of the generation keystream phase of this algorithm is given by the following simple pseudo-code:

for $\quad i = 1$ to N do

$t_1 \leftarrow s_{30} + s_{93}$

$t_2 \leftarrow s_{108} + s_{177}$

$t_3 \leftarrow s_{195} + s_{384}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{144}$

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{216}$

$t_3 \leftarrow t_3 + s_{382} \cdot s_{383} + s_{66}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

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

$(s_{178}, s_{179}, \ldots, s_{384}) \leftarrow (t_2, s_{178}, \ldots, s_{383})$

§5.3 Another New Trivium Designed for 32-bit OS

Considering 32-bit OS, we also found another construction of Trivium listed below.

$$\{3 \times 5, 3 \times 20, 3 \times 32\},\ \{3 \times 33, 3 \times 42, 3 \times 64\},\ \{3 \times 65, 3 \times 84, 3 \times 96\}.$$

The complete description of the generation keystream phase of this algorithm is given by the following simple pseudo-code:

for $\quad i = 1$ to N do

$t_1 \leftarrow s_{15} + s_{96}$

$t_2 \leftarrow s_{99} + s_{192}$

$t_3 \leftarrow s_{195} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{94} \cdot s_{95} + s_{126}$

$t_2 \leftarrow t_2 + s_{190} \cdot s_{191} + s_{252}$

$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{60}$

$(s_1, s_2, \ldots, s_{96}) \leftarrow (t_3, s_1, \ldots, s_{95})$

$(s_{97}, s_{98}, \ldots, s_{192}) \leftarrow (t_1, s_{97}, \ldots, s_{191})$

$(s_{193}, s_{194}, \ldots, s_{288}) \leftarrow (t_2, s_{193}, \ldots, s_{287})$

§6 Conclusion

Trivium is one of the most promising stream ciphers selected by eSTREAM project. Simplicity, efficiency and clarity make it a successful design. Based on the structure of Trivium, this paper introduced new principles to design Trivium-model stream ciphers. An extended definition of primitive polynomial, k-order primitive polynomial, is given in this paper. New versions of Trivium according to the principles are shown as an improvement to the original algorithm.

References

[1] http://www.ecrytp.eu.org/stream.

[2] C. De Cannière & B. Preneel. TRIVIUM Specifications. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/001, 2005. http://www.ecrypt.eu.org/stream.

[3] C. De Cannière & B. Preneel. TRIVIUM A Stream Cipher Construction Inspired by Block Cipher Design Principles. In Workshop on *Stream Ciphers Revisited (SASC2006)*, 2006.

[4] I. Dinur & A. Shamir. Cube Attacks on Tweakable Black Box Polynomials. http://eprint.icar.org/2008/385.

[5] ECRYPT Network of Excellence in Cryptology. In Workshop on *The State of the Art of Stream Ciphers (SASC2004)*, 2004.

[6] T. Eibach, E. Pilz & S. Steck. Comparing and Optimising Two Generic Attacks on Bivium. In Workshop on *The State of the Art of Stream Ciphers (SASC2008)* pages 57-68, 2008.

[7] M. Feldhofer. Comparison of Low-Power Implementations of Trivium and Grain. In Workshop on *The State of the Art of Stream Ciphers (SASC2007)* pages 236-246, 2007.

[8] S. Fischer, S. Khazaei & W. Meier. Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers. In Workshop on *The State of the Art of Stream Ciphers (SASC2008)* pages 33-42, 2008.

[9] K. Gaj, G. Southern & R. Bachimanchi. Comparison of Hardware Performance of Selected Phase II eSTREAM candidates. In Workshop on *The State of the Art of Stream Ciphers (SASC2007)* pages 225-235, 2007.

[10] T. Good, W. Chelton & M. Benaissa. Review of Stream Cipher Candidates from a Low Resource Hardware Perspective. In Workshop on *Stream Ciphers Revisited (SASC2006)*, 2006.

[11] F. K. Gürkaynak, P. luethi, N. Bernold, R. Blattmann, V. Goode, M. Marghitola, H. Kaeslin, N. Felber & W. Fichtner. Hardware Evaluation of eSTREAM Candidates. In Workshop on *Stream Ciphers Revisited (SASC2006)*, 2006.

[12] A. Maximov & A. Biryukov. Two Trivial Attacks on TRIVIUM. In Workshop on *The State of the Art of Stream Ciphers (SASC2007)* pages 1-16, 2007.

[13] M. Rogawski. Hardware Evaluation of eSTREAM Candidates. In Workshop on *The State of the Art of Stream Ciphers (SASC2007)* pages 215-224, 2007.

[14] M. S. Turan, A. Doğanaksoy & Ç. Çalik. Statistical Analysis of Synchronous Stream Ciphers. In Workshop on *Stream Ciphers Revisited (SASC2006)*, 2006.

[15] M. S. Turan & O. Kara. Linear Approximations for 2-round Trivium. In Workshop on *The State of the Art of Stream Ciphers (SASC2007)* pages 22-31, 2007.