2SC: an Efficient Code-based Stream Cipher

Mohammed Meziani, Pierre-Louis Cayrel, and Sidi Mohamed El Yousfi Alaoui

CASED – Center for Advanced Security Research Darmstadt,
Mornewegstr. 32, 64289 Darmstadt, Germany
{mohammed.meziani,pierre-louis.cayrel,elyousfi}@cased.de

Abstract. In this article, we present a new code-based stream cipher called 2SC, based on the sponge construction. The security of the keystream generation of 2SC is reducible to the conjectured intractability of the Syndrome Decoding (SD) problem, which is believed to be hard in the average case. Our stream cipher compares favorably with other provably secure stream ciphers such as QUAD and SYND in terms of efficiency and storage. In particular, 2SC is much faster than both these stream ciphers, requiring shorter keys and initial vectors (IVs) in order to attain comparable security levels (the runtime in terms of clock cycles is actually halved compared to SYND for around 170 bits of security, whereas the key size is about 50 bits smaller) .

Keywords: Stream ciphers, Provable security, Syndrome decoding.

1 Introduction

Stream ciphers and block ciphers are two very important families of symmetric encryption schemes. The former typically operates on smaller units of plaintext one at a time, usually bits, while the latter operates on large blocks of the plaintext message at the same time. Stream ciphers have to be exceptionally fast, much faster than any block cipher, and require lower computing resources. For this reason, they are used in many applications such as secure wireless communication. A stream cipher produces what is called a *keystream*, which is a sequence of bits used as a key for encryption. The ciphertext is obtained by combining the keystream with the clear text (plaintext) on a 'bit by bit' basis, usually by mean of the bitwise XOR operation.

Over the last decades, the design of stream ciphers, and more generally, pseudo-random number generators (PRNGs), has been a subject of intensive study. There exist several constructions, most of them based on linear feedback shift registers (LFSR). However, state of the art of cryptanalysis of such ciphers showed that they suffer from many security weaknesses. Another method to construct a PRNG is to use keyed pseudo-random permutation. The first provably secure construction following this approach is due to Blum and Micali [6] whose security is based on the hardness of factoring an RSA modulus. Another PRNG was proposed by Blum, Blum, and Shub [5] is secure under the assumption that factoring large Blum integers is hard. For the modified of the Blum-Micali construction developed by Kaliski [18], the corresponding hardness is the intractability of the discrete logarithm problem in the group of points on an elliptic curve defined over a finite field. Assuming the assumption of the RSA problem , Alexi, Chor, Goldreich and Schnorr [1] proposed a PRNG. The one-way function hard-core bit construction by Goldreich and Levin [13] has also led to the construction of the efficient pseudo-random number generator, called BMGL [22], which was developed by Håstad and Näslund using Rijndael.

Unfortunately, the theoretically secure constructions of PRNG have up to now mostly focused on methods based on number theoretic assumptions. The existing schemes are vulnerable to "quantum" attacks as shown by Shor [21]. Furthermore, despite their simplicity, most of these systems are inefficient thus impractical for many applications. It is therefore desirable to have efficient stream ciphers whose security relies on other assumptions, and which are more promising even for a future featuring quantum computers. The first construction addressing this challenge is due to Impagliazzo and Naor [17] is based on the subset sum problem. Later, Fisher and Stern [9] proposed a PRNG system whose security relies on the syndrome decoding problem for error-correcting codes. Recently, two provably secure PRNG constructions have proposed. The first one, called QUAD, due to Berbain, Gilbert and Patarin [3] under assumption that solving a multivariate quadratic system is difficult (MQ-problem). The second one, named SYND and proposed by Gaborit, Lauradoux and Sendrier [12], is an improved variant of Fisher-Stern's system [9]. The security of SYND is reducible to the SD problem, which has been proved to be NP-complete in [4].

Our contribution

In this paper we describe a new code-based stream cipher whose security can be reduced to the SD problem. This cipher is faster than the last one proposed in [12] and requires comparatively little storage capacity, making it attractive for practical implementations. We also propose parameters for fast keystream generation for different security levels.

Organization of the paper

Section 2 provides a short introduction to error-correcting codes. Section 3 gives a brief review of related work. A detailed description of the of 2SC stream cipher is presented in Section 4, its security is discussed in Section 5. In Section 6 secure parameters and experimental results for 2SC are presented. Section 7 concludes the paper and gives some suggestions for future research.

2 Preliminaries

In this section we provide a short introduction to error-correcting codes and recall some hard problems in this area. A more detailed description can be found in [20].

In general, a linear code $\mathcal{C}$ is a k-dimensional subspace of an n-dimensional vector space over a finite field $\mathbb{F}_q$, where k and n are positive integers with $k < n$ and q a prime power. Elements of $\mathbb{F}_q^n$ are called words and elements of $\mathcal{C}$ are called codewords. The integer $r = n - k$ is called the co-dimension of $\mathcal{C}$. The weight of a word x, denoted by $w = wt(x)$, is the number of non-zero entries in x, and the Hamming distance between two words x and y is $wt(x - y)$. The minimum distance d of a code is the smallest distance between any two distinct codewords. If the ratio n/w is a power of 2, we say words are regular if they consist of w blocks of n/w bits, each with a single non-zero entry. A generator matrix G of $\mathcal{C}$ is a matrix whose rows form a basis of $\mathcal{C}$. i.e., $\mathcal{C} = \{xG : x \in \mathbb{F}_q^k\}$. A parity check matrix H of $\mathcal{C}$ is defined by $\mathcal{C} = \{x \in \mathbb{F}_q^n : Hx^T = 0\}$ and generates the code's dual space.
Throughout this paper we set $q = 2$.

The security of code-based cryptosystems is based on the hardness of several classical coding theory problems. The most relevant in our context are the following:

Definition 1 (Binary Syndrome Decoding (SD) problem). *Let n, s, and w be positive integers with $n > s > w$. Given a binary $s \times n$ matrix H, a binary vector $y \in \mathbb{F}_2^s$,and an integer $w > 0$, find a word $x \in \mathbb{F}_2^n$ of weight w, such that $H \cdot x^T = y$.*

This problem was proved NP-complete in [4]. A particular case of SD is the Regular Syndrome Decoding problem (RSD), which was defined and proved NP-complete in [2]. This problen is stated as follows.

Definition 2 (Regular Syndrome Decoding problem (RSD)). *Let n, s, and w be positive integers with $n > s > w$. Given a binary $s \times n$ matrix H, a binary vector $y \in \mathbb{F}_2^s$, and an integer $w > 0$, find a regular word $x \in \mathbb{F}_2^n$ of weight w, such that $H \cdot x^T = y$.*

Through this paper, we will denote SD(n, s, w) and RSD(n, s, w) to indicate instances of the above problems with parameters (n, s, w).

3 Related works

In this section, we briefly review related work for our construction: The SYND stream cipher [12] and the sponge construction [10].

SYND stream cipher. SYND is a improved variant of Fisher-Stern's PRNG [9] with two improvements: the use of quasi-cyclic codes, which reduces the storage capacity and the introduction of the regular words technique used in [2], which speeds up the keystream generation of the system. SYND is a synchronous stream cipher, which uses a secret key and an initial value, both of length r (in bits). THe SYND cipher has three phases: initialization, update, and output. During each step, an r-to-r bit function is applied. The update and output function, denoted respectively by g_1 and g_2, are defined as follows:

$$\begin{aligned} g_i : \mathbb{F}_2^r &\rightarrow \mathbb{F}_2^r \qquad (i = 1, 2) \\ x &\mapsto g_i(x) = H_i \cdot \phi(x)^T, \end{aligned}$$

where the mapping $x \mapsto \phi(x)$ is an encoder which transforms a bitstring of length r into a regular word of length n and weight w, and each H_i is an $r \times n_i$ random matrix. The three round Feistel transformation using g_1 and g_2 is the initialization function, taking as input an initial value IV and a secret K, both of them $r/2$ bits long, and returning as output an initial state of size r. The security of SYND is reduced to a special case of the syndrome decoding problem, having solutions from the space of regular words (See [12] for more details).

Sponge functions. Sponge functions were introduced in [10]. They provide a new method for iterative hash function design, called the sponge construction. A sponge function takes as input a variable-length bit string and outputs a bit string of unspecified length. It is determined by a fixed-length transformation (or permutation), operating on states of fixed size b (in bits). The value b is called width. Each state is composed of two parts: the first r bits are its outer part and the last c bits are its inner part with $b = r + c$. r is called the bitrate and c is the capacity of the construction. Furthermore, all bits of the state are initialized to 0. Denote by $e = (e_r, e_c)$ a state of the sponge construction, where e_r is the outer part and e_r the inner part of e. The function $\mathcal{F}$ is evaluated in two steps, called the absorbing, and resp. the squeezing phase.

During the former phase, for an r-bits input x, the state is updated as $e \leftarrow \mathcal{F}(e_r \oplus x, e_c)$. In the squeezing phase, only the first parts of states are returned as blocks z_i of the infinite output bit string, i.e. $z_i \leftarrow e_r$; this operation is followed by the state update as $e \leftarrow \mathcal{F}(e)$. The number of output blocks is chosen by user.

Note that the inner part e_c of each state e are never directly modified by inputs and never returned during the squeezing phase. The desired security level of the construction is determined by the capacity c.

For the security analysis of the sponge construction, there are two cases to consider, keyed and resp. unkeyed sponge constructions. The first setting corresponds to scenarios where a sponge function is used in conjunction with a key, for example in the case of stream ciphers and message authentication codes (MACs). It was proved in [11] that the advantage in distinguishing a keyed sponge from a random oracle is upper bounded by max$\{\left(\left(M^2/2 + 2MN\right) 2^{-c}, N2^{-|K|}\right)\}$, where M is the data complexity, i.e. the amount of access to the keyed instance, N the number of queries to the underlying transformation (or permutation), where only "fresh" queries are considered. "Denote by $|K|$ the length of the key, and let c be the capacity. In the unkeyed setting, for example the case of hash functions, this bound this bound is larger, and equaling to $N(N+1)/2^{c+1}$ when $2^c >> N$ (See [11] for more details).

4 Our Proposal: 2SC

This section describes in detail a novel construction, called Sponge Code-based Stream Cipher (2SC). We outline the construction starting from the illustration in Figure 1. Let `K` and `IV` denote the key and the initial vector respectively, and let $r = |\texttt{K}| = |\texttt{IV}|r$. Denote $s = w \log 2(n/w)$, where $n > w$ and the ratio n/w is power of 2.

Initialization. The initialization function f takes a key $\mathtt{K}$ and an initial vector $\mathtt{IV}$ and returns an initial state as follows:

$$\begin{aligned} f : \mathbb{F}_2^{|\mathtt{K}|} \times \mathbb{F}_2^{|\mathtt{IV}|} &\to \mathbb{F}_2^s \\ (x_1, x_2) \mapsto f(x_1, x_2) &= f_1\left((f_1^{[r]}(x_1|0^c) \oplus x_2, f_1^{[c]}(x_1|0^c)) \right), \end{aligned}$$

where "|" denotes the concatenation and "0^l" is the all-zero vector of size l. We write $f_1^{[r]}(\cdot)$ and resp. $f_1^{[c]}(\cdot)$ for the outer, resp. inner part of $f_1(\cdot)$. The syndrome mapping f_1 is defined by:

$$\begin{aligned} f_1 : \mathbb{F}_2^s &\to \mathbb{F}_2^s \\ x \mapsto f_1(x) &= H_1 \cdot \phi(x)^T. \end{aligned}$$

Here, the function $x \mapsto \phi(x)$ is a regular encoder, transforming a an s-bit string into a regular word of length n and weight w. The matrix H_1 is a random binary matrix of size $s \times n$. This initialization step requires two function evaluations and two bitwise XOR operations. Thus, if optimal parameters (n, s, w) are chosen for each security level, our proposal is more efficient than SYND (see also Section 6).

Update. During this step, an additional random function g is used to update the internal state several times (say N times). The number of times that g is run is chosen by the user, affecting both the security and the efficiency of the construction. The function g is defined by:

$$g : \mathbb{F}_2^s \to \mathbb{F}_2^s \quad (1)$$

$$x \mapsto g(x) = H_2 \cdot \phi(x)^T, \quad (2)$$

where H_2 is a binary random matrix of size $s \times n$ and the function $x \mapsto \phi(x)$ is the regular encoder used in the initialization step above.

Squeezing. Let e_N be the internal state output by the update phase. The cipher 2SC generates a keystream consisting of r bit blocks $(z_i)_{i \geq 1}$ as follows:

- z_1 consists of the first r bits of the internal state $x_1 = g(e_N)$, where e_N is the output of the N iterations of g in the update state.
- For $i \geq 2$, z_i consists of the first r bits of the recursively computed state $x_i = g(x_{i-1})$.

Encryption. Let L be the maximal keystream that may produced using a single $(\mathtt{K}, \mathtt{IV})$ pair. In our construction, we have $L = rN_g$, where N_g indicates the maximal number of calls of the function g. In order to encrypt a clear text of length $l \leq L$, each of the first l bits of the keystream sequence is XOR-ed with the corresponding clear text value as in the one-time pad encryption.

In practice the parameters n, s, and w are chosen such that $\frac{n}{w} = 2^m$ and $s = w \times m$ for some integers $m > 0$. As stated in [2], this choice is due to the following consideration: the number of regular words is exactly equal to $(\frac{n}{w})^w$, thus our choice of n, s, and w implies the existence a simple regular encoder between $\mathbb{F}_2^{wm}$ and the set of regular words. In 2SC, we use the regular encoding algorithm introduced in [7] to speed-up the computation. The main idea of this algorithm is to compute the w indexes where the non-zero entries of a regular word are located. These indexes exactly correspond to the w columns used in the XOR operations.

5 Security Analysis of 2SC

This section assesses the security of 2SC. We first show that, in practice, it is hard to recover states or reconstruct the secret data (Key and IV) from the key stream. We then show that the output of 2SC is pseudo-random, i.e., the probability to distinguish the key stream output by 2SC from a random sequence is negligible.

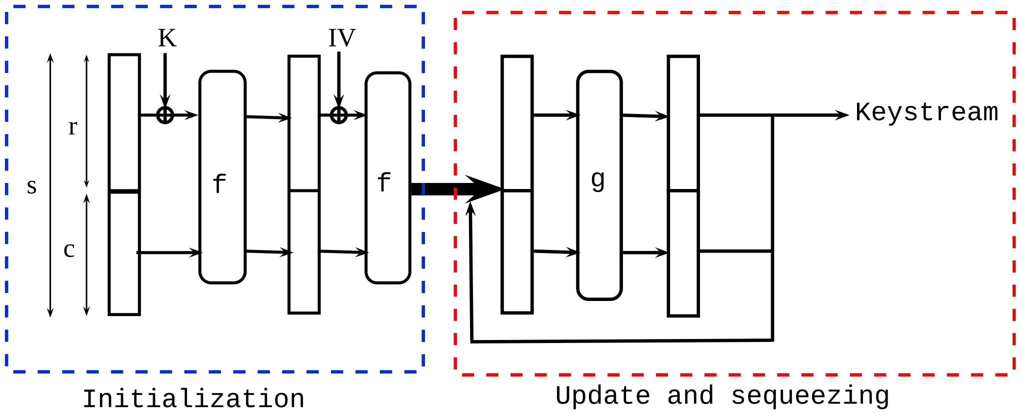

Fig. 1. A Diagram of 2SC Key Stream Generator

5.1 Best known attacks

In practice, an adversary against the security of 2SC is faced with two problems. On the one hand, knowing the blocks z_i of r bits does not allow an adversary to get the remaining $c = b - r$ bits; the larger the capacity, the more secure the system is. On the other hand, even having successfully guessed those bits, the adversary must solve an instance of the RSD problem. However, solving the RSD problem efficiently is as difficult as SD in average case, for an appropriately chosen parameter set. Indeed, all known attacks for SD are fully exponential; in fact, only two kinds of algorithms can attack the SD-based systems: Information Set Decoding (ISD) and the Generalized Birthday Algorithm (GBA). Which of the two approaches is more efficient depends on the parameters and the cryptosystem. In our setting, each instance of RSD has on average one solution due to the form of the regular words; here the best known attack is the GBA, as shown in [8]. The most recent GBA against code-based crytosystems is proposed in [8] and will be used to select secure parameters for 2SC.

Remark 1. One could also use Time Memory trade-off attacks against stream ciphers. This attack was first introduced in [15] as a generic method of attacking block ciphers. To avoid it, one must adjust the cipher parameters as shown in [16,14], i.e., the IV should be at least as large as the key, and the state should be at least twice the key.

5.2 Pseudorandomess

In this section, we analyze the security of the key stream generation as well as the security of the initialization process. We first show that the key stream produced by our scheme is indistinguishable from a random sequence. Towards achieving this goal, we first define several useful concepts.

Definition 3 (Universal Hashing). *A family $\mathcal{U}$ of hash functions $u \in \mathcal{U}$ with $u : X \to Y$ is called universal if for all $x \neq x'$ we have*

$$Prob[u(x) = u(x') \mid u \text{ sampled randomly from } \mathcal{U}] \leq \frac{1}{b},$$

where $x, x' \in X$ and $b = |Y|$.

Next, we introduce the Subset Sum Problem (SSP), which is closely related to syndrome decoding problem. The SSP has been proved NP-complete by Karp in [19] and can be stated as follows.

Definition 4 (Subset Sum Problem (SSP)). *Given n integers $(h_1, \cdots, h_n)$, each of s bits, and and an integer y called the target, find a subset $S \subset \{1, \cdots, n\}$ such that $\sum_{j \in S} h_j = y \mod 2^s$.*

As stated in [17], this problem is equivalent to inverting the function

$$g_h(S) = \sum_{j \in S} h_j = y \mod 2^s. \tag{3}$$

This function maps an n-bit string to s-bit string. When the cardinality $|S|$ of S is upper bounded by a fixed integer w (i.e. $|S| \leq w$), we get an instance of the (regular) syndrome decoding stated earlier. More precisely, take $|S| = w$; then the elements in S can be interpreted as the positions of the non-zero coordinates of an incidence vector x. Thus x has weight $|S| = w$. The elements $(h_1, \cdots, h_n)$ are the rows of a matrix H of size $s \times n$. The target y is the syndrome such that $H \cdot x^T = y$.

Without loss of generality, the transformation f in the initialization step (see section 4) and the equivalent transformation g can be regarded as a mapping $x \mapsto u(x) = H \cdot x^T$, x is a regular word, because the encoding function ϕ (as defined in the initialization and update steps outlined in section 4) is bijective.

Let $\mathcal{R}$ denote the set of regular words of length n and weight w and $\mathcal{H}$ the set of binary random matrices of size $s \times n$. In order to prove that the family $\mathcal{U} = \{u : u(x) = H \cdot x^T, x \in \mathcal{R}, H \in \mathcal{H}\}$ is universal, we state the following lemma.

Lemma 1. *There exists, on average, only one solution of each instance RSD(n, s, w), where $s = w \log 2(n/w)$.*

Proof. Let $N_{rsd}(n, s, w)$ denotes the expected number of solutions of an instance RSD(n, s, w). This number is defined as the number of regular words divided by the number of the syndromes, i.e. $N_{rsd}(n, s, w) = \frac{\left(\frac{n}{w}\right)^w}{2^s}$. By replacing s by $w \log 2(n/w)$, we obtain $N_{rsd}(n, s, w) = 1$.

Proposition 1. *The family $\mathcal{U} = \{u : u(x) = H \cdot x^T, x \in \mathcal{R}, H \in \mathcal{H}\}$ is universal.*

Proof. From Lemma 1, we know that there exists on average only one regular word that solves the syndrome decoding problem. Thus, it follows that for all for all $x \neq x'$

$$\text{Prob}[H \cdot x^T = H \cdot {x'}^T \mid H \text{ sampled randomly from } \mathcal{H}] = 0 \leq \frac{1}{2^s}$$

Proposition 2. *The probability of distinguishing the output of each $u \in \mathcal{U}$ from a random sequence of length s is negligible.*

Proof. (Sketch). The proof is inspired from [17] and works as follows. As explained above, the family $\mathcal{U}$ can be seen as a collection g_h defined as in equation (3). Due to Proposition 1 this collection of transformations is universal and therefore, as proved in [17], we can apply the Leftover hash lemma to show that if $s < \gamma n$ for some real number $\gamma < 1$, then the expected distinguishibility of $g_h(S) = H \cdot x^T$ and a random $y \in \mathbb{F}_2^s$ is at most $2^{\frac{-(1-\gamma)n}{2}}$.
In our setting, γ can be obtained as follows:

$$\psi(n, w) = \frac{s}{n} = \frac{\log_2(n/w)}{(n/w)}.$$

For simplicity, we can assume that $n > 4w$. In this case, the function ψ tends to zero when n is chosen to be large enough. Consequently, there exists an n_0 such that for all $n \geq n_0$, $\psi(n, w)$ is upper bounded by a constant $\gamma < 1/2$. Thus, for values of the code length n such that $\frac{(1-\gamma)n}{2}$ is large enough, the probability of distinguishing the output of any function $u \in \mathcal{U}$ from a random sequence of length s is negligible.

Security of the key stream generation. As explained above, the keystream output by 2SC consists of a number of blocks of length $s - c$ bits (the number of blocks depends on the number of times the squeezing operation is performed). Each block is obtained from the output of the update function g, by extracting its first $s - c$ bits. Since g is similar to the output function used in SYND [12], it is straightforward to prove that the keystream generated by our scheme is indistinguishable from a random sequence under the assumption that the regular syndrome decoding problem is intractable for some sets of parameters. The proof is inspired from [12,9] based on the proofs given in [3]. The following theorem summarizes our main result.

Theorem 1. *Let $L = \alpha(s - c)$ represent the number of keystream bits produced by our scheme after iterating the update function g α times. If there exists an algorithm A, in time T and with advantage ϵ, which distinguishes the L-bit keystream sequence produced through a known random $s \times n$ matrix (or a quasi-cyclic random matrix of the same size) multiplied by an unknown, randomly chosen regular word e from a random bit sequence, then there exists an algorithm $\mathcal{A}'$ that can recover e in time $T' \approx \frac{2^7 n^2 \alpha^2 T}{\epsilon^2}$.*

Proof. (Sketch) As in [3], the proof sketch consists of three steps. First we prove that distinguishing the keystream from a random sequence is equivalent to distinguishing the output of a random syndrome mapping $x \mapsto H \cdot x^T$ for an unknown, regular n-bit word x, from a random vector of appropriate size. Then, we prove that a distinguisher against the syndrome map can be used to build a predictor for any linear function of its inputs. Finally, we show that such a predictor allows us to invert the syndrome mapping for regular words.

Security of the initial state. As explained in Section 4, the initialization process of the 2SC consists of two stages. During the first stage, the secret key is introduced to generate a pre-initial state. For suitably chosen parameters (n, s, w), as indicated in [12], the underlying syndrome mapping behaves like a random function, since its outputs are indistinguishable from a random sequence. Therefore, the pre-initial state is random. During the second stage, the outer r-bit part of this state is first XORed with a secret initial value. Then, the resulting s-bit vector is fed to the function f to produce the initial state. This process can be viewed as XORing w random columns of a random matrix, resulting in a random s-bit initial state.

6 Parameters and Implementation Results

Suitable parameters (n, s, w) for 2SC should provide both efficiency and high security against all known attacks. Firstly, we account for Time Memory Trade-Off attacks (see section 5.1) and choose (n, s, w) such that:

$$s = w \log_2(n/w) \geq 2|\mathtt{IV}| \quad \text{and} \quad |\mathtt{IV}| \geq |\mathtt{K}|.$$

Since $|\mathtt{IV}| = |\mathtt{K}| = s - c$, we obtain $s \leq 2c$. We use the following strategy for selecting secure parameters for 2SC: according the sponge construction, we first fix c such that $c/2$ is at least the desired security level, then choose the remaining parameters (n, s, w) accordingly.

We have implemented 2SC to test a large set of potential parameters for a number of security levels. In practice, optimal parameters for this scheme should also take into account these three main implementation-specific requirements: the ratio $\frac{s}{c}$, selecting an appropriate block size for the regular encoding, and the use of int-wise (rather than byte-wise) XORing. A large value of $\frac{s}{c}$ yields a large value of r, hence allowing for better performance. We implement the regular encoding such that it uses shift operations, thus efficiently using processor architecture. The choice $\log_2(n/w) = 16$ was the most promising block size in terms of the computation time in our implementation. Finally, int-wise XORing reduces computation time by four times compared to byte-wise XORing. Our parameters should thus ensure that we can perform int-wise XORing.

Putting everything together, the choice of w, s and c is a tradeoff decision. On the one hand, a small w leads to fewer XOR operations during matrix multiplication. On the other hand, a small w implies a small s ($s = w \log_2(n/w)$). Making n large will help in increasing s. But at the same time the matrix will become very big. Last but not least, the smaller c is chosen, the more efficient the computation is, because $(r = s - c)$

becomes larger.

Table 1 presents the optimal parameter sets (n, w, s, c) resulted from running our implementation for three security levels, 100, 160,and 250 bits. All results are tested on 2,53 GHz Pentium Core2 Duo, 32 Bit Ubuntu 10.04, 6 MB Level 2 Cache, 4 GB RAM.

Security Level	n	s	w	c	Key/IV size	Speed (cycles/byte)
100	1572864	384	24	240	144	37
160	2228224	544	34	336	208	47
250	3801088	928	58	576	352	72

Table 1. Performance of 2SC using quasi-cyclic codes

In order to compare the speed of 2SC with the speed of SYND [12], we have implemented SYND with the same techniques using the parameter sets proposed in [12].

Note that the results given in [12] can not be checked, since no freely-available implementation of SYND exists. For this reason, we decided to implement SYND. On our own implementation of SYND, we obtained the results presented in Table 2. For comparable security levels, 2SC runs faster than SYND. At the same time, SYND needs significantly larger key sizes compared to 2SC. However, 2SC suffers from the drawback of having to store large matrices.

Security Level	n	s	w	Key/IV size	Speed (cycles/byte)
80	8192	256	32	128	36
170	8192	512	64	256	85
366	8192	1024	128	512	132

Table 2. Performance of SYND using quasi-cyclic codes

7 Conclusion and Future Work

This paper describes a new code-based stream cipher, called 2SC. Its design follows the sponge construction [10]. Our proposal is more efficient than the SYND and QUAD stream ciphers and its security is reducible to the syndrome decoding problem. Moreover, 2SC uses small key/IV sizes compared to SYND and QUAD. However, our construction is slower than AES in counter mode and needs a larger storage capacity. We believe that these two issues could be considerably reduced by avoiding the use of the regular encoding, which is the most time consuming part during the keystream generation. A promising alternative to the regular encoding and matrix-vector multiplication step is to introduce a one-way mapping based only on XOR operations.

References

1. W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. Rsa and rabin functions: certain parts are as hard as the whole. *SIAM J. Comput.*, 17(2):194–209, 1988.
2. D. Augot, M. Finiasz, and N. Sendrier. A Family of Fast Syndrome Based Cryptographic Hash Functions. In E. Dawson and S. Vaudenay, editors, *Mycrypt 2005*, volume 3715, pages 64–83. Springer, 2005.

3. C. Berbain, H. Gilbert, and J. Patarin. Quad: A multivariate stream cipher with provable security. *J. Symb. Comput.*, 44(12):1703–1723, 2009.
4. E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(2):384–386, May 1978.
5. L Blum, M Blum, and M Shub. A simple unpredictable pseudo random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.
6. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.
7. M. Finiasz, P. Gaborit, and N. Sendrier. Improved fast syndrome based cryptographic hash functions. In V. Rijmen, editor, *ECRYPT Hash Workshop 2007*, 2007.
8. M. Finiasz and N. Sendrier. Security Bounds for the Design of Code-based Cryptosystems. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, number 5912 in LNCS, pages 88–105. Springer, 2009.
9. J.-B. Fischer and J. Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In *EUROCRYPT'96: Proc. of the 15th annual international conference on Theory and application of cryptographic techniques*, pages 245–255. Springer, 1996.
10. M. Peeters G. Bertoni, J. Daemen and G. Van Assche. Sponge Functions. In *ECRYPT Hash Workshop 2007*, 2007.
11. M. Peeters G. Bertoni, J. Daemen and G. Van Assche. On the security of the keyed sponge construction. In *Symmetric Key Encryption Workshop (SKEW) 2011*, 2011.
12. Ph. Gaborit, C. Laudaroux, and N. Sendrier. Synd: a very fast code-based cipher stream with a security reduction. In *IEEE Conference, ISIT'07*, pages 186–190, Nice, France, July 2007.
13. O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC '89: Proc. of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989.
14. J.Dj. Golic. Cryptanalysis of alleged a5 stream cipher. In *Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'97, pages 239–255. Springer, 1997.
15. M. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26:401–406, 1980.
16. J. Hong and P. Sarkar. Rediscovery of time memory tradeoffs. Cryptology ePrint Archive, Report 2005/090, 2005. `http://eprint.iacr.org/`.
17. R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptology*, 9(4):199–216, 1996.
18. B. S. Kaliski. *Elliptic Curves and Cryptography: A Pseudorandom Bit Generator and Other Tools*. Phd thesis, MIT, Cambridge, MA, USA, 1988.
19. R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*. Plenum Press, 1972.
20. F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error Correcting Codes*. North-Holland, 1977.
21. P. W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *SFCS '94: Proc. of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society, 1994.
22. J. Håstad and M. Nä slund. Bmgl: Synchronous key-stream generator with provable security, 2001.