Cryptogr. Commun. (2012) 4:79–104
DOI 10.1007/s12095-011-0059-x

A simple stream cipher with proven properties

Wenpei Si · Cunsheng Ding

Received: 19 April 2011 / Accepted: 21 November 2011 / Published online: 24 December 2011

Abstract Both stream ciphers and block ciphers can be used to provide data confidentiality service. Stream ciphers are preferred in many applications, since they can destroy statistical properties of natural languages to some extent. However, it seems hard to design a stream cipher with many proven security properties. The objective of this paper is to present a binary stream cipher which is secure with respect to a number of attacks, and has reasonable performance. The advantage of the stream cipher over existing ones is that it has more proven security properties.

Keywords Cryptography · Linear span · Sequences · Stream Cipher

Mathematics Subject Classifications (2010) 11T71 · 68P25 · 94A55 · 94A60

1 Introduction

In analogy to error-correcting codes which are subdivided into block and convolutional codes, ciphers are generally classified into block ciphers and stream ciphers. A cipher is called a stream cipher if its encryption transformation defined by a fixed encryption key is time-varying, and is said to be a block cipher otherwise [4, 22]. The essential distinction between block ciphers and stream ciphers is the internal memory. In some books, stream and block ciphers are defined differently. This leads to confusions and is indeed a shame. Stream ciphers are preferred in many applications since they can destroy statistical properties of natural languages to some

W. Si (✉) · C. Ding
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
e-mail: siwenpei@ust.hk

C. Ding
e-mail: cding@ust.hk

extent (See [2] for experimental results demonstrating this fact). Note that any block cipher used in the CBC, CFB, OFB, or counter mode becomes a stream cipher (see Section 6.1.1. of the Handbook of Applied Cryptography [17] for clarification), and is not a block cipher anymore.

However, as many practical stream ciphers are complex in structure, it is sometimes infeasible to analyse and prove their security properties. There is a need to develop a type of stream ciphers, which are simple in structure so that it is relatively easy to evaluate their security properties. One simple type of stream ciphers is the binary additive stream ciphers with a keystream generator based on permutations, which was proposed in [4, Section 11.4]. As shown in Fig. 1, the keystream generator consists of a GF(2^n)-generator, a permutation function over GF(2^n) with good nonlinearity and a linear function from GF(2^n) to GF(2). The GF(2^n) generator produces all elements of GF(2^n) in some order. The secret key of this generator is $k = (a, b)$, where $a, b \in \mathrm{GF}(2^n)^*$. The encryption performs the exclusive-or operation on the corresponding bits of the plaintext stream and the keystream. The decryption process is the same as the encryption process [4]. However, no detailed analyses of this kind of keystream generators was provided in [4]. In addition, it was not explored in which order the elements of GF(2^n) should be generated in this reference.

With a little modification and a specification of the keystream generator proposed in [4], we describe a binary additive stream cipher depicted in Fig. 2. The keystream generator of this stream cipher consists of a cyclic counter with period $N = 2^n - 1$, the permutation $f(x) = x^{2^n-2}$ in GF(2^n), and the trace function $g(x) = \mathrm{Tr}_1^n(x)$ from GF(2^n) to GF(2), where n is a positive integer that defines the size of the secret key and the period of the cyclic counter. The cyclic counter has a memory unit and counts the integers in $\mathbb{Z}_N = \{0, 1, 2, \cdots, N-1\}$ cyclically, and the initial content of the memory unit of the cyclic counter is the initialization value IV which is an integer in $[0, N)$. Let α be any generator of GF(2^n)* (the sender and receiver must agree on the same generator α). Using the cyclic counter we can generate all elements of GF(2^n)* in the order $\alpha^{IV}, \alpha^{1+IV}, \cdots, \alpha^{N-1+IV}$ cyclically. The secret key of this generator is $k = (a, b)$, where $a, b \in \mathrm{GF}(2^n)^*$. The encryption of a message bit is the exclusive-or of the message bit and the corresponding keystream bit. The decryption process is the same as the encryption process. Our cryptographic motivations of using $f(x) = x^{2^n-2}$ and $g(x) = \mathrm{Tr}_1^n(x)$ in this cipher are explained in subsequent sections and summarized in the last section of this paper.

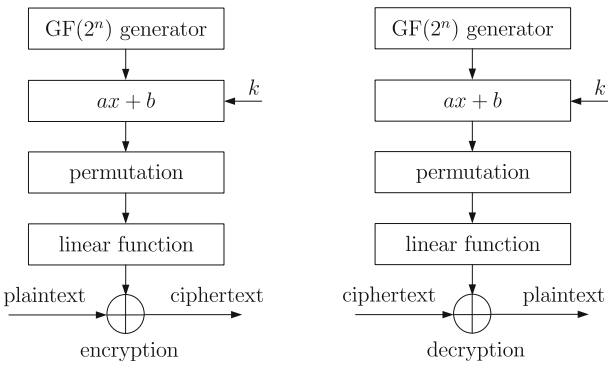

Fig. 1 A binary additive stream cipher based on permutations [4]

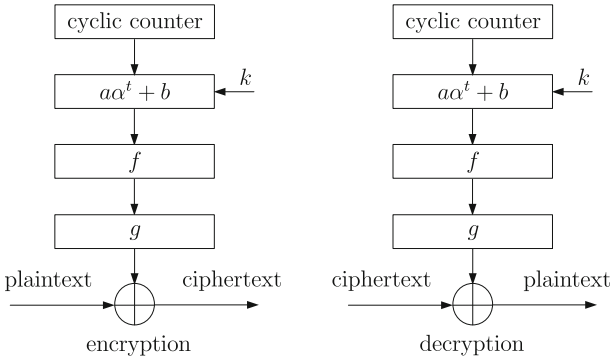

Fig. 2 A binary additive stream cipher based on permutations with $f(x) = x^{2^n-2}$ and $g(x) = \mathrm{Tr}_1^n(x)$

We now compare the original keystream generator of the cipher depicted in Fig. 1 with the keystream generator of the cipher of Fig. 2. One difference is that in the cipher of Fig. 2 f does not take b as an input value, while in the original generators f takes all elements of GF(2^n) as input values. The second difference is that in the cipher of Fig. 2 the order in which the elements of GF(2^n) are generated is clearly defined, while in the original generators the order is not specified. The third difference is that in the cipher of Fig. 2 the permutation f is defined, while in the original generators the permutation f is not specified. We also include an initialization vector for the resynchronization purpose.

The objective of this study is to provide a specific binary additive stream cipher based on the permutation x^{2^n-2} and the linear function $\mathrm{Tr}_1^n(x)$, and examine its security aspects. This paper is organized as follows. Section 2 analyses possible attacks against the stream cipher based on permutations, and presents corresponding design criteria for this stream cipher. Security properties of the stream cipher with respect to possible attacks are examined in Section 3. Section 4 deals with the performance of the stream cipher. Section 5 describes the detection of loss of synchronization of the cyclic counters. Resynchronization process is given in Section 6. Section 7 concludes this paper.

2 Possible attacks and design criteria

2.1 The linear complexity attack and the associated design criteria

As a convention, we denote the cardinality of a set S by $|S|$, and use $a \bmod \hat{N}$ to denote the least nonnegative residue r such that $a \equiv r (\mathrm{mod}\ \hat{N})$.

Let $s^L = s(0)s(1)\cdots s(L-1)$ be a finite sequence over GF(q) with length L and $s^\infty = (s(t))_{t=0}^\infty$ be a periodic sequence over GF(q) with period N, where q is a prime power. If

$$s\left(k+\hat{L}\right) + c_{\hat{L}-1}s\left(k+\hat{L}-1\right) + \cdots + c_1 s(k+1) + c_0 s(k) = 0,$$

for all $k \geq 0$, then the polynomial

$$f(x) = x^{\hat{L}} + c_{\hat{L}-1}x^{\hat{L}-1} + \cdots + c_1 x + c_0 \in \mathrm{GF}(q)[x],$$

where GF(q)[x] denotes the set of all polynomials of indeterminate x over GF(q), is called the characteristic polynomial of the sequence s^L over GF(q) [13]. A characteristic polynomial of minimal degree is called the minimal polynomial of the finite sequence s^L. We use LC_{s^L} to denote the linear complexity or linear span of s^L, which is defined to be the degree of the minimal polynomial of s^L. The definitions of the linear complexity and minimal polynomial also apply to the periodic sequence s^∞. Similarly, we denote the linear complexity of the periodic sequence s^∞ by LC_{s^∞}.

In engineering terms, the linear complexity of a sequence is the length of the shortest linear feedback shift register (LFSR) that can generate it. Sequences with large linear complexity are required in some stream ciphers [4].

If the linear complexity of the keystream sequence generated by the stream cipher is $\hat{L}$, then $2\hat{L}$ consecutive output bits of the keystream generator can be used to construct an LFSR of length $\hat{L}$ that produces the same keystream sequence. The equivalent LFSR can be constructed using the Berlekamp-Massey algorithm or by solving a system of linear equations. Hence, the keystream sequence of the stream cipher based on permutations must have large linear complexity.

Design Criterion 1 The linear complexity of the keystream sequence generated by the keystream generator based on permutation f in Fig. 1 should be large.

Since only a segment of the whole period of the keystream sequence is used in real applications, the linear complexity of each segment of the keystream sequence should be as large as possible. In order to examine the linear complexity of each segment, we can use the linear complexity profile of a sequence, which is obtained by computing the linear complexity of $s(0)s(1)\cdots s(L-1)$ against L, for $L = 1, 2, \cdots$. In [22, 23], it is stated that a cryptographically strong sequence should have a linear complexity near the maximum possible, and its linear complexity profile should follow the $L/2$ line "closely but irregularly".

Design Criterion 2 The linear complexity profile of the keystream sequence generated by the keystream generator based on permutation f in Fig. 1 should follow the $L/2$ line "closely but irregularly".

2.2 The linear complexity stability attack and the associated design criterion

Even if the keystream sequence has large linear complexity, we can still use a sequence with low linear complexity to approximate the original keystream sequence, if the Hamming distance between these two sequences is very small.

In such a case, changing a small number of entries in a sequence decreases the linear complexity of the sequence to a large extent, we say that the linear complexity of such a sequence is not stable. The linear complexity stability issue was first proposed in [6]. As a measure of the linear complexity stability, the sphere complexity for both finite and periodic sequences was introduced in [7]. Later, k-error linear complexity, which is the minimum of the linear complexity and the sphere complexity, was proposed in [23].

Let ℓ be any integer with $0 < \ell < L$. The sphere complexity of s^L is defined to be

$$\mathrm{SC}_\ell\left(s^L\right) = \min_{0<W_H(\hat{s}^L)\le\ell} \mathrm{LC}_{s^L+\hat{s}^L},$$

where $\hat{s}^L$ is any sequence of length L over GF(q) and $W_H(\hat{s}^L)$ denotes the Hamming weight of $\hat{s}^L$. Similarly, the sphere complexity of the periodic sequence s^∞ is defined as

$$\mathrm{SC}_\ell(s^\infty) = \min_{0<W_H(\hat{s}^N)\le\ell} \mathrm{LC}_{s^\infty+\hat{s}^\infty},$$

where $\hat{s}^N$ is the first periodic segment of the sequence $\hat{s}^\infty$ over GF(q).

Based on the linear complexity stability, the best affine approximation (BAA) attacks on certain stream ciphers were developed in [7, Chapter 3]. For the binary additive stream cipher based on permutations of Fig 1, if the sphere complexity $\mathrm{SC}_\ell(s^\infty)$ of the keystream sequence is small for small ℓ, we can construct an LFSR with short length to approximate the original keystream generator. Hence we have another design requirement as follows.

Design Criterion 3 The sphere complexity $\mathrm{SC}_\ell(s^\infty)$ of the keystream sequence generated by the keystream generator based on permutations in Fig. 1 should be large enough for small ℓ.

2.3 The key approximation attack and the associated design criteria

In this kind of attack, one will use a key k_2 to decrypt the ciphertext encrypted using the original secret key k_1. Let s_1^∞ and s_2^∞ be two periodic sequences generated by the keystream generator depicted in Fig. 2 with keys k_1 and k_2 respectively. The rate of correct decryption using this attack with respect to the whole period is given by $(1 + C_{s_2,s_1}(0))/2$, where

$$C_{s_2,s_1}(\tau) = \frac{1}{N}\sum_{t=0}^{N-1}(-1)^{s_2(t+\tau)-s_1(t)},\ 0 \le \tau \le N-1$$

is the normalized periodic correlation of the sequences s_1^∞ and s_2^∞. If s_1^∞ and s_2^∞ are cyclically equivalent, i.e., $s_2(t) = s_1(t+r)$ for all t's and an integer r, then $C_{s_2,s_1}(\tau) = C_{s_1,s_1}((\tau+r) \bmod N)$ is called the autocorrelation of the sequence s_1^∞. If s_1^∞ and s_2^∞ are cyclically distinct, then $C_{s_2,s_1}(\tau)$ is the cross-correlation of s_1^∞ and s_2^∞.

Let $S = \{s_1^\infty, s_2^\infty, \cdots, s_K^\infty\}$ be the set of all cyclically distinct periodic sequences with period N generated by a specific type of keystream generators based on permutations in Fig. 2, where $K = |S|$ is the number of distinct sequences in S. Define $C_{\max}(S)$ to be

$$C_{\max}(S) = \max\left|C_{s_i,s_j}(\tau)\right| \ \text{ for any } 0 \le \tau \le N-1,\ \ 1 \le i, j \le K,$$

where $\tau \neq 0$ if $i = j$. Obviously, $C_{\max}(S)$ is the maximum value of all nontrivial autocorrelation and cross-correlation of the sequences in the sequence family S. By the Welch bound [25], $C_{\max}(S) \ge \sqrt{\frac{K-1}{NK-1}}$. In order to prevent against the key approximation attack, we need the following design criterion.

Design Criterion 4 The value $C_{\max}(S)$ should be less than or equal to $c\sqrt{\frac{K-1}{NK-1}}$ for some small constant c compared with N.

In real applications, only a small segment of the keystream sequence is used for encryption. If a segment of length L of the keystream sequence is used for encryption, then the rate of correct decryption using this attack is $(1 + C^{(L)}_{s_2,s_1}(0))/2$, where

$$C^{(L)}_{s_2,s_1}(\tau) = \frac{1}{L}\sum_{t=0}^{L-1}(-1)^{s_2(t+\tau)-s_1(t)}, \quad 0 \le \tau \le N-1.$$

Let the sequence $\left\{\max_{1\le\tau\le N-1}\left|C^{(L)}_{s_i,s_i}(\tau)\right| \mid 1 \le L \le N\right\}$ be the periodic autocorrelation profile of the sequence s_i^∞, and the sequence

$$\left\{\max_{0\le\tau\le N-1}\left|C^{(L)}_{s_i,s_j}(\tau)\right| \mid 1 \le L \le N, i \ne j\right\}$$

be the periodic cross-correlation profile of the sequences s_i^∞ and s_j^∞.

Similar to $C_{\max}(S)$, we define

$$C^{(L)}_{\max}(S) = \max\left|C^{(L)}_{s_i,s_j}(\tau)\right| \text{ for any } 0 \le \tau \le N-1, \quad 1 \le i, j \le K,$$

where $\tau \ne 0$ if $i = j$. As L approaches to N, $C^{(L)}_{\max}(S)$ should approach to $c\sqrt{\frac{K-1}{NK-1}}$ for some small constant c.

Design Criterion 5 For the keystream sequence generated by the keystream generator based on permutations, $C^{(L)}_{\max}(S)$ should be close to $c\sqrt{\frac{K-1}{NK-1}}$ for some small constant c compared with N, when L is close to N.

2.4 The linear approximation attack and the associated design criterion

In order to analyze this kind of attack, we need to examine the nonlinearity of the underlying functions of the keystream generator. A function f from $(A, +)$ to $(B, +)$ is called linear if and only if $f(x + y) = f(x) + f(y)$ for all $x, y \in A$. A function g is called affine if $g = f + c$ for a linear function f from $(A, +)$ to $(B, +)$ and a constant $c \in B$. One way of measuring the nonlinearity of a function f from $(A, +)$ to $(B, +)$ is defined by

$$N_f = \min_{l\in\mathscr{L}} d(f, l),$$

where $\mathscr{L}$ denotes the set of all affine functions from $(A, +)$ to $(B, +)$, and $d(f, l) = |\{x \in A | f(x) - l(x) \ne 0\}|$ is the Hamming distance of the two functions f and l. This measure is related to affine (linear) approximation attacks [4].

In cryptography, both linear and nonlinear building blocks are needed to achieve diffusion and confusion. The basic idea of linear approximation attack on the stream cipher is to use a linear function to replace a nonlinear function. If a piece of the keystream sequence is known, one can use another decryption transform to approximate the original decryption transform. In order to correctly decrypt the ciphertext with a higher probability, the Hamming distance between the linear function and the nonlinear function should be as small as possible. The best affine approximation (BAA) attacks on some stream ciphers [7] were based on this idea.

The best affine approximation attack on the stream cipher based on permutations is a known plaintext attack and is conducted as follows. Assume that a piece of

keystream sequence $s(i)s(i+1)\cdots s(i+m)$ is known. By Fig. 2, we have $h(x) = h(a\alpha^j + b) = s(j)$ for all $j = i, i+1, \cdots, i+m$, where $h(x) = g(f(x)) = \mathrm{Tr}_1^n\left(x^{2^n-2}\right)$ as described in Section 1. Consider all affine functions from GF(2^n) to GF(2). Choose the one with the smallest Hamming distance with the function h. We can use the selected affine function to replace h and use the obtained generator to approximate the original keystream generator. It is well known that any linear function from GF(2^n) to GF(2) can be expressed as $\mathrm{Tr}_1^n(wx)$ for some $w \in \mathrm{GF}(2^n)$ [13]. Thus any affine function from GF(2^n) to GF(2) can be denoted by $\mathrm{Tr}_1^n(wx) + c, i \geq 0$, for some $c \in \mathrm{GF}(2)$.

Let $x = a\alpha^i + b$, then we have the following equations

$$\begin{aligned}
&\mathrm{Tr}_1^n\left(wa\alpha^i\right) + \mathrm{Tr}_1^n(wb) + c = s(i),\\
&\mathrm{Tr}_1^n\left(wa\alpha^{i+1}\right) + \mathrm{Tr}_1^n(wb) + c = s(i+1),\\
&\qquad\qquad \cdots\\
&\mathrm{Tr}_1^n\left(wa\alpha^{i+m}\right) + \mathrm{Tr}_1^n(wb) + c = s(i+m).
\end{aligned}$$

It follows that

$$\begin{aligned}
&\mathrm{Tr}_1^n\left(wa\alpha^i(\alpha-1)\right) = s(i+1) - s(i),\\
&\mathrm{Tr}_1^n\left(wa\alpha^{i+1}(\alpha-1)\right) = s(i+2) - s(i+1),\\
&\qquad\qquad \cdots\\
&\mathrm{Tr}_1^n\left(wa\alpha^{i+m-1}(\alpha-1)\right) = s(i+m) - s(i+m-1). \qquad (1)
\end{aligned}$$

Note that the sequence $\left(\mathrm{Tr}_1^n(wa\alpha^j(\alpha-1)\right)_{j=0}^{\infty}$ is an m-sequence if $wa \neq 0$. When $m \geq 2n$, we can use (1) to construct an LFSR to approximate the original keystream generator.

In order to thwart the best affine approximation attacks, the Hamming distance between the function h and any affine function from GF(2^n) to GF(2) should be as large as possible. However, according to the so-called covering radius bound [3], the minimum Hamming distance between a Boolean function $\hat{f}$ from GF(2^n) to GF(2) and all affine functions from GF(2^n) to GF(2) is $N_{\hat{f}} \leq 2^{n-1} - 2^{n/2-1}$.

Design Criterion 6 For the function $h = g \circ f$ of the keystream generator based on permutations in Fig. 2, the Hamming distance between h and every affine function from GF(2^n) to GF(2) should be as close to $2^{n-1} - 2^{n/2-1}$ as possible.

3 Security properties of the binary additive stream cipher based on $f(x) = x^{2^n-2}$ and $g(x) = \mathrm{Tr}_1^n(x)$

According to Fig. 2, the periodic keystream sequence $\hat{s}^{\infty}$ of the stream cipher is

$$\hat{s}(t) = g\left(f\left(a\alpha^t + b\right)\right) = \mathrm{Tr}_1^n\left((a\alpha^t + b)^{2^n-2}\right), \qquad (2)$$

where t is the content of the counter at time unit t.

3.1 The security of the stream cipher with respect to the linear complexity attack

The following Lemma is needed to compute the linear complexity of the keystream sequence of the stream cipher.

Lemma 1 [1] *Any periodic sequence s^∞ over* GF(q) *of period q^n-1 has a unique expansion of the form*

$$s(t)=\sum_{i=0}^{q^n-2} c_i\alpha^{it},\ \textit{for all } t\geq 0,$$

where α is a generator of GF$(q^n)^*$ *and $c_i\in$* GF(q^n). *Let the index set $I=\{i\mid c_i\neq 0\}$, then the minimal polynomial $m(x)$ of s^∞ is*

$$m(x)=\prod_{i\in I}\left(x-\alpha^i\right),$$

and the linear complexity of s^∞ is $|I|$.

Let $a=\alpha^{r_a}$ and $b=\alpha^{r_b}$, then (2) becomes

$$\hat{s}(t)=\mathrm{Tr}_1^n\left((\alpha^{t+r_a}+b)^{2^n-2}\right)=\mathrm{Tr}_1^n\left(b^{2^n-2}(\alpha^{t+r_a-r_b}+1)^{2^n-2}\right). \tag{3}$$

Let $\beta=b^{2^n-2}$. For the periodic sequence s^∞ with the expression

$$s(t)=\mathrm{Tr}_1^n\left(\beta(\alpha^t+1)^{2^n-2}\right), \tag{4}$$

we have $s(t+r_a-r_b)=\hat{s}(t)$ for all t's. It follows that $\hat{s}^\infty$ and s^∞ are cyclically equivalent. Thus $\mathrm{LS}_{s^\infty}=\mathrm{LS}_{\hat{s}^\infty}$, and we just need to examine the linear complexity of the sequence s^∞ instead of $\hat{s}^\infty$.

Using the well-known binomial formula

$$(u+v)^m=\sum_{i=0}^{m}\binom{m}{i}u^iv^{m-i},\ \text{where}\ \binom{m}{i}=\frac{m!}{i!(m-i)!},$$

we can expand (4) as,

$$s(t)=\mathrm{Tr}_1^n\left(\beta\left(\sum_{i=0}^{2^n-2}\binom{2^n-2}{i}\alpha^{it}\right)\right)=\mathrm{Tr}_1^n\left(\sum_{i=0}^{2^n-2}\beta\binom{2^n-2}{i}\alpha^{it}\right).$$

The following Lemma is needed to evaluate $\binom{2^n-2}{i}$ mod 2.

Lemma 2 [15] *If m and i are positive integers with the base-2 representations $m=m_r2^r+\cdots+m_12+m_0$ and $i=i_r2^r+\cdots+i_12+i_0$ respectively, then*

$$\binom{m}{i}\equiv\binom{m_r}{i_r}\cdots\binom{m_1}{i_1}\binom{m_0}{i_0}\ (\mathrm{mod}\ 2).$$

Since $0\leq i\leq 2^n-2$, we have the base-2 representations $2^n-2=1\cdot 2^{n-1}+\cdots+1\cdot 2^1+0\cdot 2^0$ and $i=i_{n-1}2^{n-1}+\cdots+i_12+i_0$ respectively. By Lemma 2, we have

$$\binom{2^n-2}{i}\equiv\binom{1}{i_{n-1}}\cdots\binom{1}{i_1}\binom{0}{i_0}\equiv\binom{0}{i_0}\ (\mathrm{mod}\ 2).$$

Thus, $\binom{2^n-2}{i} \bmod 2 = 1$ iff $i_0 = 0$, which means $i \bmod 2 = 0$. Since s^∞ is over GF(2), we have

$$s(t) = \mathrm{Tr}_1^n \left(\sum_{i=0}^{2^{n-1}-1} \beta \alpha^{2it} \right). \tag{5}$$

To determine the linear complexity, we need to introduce cyclotomic cosets. The cyclotomic coset C_j modulo $2^n - 1$ is defined as

$$C_j = \left\{ j, j2, j2^2, \cdots, j2^{m_j-1} \right\},$$

where m_j is the smallest positive integer such that $j2^{m_j} \equiv j \pmod{2^n - 1}$ [16, 18]. The integer m_j is also called the size of C_j. The subscript j is chosen as the smallest integer in C_j, and j is called the coset leader of C_j. We denote by Γ the set of all coset leaders modulo $2^n - 1$, and by Γ^* the set $\Gamma - \{0\}$. Let $B_j = \{i \mid i \in C_j \text{ and } 0 \le i \le 2^{n-1} - 1\}$.

Lemma 3 [9] *A 2-cyclotomic coset modulo $2^n - 1$ of length l exists iff l divides n.*

By Lemma 3, for all $j \in \Gamma$ there exists an unique integer l_j such that $n = m_j * l_j$.

3.1.1 The general case

For any $j \in \Gamma$ and $i \in B_j$, there is one and only one $0 \le k_{ij} \le m_j - 1$, such that

$$2i \cdot 2^{k_{ij}} \equiv j \pmod{N}. \tag{6}$$

It follows from (5) that

$$\begin{aligned}
s(t) &= \mathrm{Tr}_1^n \left(\sum_{i=0}^{2^{n-1}-1} \beta \alpha^{2it} \right) \\
&= \mathrm{Tr}_1^n \left(\sum_{j \in \Gamma} \sum_{i \in B_j} \beta \alpha^{2it} \right) \\
&= \sum_{j \in \Gamma} \sum_{i \in B_j} \mathrm{Tr}_1^n \left(\beta \alpha^{2it} \right) \\
&= \sum_{j \in \Gamma} \sum_{i \in B_j} \sum_{u=0}^{m_j-1} \sum_{h=0}^{l_j-1} \left(\beta^{2^{k_{ij}}} \alpha^{jt} \right)^{2^{u+m_j h}} \\
&= \sum_{j \in \Gamma} \sum_{i \in B_j} \sum_{u=0}^{m_j-1} \sum_{h=0}^{l_j-1} \left(\beta^{2^{k_{ij}+u+m_j h}} \alpha^{jt 2^{u+m_j h}} \right)
\end{aligned}$$

$$= \sum_{j\in\Gamma}\sum_{i\in B_j}\sum_{u=0}^{m_j-1}\alpha^{jt2^u}\left(\sum_{h=0}^{l_j-1}\beta^{2^{k_{ij}+u+m_jh}}\right)$$
$$= \sum_{j\in\Gamma}\sum_{u=0}^{m_j-1}\alpha^{jt2^u}\left(\sum_{i\in B_j}\sum_{h=0}^{l_j-1}\beta^{2^{k_{ij}+u+m_jh}}\right)$$
$$= \sum_{j\in\Gamma}\sum_{u=0}^{m_j-1}\alpha^{jt2^u}\left(\sum_{i\in B_j}\sum_{h=0}^{l_j-1}\beta^{2^{k_{ij}+m_jh}}\right)^{2^u}. \tag{7}$$

For the sake of representation, we want to rewrite (6) and (7) in a simpler way.

Lemma 4 *For any $j\in\Gamma$, if $(i_1,h_1)\neq(i_2,h_2)$, where $i_1,i_2\in B_j$ and $h_1,h_2\in\{0,1,\cdots,l_j-1\}$, we have $2^{k_{i_1j}+m_jh_1}\not\equiv 2^{k_{i_2j}+m_jh_2} \pmod N$.*

Proof Suppose

$$2^{k_{i_1j}+m_jh_1}\equiv 2^{k_{i_2j}+m_jh_2} \pmod N. \tag{8}$$

Since $0\le k_{i_1j}+m_jh_1\le n-1$ and $0\le k_{i_2j}+m_jh_2\le n-1$, we have $1\le 2^{k_{i_1j}+m_jh_1}\le 2^{n-1}$ and $1\le 2^{k_{i_2j}+m_jh_2}\le 2^{n-1}$. Thus (8) gives $2^{k_{i_1j}+m_jh_1}=2^{k_{i_2j}+m_jh_2}$. Then we obtain

$$k_{i_1j}+m_jh_1=k_{i_2j}+m_jh_2. \tag{9}$$

Since $(k_{i_1j}+m_jh_1)\equiv(k_{i_2j}+m_jh_2) \pmod{m_j}$, we have that $k_{i_1j}\equiv k_{i_2j} \pmod{m_j}$. Note that $0\le k_{i_1j}\le m_j-1$ and $0\le k_{i_2j}\le m_j-1$, we have then

$$k_{i_1j}=k_{i_2j}. \tag{10}$$

Also, from (9) and (10), we obtain $m_jh_1=m_jh_2$, which gives $h_1=h_2$. Hence $(i_1,h_1)=(i_2,h_2)$. □

Let $f_j(x)=\sum_{i\in B_j}\sum_{h=0}^{l_j-1}x^{2^{k_{ij}+m_jh} \bmod N}\in \mathrm{GF}(2)[x]$, where $j\in\Gamma$. By Lemma 4, no terms in $f_j(x)$ can cancel each other. Also, $0\le k_{ij}+m_jh\le n-1$, we have

$$f_j(x)=\sum_{i\in B_j}\sum_{h=0}^{l_j-1}x^{2^{k_{ij}+m_jh}}=\sum_{k=0}^{n-1}f_{j,k}x^{2^k}\in\mathrm{GF}(2)[x].$$

where

$$f_{j,k}=\begin{cases}1 \text{ if } k=k_{ij}+m_jh \text{ for some } i\in B_j \text{ and } h\in\{0,1,\cdots,l_j-1\},\\ 0 \text{ otherwise.}\end{cases} \tag{11}$$

Lemma 5 *For any $j\in\Gamma$ and $k\in\{0,1,\cdots,n-1\}$, $k=k_{ij}+m_jh$ for some $i\in B_j$ and $h\in\{0,1,\cdots,l_j-1\}$ holds iff $(j2^{n-k} \bmod N) \bmod 2=0$.*

Proof If $k=k_{ij}+m_jh$ for some $i\in B_j$ and $h\in\{0,1,\cdots,l_j-1\}$, then by (6), we have $2i\cdot 2^{k-m_jh}\equiv 2i\cdot 2^k\equiv j \pmod N$. Thus $2i\equiv j2^{n-k} \pmod N$. Since $0\le 2i\le 2^n-2$, we obtain $(j2^{n-k} \bmod N)=2i$. Thus $(j2^{n-k} \bmod N) \bmod 2=0$.

If $(j2^{n-k} \bmod N) \bmod 2 = 0$, then $(j2^{n-k} \bmod N) = 2i'$. Since $0 \le 2i' \le 2^n - 2$, we have $0 \le i' \le 2^{n-1} - 1$. Note that $i' \in C_j$, we have $i' \in B_j$. It follows from (6) that $2i' \cdot 2^k \equiv 2i' \cdot 2^{k \bmod m_j} \equiv j \equiv 2i' \cdot 2^{k_{i'j}} \pmod N$. Thus

$$\left(2i' \cdot 2^{k \bmod m_j} \bmod N\right) = \left(2i' \cdot 2^{k_{i'j}} \bmod N\right).$$

Since $0 \le k_{i'j} \le m_j - 1$, we have $k \bmod m_j = k_{i'j}$. Hence $k = k_{i'j} + m_j h$ for some $i' \in B_j$ and $h \in \{0, 1, \cdots, l_j - 1\}$. □

By Lemma 5 and (11) we have

$$f_{j,k} = \begin{cases} 1 \text{ if } (j2^{n-k} \bmod N) \bmod 2 = 0, \\ 0 \text{ otherwise.} \end{cases} \tag{12}$$

Then (7) becomes

$$\begin{aligned} s(t) &= \sum_{j \in \Gamma} \sum_{u=0}^{m_j - 1} \alpha^{jt2^u} \left(f_j(\beta)\right)^{2^u} \\ &= \sum_{j \in \Gamma} \sum_{u=0}^{m_j - 1} \alpha^{jt2^u} \left(\sum_{k=0}^{n-1} f_{j,k} \beta^{2^k}\right)^{2^u} \\ &= \sum_{j \in \Gamma} \sum_{u=0}^{m_j - 1} \alpha^{jt2^u} \left(\sum_{k=0}^{n-1} f_{j,k} \beta^{2^{k+u}}\right). \end{aligned} \tag{13}$$

Since $\beta = b^{2^n - 2} = \alpha^{r_b(2^n - 2)} \in \mathrm{GF}(2^n)$, $\beta^{2^n} = \beta$. Hence (13) becomes

$$s(t) = \sum_{j \in \Gamma} \sum_{u=0}^{m_j - 1} \alpha^{jt2^u} \left(\sum_{k=0}^{n-1} f_{j,k} \beta^{2^{(k+u) \bmod n}}\right). \tag{14}$$

Thus we have expanded the keystream sequence as the sum of powers-of-α. Then by Lemma 1, we need to count the number of nonzero terms in (14). Since in (14), the terms α^{jt2^u}, where $j \in \Gamma$ and $u \in \{0, 1, \cdots, m_j - 1\}$, are pairwise distinct, we have

$$\begin{aligned} \mathrm{LC}_{s^\infty} &= \left| \left\{ (j,u) \mid \sum_{k=0}^{n-1} f_{j,k}\, \beta^{2^{(k+u) \bmod n}} \neq 0,\ j \in \Gamma, u \in \{0, 1, \cdots, m_j - 1\} \right\} \right| \\ &= \left| \left\{ (j,u) \mid j \in \Gamma, u \in \{0, 1, \cdots, m_j - 1\} \right\} \right| \\ &\quad - \left| \left\{ (j,u) \mid \sum_{k=0}^{n-1} f_{j,k}\, \beta^{2^{(k+u) \bmod n}} = 0,\ j \in \Gamma, u \in \{0, 1, \cdots, m_j - 1\} \right\} \right| \\ &= 2^n - 1 - \left| \left\{ (j,u) \mid \sum_{k=0}^{n-1} f_{j,k}\, \beta^{2^{(k+u) \bmod n}} = 0,\ j \in \Gamma, u \in \{0, 1, \cdots, m_j - 1\} \right\} \right| \\ &= 2^n - 1 - \left| \left\{ (j,u) \mid \sum_{k=0}^{n-1} f_{j,(k-u) \bmod n}\, \beta^{2^k} = 0,\ j \in \Gamma, u \in \{0, 1, \cdots, m_j - 1\} \right\} \right|. \end{aligned} \tag{15}$$

To determine the number of (j, u)'s such that $\sum_{k=0}^{n-1} f_{j,(k-u) \bmod n}\, \beta^{2^k} = 0$, we need the following two lemmas.

Lemma 6 *For all $j \in \Gamma^*$, we have that j is odd and $1 \le j \le 2^{n-1} - 1$.*

Proof Suppose j is even. Since $j \equiv \frac{j}{2} \cdot 2 \pmod N$, we obtain $\frac{j}{2} \in C_j$. Since $2 \le j \le 2^n - 2$, we have $1 \le \frac{j}{2} < j \le 2^n - 2$, which contradicts with the definition of Γ^*.

Suppose $2^{n-1} \le j \le 2^n - 2$, then $2^n \le 2j \le 2^{n+1} - 4$. Thus $2^n - N \le 2j - N \le 2^{n+1} - 4 - N$, which gives $1 \le 2j - N \le 2^n - 3$. However, since $2j - N < j$ and $2j - N \in C_j$, this also contradicts with the definition of Γ^*. □

Then we examine the polynomial $\sum_{k=0}^{n-1} f_{j,(k-u) \bmod n}\, x^{2^k} \in \mathrm{GF}(2)[x]$.

Lemma 7 *If $(j_1, u_1) \ne (j_2, u_2)$, where $j_1, j_2 \in \Gamma$, $u_i \in \{0, 1, \cdots, m_{j_i} - 1\}$ for $i \in \{1, 2\}$, then the polynomial $\sum_{k=0}^{n-1} f_{j_1,(k-u_1) \bmod n}\, x^{2^k}$ is different from $\sum_{k=0}^{n-1} f_{j_2,(k-u_2) \bmod n}\, x^{2^k}$.*

Proof Suppose $\sum_{k=0}^{n-1} f_{j_1,(k-u_1) \bmod n}\, x^{2^k}$ and $\sum_{k=0}^{n-1} f_{j_2,(k-u_2) \bmod n}\, x^{2^k}$ are identical. We have

$$f_{j_1,(k-u_1) \bmod n} = f_{j_2,(k-u_2) \bmod n}, \tag{16}$$

for all $k \in \{0, 1, \cdots, n-1\}$. According to (12), (16) is equivalent to

$$\left(j_1\, 2^{n-((k-u_1) \bmod n)} \bmod N\right) \bmod 2 = \left(j_2\, 2^{n-((k-u_2) \bmod n)} \bmod N\right) \bmod 2,$$

which is further equivalent to

$$\left(j_1\, 2^{n-k+u_1} \bmod N\right) \bmod 2 = \left(j_2\, 2^{n-k+u_2} \bmod N\right) \bmod 2, \tag{17}$$

for all $k \in \{0, 1, \cdots, n-1\}$. Since $0 \le j_1 \le 2^n - 2$ and $0 \le j_2 \le 2^n - 2$, let the base-2 representations of j_1 and j_2 be $j_1 = \sum_{i=0}^{n-1} j_{1,i} 2^i$ and $j_2 = \sum_{i=0}^{n-1} j_{2,i} 2^i$ respectively. Then we have

$$j_1\, 2^{n-k+u_1} \bmod N = \sum_{i=0}^{n-1} j_{1,i} 2^{(i+n-k+u_1) \bmod n} = \sum_{i=0}^{n-1} j_{1,(i-n+k-u_1) \bmod n} 2^i.$$

It follows that

$$\left(j_1\, 2^{n-k+u_1} \bmod N\right) \bmod 2 = j_{1,(0-n+k-u_1) \bmod n} = j_{1,(k-u_1) \bmod n}.$$

Similarly, we have

$$\left(j_2\, 2^{n-k+u_2} \bmod N\right) \bmod 2 = j_{2,(k-u_2) \bmod n}.$$

By (17), we obtain

$$j_{1,(k-u_1) \bmod n} = j_{2,(k-u_2) \bmod n}, \tag{18}$$

for all $k \in \{0, 1, \cdots, n-1\}$. Note that

$$j_1\, 2^{u_1} \bmod N = \sum_{k=0}^{n-1} j_{1,k} 2^{(k+u_1) \bmod n} = \sum_{k=0}^{n-1} j_{1,(k-u_1) \bmod n} 2^k$$

and

$$j_2\, 2^{u_2} \bmod N = \sum_{k=0}^{n-1} j_{2,k} 2^{(k+u_2) \bmod n} = \sum_{k=0}^{n-1} j_{2,(k-u_2) \bmod n} 2^k.$$

By (18), we have

$$j_1\, 2^{u_1} \bmod N = j_2\, 2^{u_2} \bmod N. \tag{19}$$

It follows that j_1 and j_2 are in the same cyclotomic coset. Since $j_1 \in \Gamma$ and $j_2 \in \Gamma$, we have $j_1 = j_2$. Hence (19) becomes $(j_1\, 2^{u_1} \bmod N) = (j_1\, 2^{u_2} \bmod N)$. Since $0 \le u_1 \le m_{j_1} - 1$ and $0 \le u_2 \le m_{j_1} - 1$, we obtain $u_1 = u_2$. Hence $(j_1, u_1) = (j_2, u_2)$. □

Theorem 1 *The linear complexity* LC_{s^∞} *of the sequence* $s^\infty \ge 2^{n-1}$.

Proof Let $V(x) = \sum_{k=0}^{n-1} v_k x^{2^k} \in \mathrm{GF}(2)[x]$ and $V^* = \{V(x) \mid V(x) \ne 0\}$. Then there are $2^n - 1$ different polynomials in V^*. Since there are totally $2^n - 1$ different values of (j, u), where $j \in \Gamma$ and $u \in \{0, 1, \cdots, m_j - 1\}$, according to Lemma 7, there are $2^n - 1$ different polynomials $\sum_{k=0}^{n-1} f_{j,(k-u) \bmod n}\, x^{2^k}$. Also, from Lemma 6, $2j \in C_j$ is even, for all $j \in \Gamma^*$. Thus $f_{j,n-1} = 1$ by (12). When $j = 0$, we have $f_{j,k} = 1$, for all $k \in \{0, 1 \cdots, n-1\}$. It follows that the polynomials $\sum_{k=0}^{n-1} f_{j,(k-u) \bmod n}\, x^{2^k} \ne 0$, for all $j \in \Gamma$. Hence the polynomials $\sum_{k=0}^{n-1} f_{j,(k-u) \bmod n}\, x^{2^k}$ exhaust all possible polynomials in V^*. By (15) and Lemma 7, we have

$$\begin{aligned}
\mathrm{LC}_{s^\infty} &= 2^n - 1 - \left| \left\{ (j, u) \mid \sum_{k=0}^{n-1} f_{j,(k-u) \bmod n}\, \beta^{2^k} = 0,\ j \in \Gamma, u \in \{0, 1, \cdots, m_j - 1\} \right\} \right| \\
&= 2^n - 1 - \left| \left\{ V(x) \mid V(\beta) = 0, V(x) \in V^* \right\} \right| \\
&= 2^n - 1 - \left(\left| \left\{ V(x) \mid V(\beta) = 0 \right\} \right| - 1 \right) \\
&= 2^n - \left| \left\{ V(x) \mid V(\beta) = 0 \right\} \right|. \qquad (20)
\end{aligned}$$

Note that $C = \left\{ (v_0, v_1, \cdots, v_{n-1}) \mid V(x) = \sum_{k=0}^{n-1} v_k x^{2^k} \in \mathrm{GF}(2)[x], V(\beta) = 0 \right\}$ is an $[n, m, d]$ linear code over GF(2), where m is the dimension and d is the minimum nonzero Hamming weight of C. Since $\beta^{2^k} \ne 0$ for all $k \in \{0, 1, \cdots, n-1\}$,

$$d \ge 2. \tag{21}$$

According to the Singleton bound [19], $m \le n - d + 1 \le n - 1$. It follows that the size of the linear code is $2^m \le 2^{n-1}$. Thus $|\{V(x) \mid V(\beta) = 0\}| \le 2^{n-1}$. Then we have $\mathrm{LC}_{s^\infty} \ge 2^n - 2^{n-1} = 2^{n-1}$. □

3.1.2 Special cases

Theorem 2 *If* n *is a prime and* $\beta \ne 1$, *then* $\mathrm{LC}_{s^\infty} \ge 2^{n-1} + 2^{n-2}$.

Proof Let $s = r_b(2^n - 2) \bmod N$. When $\beta = b^{2^n-2} = \alpha^{r_b(2^n-2)} = \alpha^s \ne 1$, we have $s \ne 0$. If there are k_1 and k_2, where $0 \le k_1, k_2 \le n-1$, such that $\beta^{2^{k_1}} + \beta^{2^{k_2}} = 0$, then $\beta^{2^{k_1}} = \alpha^{s\,2^{k_1}} = \beta^{2^{k_2}} = \alpha^{s\,2^{k_2}}$. Since α is a generator of $\mathrm{GF}(2^n)^*$, we have $\alpha^{s\,2^{k_1}} = \alpha^{s\,2^{k_2}}$ iff $\left(s\,2^{k_1} \bmod N\right) = \left(s\,2^{k_2} \bmod N\right)$. Since n is prime, by Lemma 3, the length of each

cyclotomic coset except $\{0\}$ is n. Thus $k_1 = k_2$. It follows that $d \neq 2$. By (21), the minimum nonzero Hamming weight $d \geq 3$. According to the Singleton bound [19], the size of the linear code is $2^m \leq 2^{n-2}$. Thus $|\{V(x) \mid V(\beta) = 0\}| \leq 2^{n-2}$ and

$$\mathrm{LC}_{s^\infty} \geq 2^n - 2^{n-2} = 2^{n-1} + 2^{n-2}.$$

□

Theorem 3 *If* $\beta = 1$, *then* $\mathrm{LC}_{s^\infty} = 2^{n-1}$.

Proof By (20), when $\beta = 1$,

$$\begin{aligned}
\mathrm{LC}_{s^\infty} &= 2^n - |\{V(x) \mid V(1) = 0, V(x) \in \mathrm{GF}(2)[x]\}| \\
&= 2^n - \left| \left\{ (v_0, v_1, \cdots, v_{n-1}) \mid \left(\sum_{k=0}^{n-1} v_k \right) \bmod 2 = 0, v_k \in \mathrm{GF}(2) \right\} \right| \\
&= 2^n - \left(\binom{n}{0} + \binom{n}{2} + \binom{n}{4} + \cdots \right) \\
&= 2^n - 2^{n-1} \\
&= 2^{n-1}.
\end{aligned}$$

□

According to Theorems 1, 2 and 3, the linear complexity of the periodic keystream sequence is quite large compared with its period.

Although it is difficult to determine the linear complexity profile of the first period of the keystream sequence of the binary additive stream cipher based on permutations, our experiments show that the sequence has very good linear complexity profile. Since the whole period of the keystream sequence is too large when n is large and only a segment of the keystream sequence is used in practice, we only illustrate the linear complexity profile of a piece of the whole period of the keystream sequence. Figure 3 illustrates the linear complexity profile of the first 20,000 bits of the first period of the keystream sequence s^∞ generated by the keystream generator with $n = 127$, $k = (\alpha^{-3}, \alpha^{-3})$ and $IV = 0$.

Since the linear complexity of each segment of the keystream sequence is about half of its length, the linear complexity attack on the stream cipher does not work.

3.2 The security of the stream cipher with respect to the linear complexity stability attack

In general, it is difficult to obtain theoretical results on the sphere complexity and the sphere complexity profile. Since the sphere complexity profile is more important in practice, we focus attention on the sphere complexity profile. Experimental results show that $\mathrm{SC}_\ell(s^L)$ is good for small ℓ. Figure 4 illustrates the $\mathrm{SC}_1(s^L)$ and $\mathrm{SC}_2(s^L)$ profiles of the first 700 bits of the first period of the keystream sequence s^∞ generated by the keystream generator with $n = 127$, $k = (\alpha^{-3}, \alpha^{-3})$ and $IV = 0$, which indicates that the $\mathrm{SC}_1(s^L)$ and $\mathrm{SC}_2(s^L)$ profiles are near $L/2$. Thus the keystream sequence should be able to resist linear complexity stability attacks.

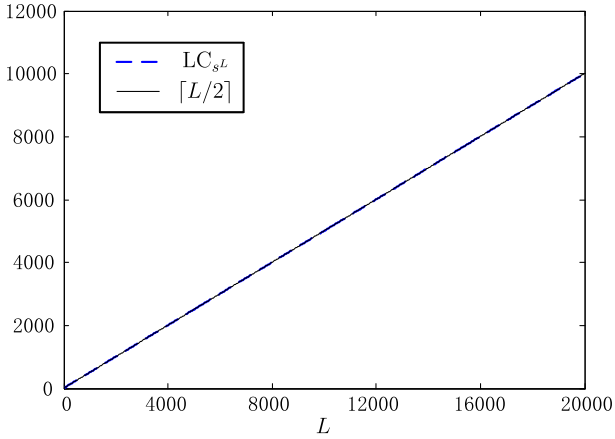

Fig. 3 A linear complexity profile

3.3 The security of the stream cipher with respect to the key approximation attack

In (4), since $|\{\beta \mid \beta \in \mathrm{GF}(2^n)^*\}| = N$, there are N cyclically different keystream sequences with expression (4), which can be generated by the stream cipher with $k = (\beta^{2^n-2}, \beta^{2^n-2})$. Thus in order to give an upper bound on the correlation of all the keystream sequences generated by the stream cipher, it is sufficient to give an upper bound on the correlation of cyclically different keystream sequences with expression (4) generated by the stream cipher.

Let $s_1(t) = \mathrm{Tr}_1^n(\beta_1(\alpha^t + 1)^{2^n-2})$ and $s_2(t) = \mathrm{Tr}_1^n(\beta_2(\alpha^t + 1)^{2^n-2})$. The function χ_1 defined by

$$\chi_1(c) = (-1)^{\mathrm{Tr}_1^n(c)} \text{ for all } c \in \mathrm{GF}(2^n),$$

is the canonical additive character of $\mathrm{GF}(2^n)$. For any $b, c \in \mathrm{GF}(2^n)$, function $\chi_b(c) = \chi_1(bc)$ is also an additive character of $\mathrm{GF}(2^n)$.

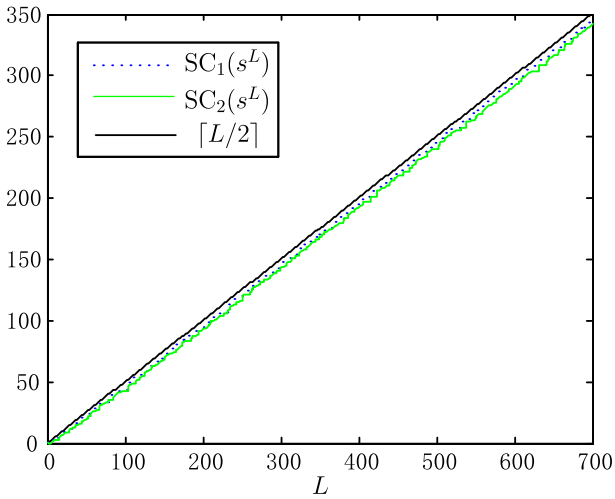

Fig. 4 Sphere complexity $\mathrm{SC}_1(s^L)$ and $\mathrm{SC}_2(s^L)$ profiles

Theorem 4 *For any* $0 \le \tau \le N-1$ *and* $\beta_1, \beta_2 \in \mathrm{GF}(2^n)^*$, *we have*

$$\left|C_{s_2,s_1}(\tau)\right| \le \frac{1}{N}\left(2\sqrt{2^n}+4\right),$$

where $\tau \ne 0$ *if* $\beta_1 = \beta_2$.

Proof When $1 \le \tau \le N-1$ and $\beta_1, \beta_2 \in \mathrm{GF}(2^n)^*$, the correlation of the sequences s_1^∞ and s_2^∞ is

$$\begin{aligned}
C_{s_2,s_1}(\tau) &= \frac{1}{N}\sum_{t=0}^{N-1}(-1)^{s_2(t+\tau)-s_1(t)} \\
&= \frac{1}{N}\sum_{t=0}^{N-1}(-1)^{\mathrm{Tr}_1^n(\beta_2(\alpha^{t+\tau}+1)^{2^n-2})-\mathrm{Tr}_1^n(\beta_1(\alpha^t+1)^{2^n-2})} \\
&= \frac{1}{N}\sum_{t=0}^{N-1}(-1)^{\mathrm{Tr}_1^n\left(\beta_2(\alpha^{t+\tau}+1)^{2^n-2}-\beta_1(\alpha^t+1)^{2^n-2}\right)} \\
&= \frac{1}{N}\sum_{\substack{x\in\mathrm{GF}(2^n)\\ x\ne 0, x\ne -\frac{1}{\alpha^\tau}, x\ne -1}} \chi_1\left(\beta_2(\alpha^\tau x+1)^{-1}-\beta_1(x+1)^{-1}\right) \\
&\quad + \frac{1}{N}\chi_1\left(-\beta_1\left(1-\frac{1}{\alpha^\tau}\right)^{-1}\right) + \frac{1}{N}\chi_1\left(\beta_2(1-\alpha^\tau)^{-1}\right). \qquad (22)
\end{aligned}$$

Let $y = \frac{1-\alpha^\tau}{\alpha^\tau(1+x)} + 1$. Replacing x in (22) by $\frac{1-\alpha^\tau}{\alpha^\tau(y-1)} - 1$, we have

$$\begin{aligned}
C_{s_2,s_1}(\tau) &= \frac{1}{N}\sum_{\substack{y\in\mathrm{GF}(2^n)\\ y\ne\frac{1}{\alpha^\tau}, y\ne 0, y\ne 1}} \chi_1\left(\beta_2\left(\alpha^\tau\left(\frac{1-\alpha^\tau}{\alpha^\tau(y-1)}-1\right)+1\right)^{-1}\right. \\
&\qquad \left. -\beta_1\left(\frac{1-\alpha^\tau}{\alpha^\tau(y-1)}-1+1\right)^{-1}\right) \\
&\quad + \frac{1}{N}\chi_1\left(\frac{\beta_1\alpha^\tau}{1-\alpha^\tau}\right) + \frac{1}{N}\chi_1\left(\frac{\beta_2}{1-\alpha^\tau}\right) \\
&= \frac{1}{N}\sum_{\substack{y\in\mathrm{GF}(2^n)\\ y\ne\frac{1}{\alpha^\tau}, y\ne 0, y\ne 1}} \chi_1\left(\beta_2\left(\frac{1-\alpha^\tau}{y-1}-\alpha^\tau+1\right)^{-1}-\beta_1\left(\frac{\alpha^\tau(y-1)}{1-\alpha^\tau}\right)\right) \\
&\quad + \frac{1}{N}\chi_1\left(\frac{\beta_1\alpha^\tau}{1-\alpha^\tau}\right) + \frac{1}{N}\chi_1\left(\frac{\beta_2}{1-\alpha^\tau}\right) \\
&= \frac{1}{N}\sum_{\substack{y\in\mathrm{GF}(2^n)\\ y\ne\frac{1}{\alpha^\tau}, y\ne 0, y\ne 1}} \chi_1\left(\frac{\beta_2(y-1)}{(1-\alpha^\tau)y}-\frac{\beta_1\alpha^\tau(y-1)}{1-\alpha^\tau}\right) \\
&\quad + \frac{1}{N}\chi_1\left(\frac{\beta_1\alpha^\tau}{1-\alpha^\tau}\right) + \frac{1}{N}\chi_1\left(\frac{\beta_2}{1-\alpha^\tau}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{N} \sum_{\substack{y\in \mathrm{GF}(2^n) \\ y\neq \frac{1}{\alpha^\tau}, y\neq 0, y\neq 1}} \chi_1\left(\frac{\beta_2+\beta_1\alpha^\tau}{1-\alpha^\tau} - \frac{\beta_2}{(1-\alpha^\tau)y} - \frac{\beta_1\alpha^\tau y}{1-\alpha^\tau}\right) \\
&\quad + \frac{1}{N}\chi_1\left(\frac{\beta_1\alpha^\tau}{1-\alpha^\tau}\right) + \frac{1}{N}\chi_1\left(\frac{\beta_2}{1-\alpha^\tau}\right) \\
&= \frac{1}{N} \sum_{y\in \mathrm{GF}(2^n)^*} \chi_1\left(\frac{\beta_2+\beta_1\alpha^\tau}{1-\alpha^\tau} - \frac{\beta_2}{(1-\alpha^\tau)y} - \frac{\beta_1\alpha^\tau y}{1-\alpha^\tau}\right) \\
&\quad + \frac{1}{N}\chi_1\left(\frac{\beta_1\alpha^\tau}{1-\alpha^\tau}\right) + \frac{1}{N}\chi_1\left(\frac{\beta_2}{1-\alpha^\tau}\right) \\
&\quad - \frac{1}{N}\chi_1\left(\frac{\beta_2+\beta_1\alpha^\tau}{1-\alpha^\tau} - \frac{\beta_2}{(1-\alpha^\tau)\frac{1}{\alpha^\tau}} - \frac{\beta_1\alpha^\tau \frac{1}{\alpha^\tau}}{1-\alpha^\tau}\right) \\
&\quad - \frac{1}{N}\chi_1\left(\frac{\beta_2+\beta_1\alpha^\tau}{1-\alpha^\tau} - \frac{\beta_2}{1-\alpha^\tau} - \frac{\beta_1\alpha^\tau}{1-\alpha^\tau}\right) \\
&= \frac{1}{N} \sum_{y\in \mathrm{GF}(2^n)^*} \chi_1\left(\frac{\beta_2+\beta_1\alpha^\tau}{1-\alpha^\tau}\right)\chi_1\left(-\frac{\beta_2}{(1-\alpha^\tau)y} - \frac{\beta_1\alpha^\tau y}{1-\alpha^\tau}\right) + \frac{1}{N}\chi_1\left(\frac{\beta_1\alpha^\tau}{1-\alpha^\tau}\right) \\
&\quad + \frac{1}{N}\chi_1\left(\frac{\beta_2}{1-\alpha^\tau}\right) - \frac{1}{N}\chi_1(\beta_2-\beta_1) - \frac{1}{N}\chi_1(0).
\end{aligned}
$$

Applying the bound on Kloosterman sum [13], we have

$$
\begin{aligned}
\left|C_{s_2,s_1}(\tau)\right| &\le \frac{1}{N}\left|\sum_{y\in \mathrm{GF}(2^n)^*} \chi_1\left(\frac{\beta_2+\beta_1\alpha^\tau}{1-\alpha^\tau}\right)\chi_1\left(-\frac{\beta_2}{(1-\alpha^\tau)y} - \frac{\beta_1\alpha^\tau y}{1-\alpha^\tau}\right)\right| \\
&\quad + \frac{1}{N}\left|\chi_1\left(\frac{\beta_1\alpha^\tau}{1-\alpha^\tau}\right)\right| \\
&\quad + \frac{1}{N}\left|\chi_1\left(\frac{\beta_2}{1-\alpha^\tau}\right)\right| + \frac{1}{N}\left|-\chi_1(\beta_2-\beta_1)\right| + \frac{1}{N}\left|-\chi_1(0)\right| \\
&\le \frac{1}{N}\left|\sum_{y\in \mathrm{GF}(2^n)^*} \chi_1\left(-\frac{\beta_2}{(1-\alpha^\tau)y} - \frac{\beta_1\alpha^\tau y}{1-\alpha^\tau}\right)\right| + \frac{4}{N} \\
&\le \frac{1}{N}\left(2\sqrt{2^n}+4\right). \qquad (23)
\end{aligned}
$$

When $\tau = 0$ and $\beta_1 \neq \beta_2$, we obtain

$$
\begin{aligned}
C_{s_2,s_1}(0) &= \frac{1}{N}\sum_{t=0}^{N-1}(-1)^{s_2(t)-s_1(t)} \\
&= \frac{1}{N}\sum_{t=0}^{N-1}(-1)^{\mathrm{Tr}_1^n(\beta_2(\alpha^t+1)^{2^n-2})-\mathrm{Tr}_1^n(\beta_1(\alpha^t+1)^{2^n-2})}
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{N}\sum_{t=0}^{N-1}(-1)^{\mathrm{Tr}_1^n\left((\beta_2-\beta_1)(\alpha^t+1)^{2^n-2}\right)} \\
&= \frac{1}{N}\sum_{\substack{y\in \mathrm{GF}(2^n)\\ y\neq 1, y\neq 0}} \chi_{\beta_2-\beta_1}(y) + \frac{1}{N}\chi_{\beta_2-\beta_1}(0) \\
&= \frac{1}{N}\sum_{y\in \mathrm{GF}(2^n)} \chi_{\beta_2-\beta_1}(y) + \frac{1}{N}\chi_{\beta_2-\beta_1}(0) - \frac{1}{N}\chi_{\beta_2-\beta_1}(1) - \frac{1}{N}\chi_{\beta_2-\beta_1}(0) \\
&= 0 - \frac{1}{N}\chi_{\beta_2-\beta_1}(1) \\
&= -\frac{1}{N}(-1)^{\mathrm{Tr}_1^n(\beta_2-\beta_1)}.
\end{aligned}
$$

Thus in this case,

$$\left|C_{s_2,s_1}(0)\right| = \frac{1}{N}. \tag{24}$$

By (23) and (24), we have

$$\left|C_{s_2,s_1}(\tau)\right| \le \frac{1}{N}\left(2\sqrt{2^n}+4\right) \quad \text{for all } 0 \le \tau \le N-1 \text{ and } \beta_1, \beta_2 \in \mathrm{GF}(2^n)^*,$$

where $\tau \neq 0$ if $\beta_1 = \beta_2$. □

Clearly, the number of distinct sequences in S is $|S| = K = N$. By Theorem 4 and mathematical induction, we can easily prove that

$$C_{\max}(S) \le \frac{2\sqrt{N+1}+4}{N} \approx 2\sqrt{\frac{N-1}{N^2-1}} = 2\sqrt{\frac{K-1}{NK-1}}, \tag{25}$$

when $n \ge 20$. Since n is larger than 20 in practice, Design Criterion 4 is satisfied.

In practice, it is difficult to acquire theoretical results of the autocorrelation and cross-correlation profiles of the keystream sequences. However, our experiments show that $C_{\max}^{(L)}(S)$ approaches to $3\sqrt{\frac{K-1}{NK-1}}$ when L is close to N. Figure 5 illustrates the autocorrelation profile of the first period of the keystream sequence s^∞ generated by the keystream generator with $n = 12$, $k = (\alpha^{-3}, \alpha^{-3})$ and $IV = 0$. However, when n is large, i.e., n is more than 64, it is impossible for us to give experimental results of $\max_{1\le\tau\le N-1}|C_{\mathbf{s},\mathbf{s}}^{(L)}(\tau)|$ due to the large computational complexity.

The theoretical and experimental results show that the key approximation attack should not work.

Two encryption keys are called *equivalent* if they define the same encryption transformation. The discussions in this section proved that the cipher described in this paper does not have equivalent keys, while in most ciphers it is open if equivalent

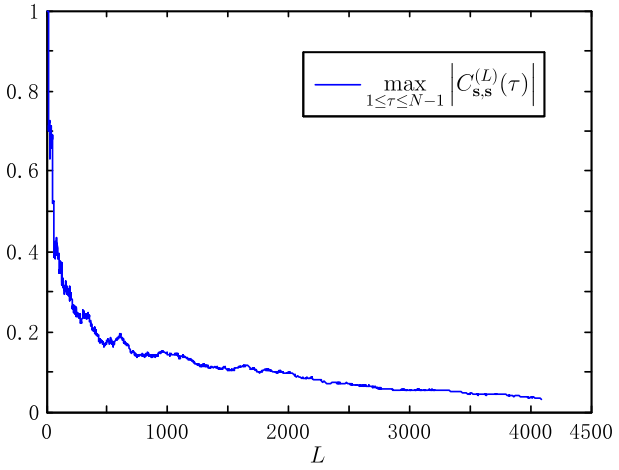

Fig. 5 An autocorrelation profile

keys exist. We are able to prove this security property for this cipher due to the simplicity of its encryption algorithm.

3.4 The security of the stream cipher with respect to the linear approximation attack

We now estimate the Hamming distance between $h(x) = g(f(x)) = \mathrm{Tr}_1^n\left(x^{2^n-2}\right)$ as described in Section 1 and any affine function from GF(2^n) to GF(2). Any affine function from GF(2^n) to GF(2) can be denoted to be $\mathrm{Tr}_1^n(wx) + c$ for some $w \in$ GF(2^n) and $c \in$ GF(2), as described in Section 2.4.

Define

$$H = \sum_{x\in \mathrm{GF}(2^n)} (-1)^{h(x)+\mathrm{Tr}_1^n(wx)+c}.$$

Then the Hamming distance between $h(x)$ and $\mathrm{Tr}_1^n(wx) + c$ is equal to

$$\frac{2^n - H}{2}.$$

Using the similar method in Section 3.3, we have

$$\begin{aligned} H &= \sum_{x\in \mathrm{GF}(2^n)} (-1)^{\mathrm{Tr}_1^n\left(x^{2^n-2}+wx\right)+c} \\ &= (-1)^c \sum_{x\in \mathrm{GF}(2^n)} (-1)^{\mathrm{Tr}_1^n\left(x^{2^n-2}+wx\right)} \\ &= (-1)^c \sum_{x\in \mathrm{GF}(2^n)^*} \chi_1\left(x^{-1}+wx\right) + (-1)^c \chi_1(0). \end{aligned}$$

By the Kloosterman sum theorem [13], we have

$$\begin{aligned} |H| &\le \left| (-1)^c \sum_{x \in \mathrm{GF}(2^n)^*} \chi_1\left(x^{-1} + wx\right) \right| + \left| (-1)^c \chi_1(0) \right| \\ &= \left| \sum_{x \in \mathrm{GF}(2^n)^*} \chi_1\left(x^{-1} + wx\right) \right| + 1 \\ &\le 2\sqrt{2^n} + 1. \end{aligned}$$

Thus $-2\sqrt{2^n} - 1 \le H \le 2\sqrt{2^n} + 1$. Hence

$$2^{n-1} - 2^{n/2} - \frac{1}{2} \le \frac{2^n - H}{2} \le 2^{n-1} + 2^{n/2} + \frac{1}{2},$$

which is close to the covering radius bound [3]. Hence any linear attack on the stream cipher should not work.

Here the lower and upper bounds on H come directly from the well known Kloosterman sum bound. In fact, more information on H can be found in [14].

3.5 The security of the cipher with respect to algebraic attacks

Although algebraic attacks on ciphers have different technical details from case to case, the general idea of any algebraic attack is the following.

1. Find out a set of equations involving plaintext bits, ciphertext bits and secret key bit using the encryption or decryption function.
2. Simplify this set of equations for the easiness of solving them when secret-key bits are treated as unknowns and plaintext bits and ciphertext bits are treated as known constants.
3. Solve the set of simplified equations obtained in Step 2 using given plaintext-ciphertext pairs, aiming at recovering the secret key.

Step 1 is easy to do, as a set of equations could be obtained easily from the encryption or decryption function of the cipher. Any algebraic attack must start from the encryption or decryption function. However, Step 2 above may be hard to carry out as the set of equations obtained in Step 1 may be extremely complex for a cipher with good confusion and diffusion defined by Claude Shannon. In addition, the way for carrying Step 2 may not be unique. Anyway, if a set of simple equations cannot be obtained in Step 2, Step 3 would be infeasible. Clearly, this idea of attacking applies to any cipher. So it should not work for well designed ciphers as the idea is too general.

Now we consider the security of the cipher described in this paper with respect to an algebraic attack. We assume that a segment of the sequence $(\hat{s}(t))$ are known to the adversary. The objective of the adversary is to recover the secret key (a, b).

From the discussion in Section 3.1, the adversary has

$$\begin{aligned}\hat{s}(t) &= \mathrm{Tr}_1^n\left((a\alpha^t+b)^{2^n-2}\right)\\ &= \mathrm{Tr}_1^n\left(\sum_{i=0}^{2^n-2}\binom{2^n-2}{i}(a\alpha^t)^i\, b^{2^n-2-i}\right)\\ &= \mathrm{Tr}_1^n\left(\sum_{i=0}^{2^{n-1}-1}(a\alpha^t)^{2i}\, b^{2^n-2-2i}\right)\\ &= \sum_{i=0}^{2^{n-1}-1}\sum_{j=0}^{n-1}\left(b^{(2^{n-1}-1)2^j}\left(b^{-1}a\alpha^t\right)^{i2^j}\right). \end{aligned} \tag{26}$$

To carry out Step 2, the adversary needs to simplify the expression in (26), which has $n2^{n-1}$ terms. It is possible that some of the $n2^{n-1}$ terms in (26) cancel each other. However, after the cancellation of some terms, the simplified expression of (26) has at least 2^{n-1} terms, as the linear complexity of the keystream sequence is at least 2^{n-1}. Hence, what the adversary has after carrying out Step 2 is a set of equations, where each equation has at least 2^{n-1} terms, given a segment of the keystream sequence. In practice, n should be at least 64. So in each equation, there are at least 2^{63} terms. On the other hand, for the following set of equations

$$\hat{s}(t) = \sum_{i=0}^{2^{n-1}-1}\sum_{j=0}^{n-1}\left(b^{(2^{n-1}-1)2^j}\left(b^{-1}a\alpha^t\right)^{i2^j}\right),\quad t=t_1,t_1+1,\cdots,t_1+L-1, \tag{27}$$

where t_1 and L are some nonnegative integers, it looks hard to find out simple relations among them, so that simpler equations can be derived from the L equations. Hence, the complexity of Step 3 would be too high. It looks that this algebraic attack should not work. Of course, this is an argument, rather than a rigorous proof.

The cipher of this paper makes use of the function $\mathrm{Tr}_1^n(ux^{2^n-2})$, which can be viewed as a Boolean function with n variables when x is expressed as $x=\sum_{i=1}^n x_i\beta_i$ where $\{\beta_1,\beta_2,\cdots,\beta_n\}$ is a basis of GF(2^n) over GF(2). The algebraic immunity of the function $\mathrm{Tr}_1^n(ux^{2^n-2})$ is upper bounded by $2\lfloor\sqrt{n}\rfloor$ [18]. Although this upper bound is not high, it would be hard to carry out an algebraic attack on this cipher using the algebraic immunity of the functions $\mathrm{Tr}_1^n(ux^{2^n-2})$ due to the following:

1. According to [18], it is too complex to determine the algebraic immunity of each function $\mathrm{Tr}_1^n(ux^{2^n-2})$ when $n>25$, while in real applications n should be at least 64.
2. For any two distinct positive integers $0\le t<2^n-1$ and $0\le t'<2^n-1$, let $x=a\alpha^t+b=\sum_{i=1}^n x_i\beta_i$ and $y=a\alpha^{t'}+b=\sum_{i=1}^n y_i\beta_i$. The relation between these x_i and these y_j are nontrivial.

There might be other algebraic attacks. The reader is invited to consider other possible algebraic attacks on this cipher.

3.6 Security against other attacks

A distinguishing attack on a cipher is based on a formal model of security, where an adversary tries to distinguish between the output of a particular cipher and the output of a truly random process, with a non-negligible probability. In the context of additive synchronous stream ciphers, a distinguishing attack tries to distinguish a keystream sequence from a truly random sequence. Any keystream sequence produced by any machine must be different from a truly random sequence as the memory of the machine must be limited. Many properties of the keystream sequences of the cipher dealt with in this paper are very close to those of a truly random sequence. We do not see any distinguishing attack can recover the key of the cipher, and invite the reader to propose such an attack. Greg Rose and Philip Hawkes concluded that the existence of distinguishing attacks against stream ciphers is unrelated to their security in practical use [21].

In a guess-and-determine (GD) attack, the attacker first guesses (the values of) a set of state elements of a cipher, called a basis. The basis can correspond to different elements of different states or key. Next, the attacker determines the remaining state elements or remaining part of the key. If the remaining state elements or part of the key has a solution, then the guessed values are regarded as true; otherwise, the attacker should repeat the above scenario with other guessed values. Thus, the attack complexity is roughly equal to the computation needed for repeating the above scenario for all possible guesses.

For the cipher of this paper, the initial state of the counter is the IV. The attacker may guess a value for a, and try to solve the set of equations

$$\hat{s}(t) = \sum_{i=0}^{2^{n-1}-1} \sum_{j=0}^{n-1} \left(b^{(2^{n-1}-1)2^j} \left(b^{-1} a \alpha^t \right)^{i2^j} \right), \quad t = t_1, t_1 + 1, \cdots, t_1 + L - 1,$$

aiming to solve b. If the set of equations has a solution b, the pair (a, b) is regarded as the true key. Otherwise, the attacker will make another guess for a. However, after guessing a value for a, we do not see any easy way to solve this set of equations. We have the same conclusion when the attacker first guess a value of b and then solve the set of equations. Hence, we do not see any guess-and-determine attack that works on the cipher of this paper.

There might be other attacks on the cipher of this paper. We invite the reader to analyse the security of this cipher.

3.7 The mathematical problem defining the security of this cipher

The security of the cipher of this paper with respect to known-plaintext attacks is equivalent to the difficulty of solving the following set of equations:

$$\mathrm{Tr}_1^n \left(\left(a\alpha^t + b \right)^{2^n - 2} \right) = \hat{s}(t), \quad t = 1, 2, \cdots, 2n, \tag{28}$$

where a and b are unknowns. The cipher is secure with respect to known-plaintext attacks if and only if there is no polynomial-time algorithm for solving this set of equations.

As stated in Section 3.5, (28) can be expressed as (27) with $L = 2n$. Solving (27) with $L = 2n$ should be difficult, due to the reasons given in Section 3.5:

1. Each equation has at least 2^{n-1} terms even if some terms cancel each other, where n is large in practice.
2. It seems hard to find relations among these $2n$ equations.

Also, we can convert (28) to a system of Boolean functions, in order to show the hardness of this problem. The function $\mathrm{Tr}_1^n((a\alpha^t + b)^{2^n-2})$, where a and b are unknowns, can be viewed as a Boolean function with $2n$ variables when a and b are expressed as $a = \sum_{i=1}^{n} a_i\beta_i$ and $b = \sum_{i=1}^{n} b_i\beta_i$ respectively, where $\{\beta_1, \beta_2, \cdots, \beta_n\}$ is a basis of GF(2^n) over GF(2). Then (28) can be expressed as a system of $2n$ Boolean equations with $2n$ variables. According to [11], the general problem of solving a Boolean equation system is computationally hard, and there is no polynomial time solution technique. Since n is usually larger than 64 in practice, solving the system of $2n$ Boolean equations with $2n$ variable should be a computational hard task.

We invite the reader to analyse whether there are efficient methods to solve the system of equations (28).

4 Performance of the stream cipher

If we express every element w in GF(2^n) as $(w_0, w_1, \cdots, w_{n-1}) \in$ GF$(2)^n$, where $w = w_0\alpha^0 + w_1\alpha^1 + \cdots + w_{n-1}\alpha^{n-1}$, then the secret key $k = (a, b)$ of the stream cipher has $2n$ bits. We tested the performance of the stream cipher using the Magma software. The build-in trace function and the inverse function x^{-1} in Magma were used. All experiments were run on a Pentium(R) 4 computer with 3.20GHz and 2.99GB main memory. The encryption or decryption rate of the stream cipher with $n = 127$ was 5.47 kilobytes per second. When $n = 193$, the encryption or decryption rate was 2.04 kilobytes per second. Although its performance is not good, it can be used for offline applications. Since computing the inverse x^{-1} over GF(2^n) is more time-consuming compared with the addition, the subtraction and the multiplication, fast algorithms of computing the multiplicative inverse over GF(2^n) are the keys to improve the performance of the software implementation of the stream cipher. In [10, 12, 24] some algorithms of computing the multiplicative inverse over GF(2^n) are presented. With the implementation of some of these algorithms, the cipher of this paper should have a reasonable performance in software.

We now examine the performance of a hardware implementation of the stream cipher. Reference [20] gives a fast algorithm of computing the multiplicative inverse over GF(2^n), and implements the proposed algorithm in hardware for the computation of x^{-1} over GF(2^{193}). With respect to our stream cipher, we can use the field squarer block given in [20] to compute x^{2^n} over GF(2^{193}), in order to compute the trace function $\sum_{i=0}^{n-1} x^{2^i}$ in hardware efficiently, where $x \in$ GF(2^{193}). We may also directly use the hardware implementation of the inverse function x^{-1} over GF(2^{193}) given in [20]. Since only the timing of the inverse computation is given in [20] and the inversion is the most consuming computation, the encryption or decryption rate of the hardware implementation of the stream cipher with $n = 193$ is expected to be about 100 kilobytes per second, which is about 50 times faster than the speed of the stream cipher given in our software experiment. As there are always trade-offs

between performance and security, the performance is acceptable as we have a cipher with large key size when $n = 193$.

5 The detection of the loss of synchronization of the cyclic counters

Note that the cipher depicted in Fig. 2 is a synchronous stream cipher. Synchronous stream ciphers have the advantage that bit errors in the ciphertext introduced by channel noise during transmission affect only the corresponding bits in the decrypted text. However, the price paid for this strength is that synchronization is required between the sender and receiver for this kind of stream ciphers [5]. Thus the detection of the loss of synchronization and the resynchronization are necessary.

There are different methods for detecting the loss of synchronization. One method may have advantages and disadvantages over another one. A patent of detecting loss of cipher synchronization in a video processing system has been given in [8]. For the cipher of this paper, we propose the following simple method.

We assume that the maximum length of any message to be encrypted is 2^m, where m is a positive integer and is agreed by the sender and receiver. It is safe to assume that $m = 64$. To have the capability of detecting the loss of synchronization, an m-bit block is padded to the original message by the sender. The padded block is the binary representation of the length of the original message to be encrypted. The sender will then encrypt the padded message. After receiving the ciphertext, the receiver will do decryption first. The receiver will then use the last m bits in the decrypted text to compute the length value. The receiver will next count the length of the remaining part in the decrypted text. If the computed length value matches the length of the remaining decrypted text, synchronization is okay. Otherwise, the loss of synchronization is detected and resynchronization is invoked.

6 The resynchronization process

The resynchronization of the cipher in Fig. 2 is to ensure that the counters at the sender and the receiver side have the same internal state at the same time unit. If the loss of synchronization is detected, resynchronization should be initiated so that decryption is correctly done at the receiver's side.

As described in Section 1, the initial content of the memory unit of the counter in Fig. 2 is the IV. At both the sender and receiver side, the resynchronization of the stream cipher depicted in Fig. 2 is conducted as follows:

1. Change the contents of the memory units of the counters at both sides to be IV.
2. Clock the keystream generator n times without changing the secret key. In this way an n-bit keystream $z_0 z_1 \cdots z_{n-1}$ is output by the keystream generators. Now the new IV for resynchronization is $(\sum_{i=0}^{n-1} z_i 2^i) \bmod N$. This new IV is computed by both sides in the same way.
3. Both sides put the newly computed IV into the counter as the current content.

Note that the new IV is computed from the current IV and the secret key using the algorithm of the keystream generator. The resynchronization process does not decrease the security level of the cipher.

7 Conclusions and remarks

In this paper, we analysed possible attacks of a specific binary additive stream cipher based on permutations, and derived a number of design criteria. Experiments of this stream cipher were conducted in the case that no theoretical result is available. A detection of loss of synchronization of the cyclic counters and a resynchronization process are also described so that the stream cipher is ready for applications. Although we are not able to claim the security of the stream cipher based on the properties examined in this paper, the stream cipher is secure with respect to a number of attacks. In addition, this stream cipher is practical for offline applications with high security requirements, as its performance is somewhat inferior to those in the literature. The advantages of the stream cipher presented in this paper are that it is simpler in structure and has more proven security properties. Of course, we are not claiming that the cipher proposed in this paper is secure. In fact, the only cipher with proven security is the one-time pad which is not practical at all. The reader is invited to further analyse the security of cipher.

The motivations for choosing the inverse function x^{2^n-2} for the cipher are the following:

1. It is a one-to-one function so that the keystream sequences are balanced.
2. It has optimal nonlinearity with respect to $(\mathrm{GF}(2^n), +)$ and $(\mathrm{GF}(2), +)$.
3. It guarantees that the linear complexity and its stability of the keystream sequences are good enough.
4. It guarantees that the correlation between any two keystream sequences defined by two distinct keys are good enough.

We tested other known APN permutations and they are not as good as the inverse function x^{2^n-2} considering all the requirements above.

Acknowledgements The authors wish to thank the reviewers for their comments and suggestions that improved the quality of this paper.

References

1. Antweiler, M., Bomer, L.: Complex sequences over $\mathrm{GF}(p^M)$ with a two-level autocorrelation function and a large linear span. IEEE Trans. Inf. Theory **38**, 120–130 (1992)
2. Block cipher modes of operation. http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation. Accessed 10 Apr 2011
3. Carlet, C., Ding, C.: Nonlinearities of S-boxes. Finite Fields Their Appl **13**, 121–135 (2007)
4. Cusick, T., Ding, C., Renvall, A.: Stream Ciphers and Number Theory, Revised edn., vol. 55. The North-Holland Mathematical Library, Elsevier, Amsterdam (1998)
5. Deamen, J., Govaerts, R., Vandewalle, J.: Resynchronization weakness in synchronous stream ciphers. In: Helleseth, T. (ed.) Advances in Cryptology – EUROCRYPT'93. Lecture Notes in Computer Science, vol. 765, pp. 159–167. Springer-Verlag (1993)
6. Ding, C.: Lower bounds on the weight complexity of cascaded binary sequences. In: Seberry, J. (ed.) Proc. of Auscrypt'90, Advances in Cryptology. LNCS 453, pp. 39–43. Springer-Verlag, Heidelberg (1990)
7. Ding, C., Xiao, G., Shan, W.: The stability theory of stream ciphers. Lecture Notes in Computer Science, vol. 561. Springer-Verlag, Heidelberg (1991)
8. Graunke, G.L.: Method and apparatus for detection of loss of cipher synchronization. Patent number: US7369661 (2008)

9. Gupta, K.C., Maitra, S.: Primitive polynomials over GF(2) – a cryptologic approach. In: Qing, S., Okamoto, T., Zhou, J. (eds.) Information and Communications Security. Lecture Notes in Computer Science, vol. 2229, pp. 23–34. Springer-Verlag, Heidelberg
10. Itoh, T., Tsujii, S.: A fast algorithm for computing multiplicative inverses in GF(2^m) using normal basis. Inf. Control **78**, 171–177 (1988)
11. Keinänen, M.: Techniques for solving Boolean equation systems. Research Report A105, Doctoral Dissertation, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland, 3–5 (2006)
12. Li, Y., Chen, G., Chen Y., Li, J.: An extension of TYT inversion algorithm in polynomial basis. Inf. Process. Lett. **110**, 300–303 (2010)
13. Lidl, R., Niederreiter, H.: Finite Fields. Cambridge Univ. Press, Cambridge (1997)
14. Lachaud, G., Wolfmann, J.: The weights of the orthogonals of the extended quadratic binary Goppa codes. IEEE Trans. Inf. Theory **36**, 686–692 (1990)
15. Lucas, E.: Théorie des fonctions numériques simplement périodiques. Am. J. Math. **1**, 229–231 (1878)
16. MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error Correcting Codes. North Holland, Amsterdam (1986)
17. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1996)
18. Nawaz, Y., Gong, G., Gupta, K.C.: Upper bounds on algebraic immunity of boolean power functions. In: Fast Software Encryption, Lecture Notes in Computer Science, vol. 4047, pp. 375–389. Springer-Verlag, Berlin (2006)
19. Pless, V.S., Huffman, W.C., Brualdi, R.A.: An introduction to algebraic codes. In: Pless, V.S., Huffman, W.C. (eds.) Handbook of Coding Theory, pp. 3–139. Elsevier, Amsterdam (1998)
20. Rodríguez-Henríquez, F., Morales-Luna, G., Saqib, N.A., Cruz-Cortés, N.: Parallel Itoh-Tsujii multiplicative inversion algorithm for a special class of trinomials. Des. Codes Cryptography **45**, 19–37 (2007)
21. Rose, G., Hawkes, P.: On the applicability of distinguishing attacks against stream ciphers. In: Proceedings of the 3rd NESSIE Workshop (2002)
22. Rueppel, R.A.: Analysis and Design of Stream Ciphers. Springer-Verlag (1986)
23. Stamp, M., Martin, C.F.: An algorithm for the k-error linear complexity of binary sequences with period 2^n. IEEE Trans. Inf. Theory **39**, 1398–1401 (1993)
24. Takagi, N., Yoshiki, J., Takagi, K.: A fast algorithm for multiplicative inversion in GF(2^m) using normal basis. IEEE Trans. Comput. **50**, 394–398 (2001)
25. Welch, L.R.: Lower bounds on the maximum cross correlation of signals. IEEE Trans. Inf. Theory **20**, 397–399 (1974)