American Journal of Applied Sciences 5 (7): 896-904, 2008
ISSN 1546-9239
© 2008 Science Publications

DSP-128: Stream Cipher Based On Discrete Log Problem And Polynomial Arithmetic

Khaled M. Suwais and Azman Samsudin
School of Computer Sciences, Universiti Sains Malaysia (USM), 11800 Penang, Malaysia

Abstract: DSP-128 is a new multithreaded stream cipher based on the intractability of the Discrete Logarithm Problem (DLP) with key size of 128-bit. The design of DSP-128 is divided into three stages: Initialization Stage, Keystream Generation Stage, and Encryption Stage. The design goal of DSP-128 is to come up with a secure stream cipher with good performance for data encryption. The experimental results show that the encryption rate of DSP-128 is one time slower (running on single processor) than the widely adapted stream cipher RC4, with a higher level of security against possible cryptanalysis attacks. However, because of its multithreaded nature, DSP-128 can take the speed up advantage of multi-core processor architectures which are available widely.

Key words: Discrete logarithm problem, cryptography, stream cipher, polynomial, multithreading

INTRODUCTION

Cryptography, in modern times, is an intersection branch between Computer Science and Mathematics. Tremendous researches have done by mathematicians in order to find mathematical problems that can be used in cryptographic applications. Computer scientists in turn are trying to make use of those mathematical problems in order to develop new cryptographic primitives that are secure against cryptanalysis attacks. The main purpose of any development in cryptography is to provide secure communication over public and unsecured communication channels. Therefore, in this paper we are interested in applying mathematical hard problem into one class of the Symmetric Key cryptosystems, which is known as Stream Cipher.

Currently most of the widely adapted stream ciphers are based on some logical and mathematical operations (e.g. XOR, shift, rotation, etc) while others are based on the use of Linear Feedback Shift Registers (LFSR). Recent research [1] shows that the majority of current qualified stream ciphers of eSTREAM project are based on LFSR or on Non-Linear Combining/Filtering Functions. The work presented in this paper is introducing a novel work by providing a multithreaded stream cipher based on DLP which is a mathematical Non-deterministic Polynomial (NP)-hard problem [2].

The idea behind our proposed stream cipher is based on the intractability of solving the DLP founded on polynomial arithmetic. DLP has been applied effectively in different areas in cryptography such as Key Exchange[3], Digital Signatures[4], and asymmetric encryption algorithms[5]. The DLP applies to a collection of elements together with a binary operation (group G). The notion of DLP rests on the difficulty of finding integer x such that $z^X = y \bmod p$ where z, y are two elements in finite group G, and p is a prime number. In fact, the reason of choosing DLP as a core for our stream cipher rests on the belief that DLP is providing a high security cipher since there is no known polynomial time algorithm for solving this problem.

The DLP of DSP-128 is founded on polynomial arithmetic, whereas the elements of the finite filed G are represented in polynomial representations. The original DLP implies a prime number for its module operation, and the same technique is used in DSP-128 but considering an *irreducible* (prime) polynomial instead of an integer prime number. The following sections will describe our multithreaded design in detail and illustrate the benefits of using DLP in term of the security provided, as well as the achieved performance by representing the underlying DLP in polynomial representation.

CURRENT STREAM CIPHERS DESIGNS

In contrast to block cipher, stream cipher has no standard model for its construction design, which leads cryptographers to construct various models of stream ciphers. The basic structures often found on stream ciphers may include LFSR-based cipher, NLFSR-based cipher, Block Cipher-based Stream cipher, T-Function-based cipher and other alternative designs

Corresponding Author: Azman Samsudin, School of Computer Sciences, Universiti Sains Malaysia (USM), 11800, Penang, Malaysia, Tel: +604-6533888 Fax: +604-6573335

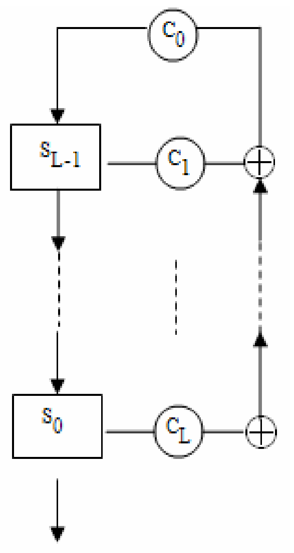


Fig.1: General Design of LFSR of L length

which may not fit the above categories such as RC4 and SEAL. In the following sections we will briefly discuss the properties of the major categories of stream ciphers. The reason for discussing these categories is that current stream ciphers being evaluated in the third phase of eSTREAM project are generally part of the categories mentioned above. On the other hand, we will highlight RC4 since it is a widely adapted stream cipher and it is used in many protocols (e.g. Secure Socket Layer Protocol (SSL)[6]).

Linear Feedback Shift Register (LFSR) based stream cipher: LFSR is a hardware device made up by registers, which is capable of holding one value at a time [7]. The values are elements from a chosen field $\mathbb{F}_q$ (for binary field q = 2, or q = 2^W for extension field of the binary field). The purpose of using LFSR as shown in Fig.1 was to deliver a stream cipher with high performance properties and for achieving uniformed distribution of the values generated by the stream cipher.

In most cases, the immediate output of LFSR is not acceptable as a keystream since the value production is done in linear fashion. Therefore, stream ciphers which belong to this category are required to use other techniques as a combination with the LFSR to reach a reasonable level of security.

Non-Linear Feedback Shift Register (NLFSR)-based stream cipher: Due to the infeasibility of using LFSR without any combination with other non-linear components, feedback registers with non-linear

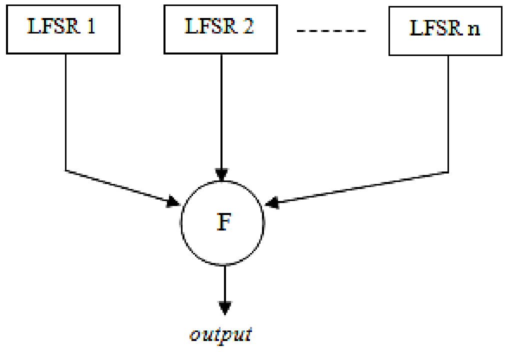


Fig. 2: Non-linear F combined with LFSRs outputs

functions were studied intensively and they become important components of modern stream ciphers. The use of non-linear functions combined with LFSRs will insure more uniformity of the values distribution generated by the corresponding keystream generator. The combination between LFSRs and non-linear function is graphically illustrated in Fig. 2.

The above combination can be achieved in many ways, such as by using Boolean functions or S-Boxes. Boolean function is substantially a mapping of *n* binary input variables to one binary output variable in the finite filed of characteristic two as appeared in Eq.1:

$$F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2 \qquad (1)$$

The representation of Boolean functions depends generally on the number of the input variables n. If the value of n is small, a truth table can be built where all probable input values are listed with the corresponding output value. On the other hand, if the value of n is large, a compact description such as Algebraic Normal Form (ANF) presented in Eq.2[8]:

$$F(\chi_1,\chi_2,\ldots,\chi_n) = a_0 + \sum_{i=1}^{n} a_i\chi_i + \sum_{i=2}^{n} a_{i-1,i}\chi_{i-1}\chi_i + \ldots + a_{1,2,\ldots,n}\chi_1\chi_2\cdots\chi_n \qquad (2)$$

The other common combination is the use of the substitution box (S-Box). The S-Box can be viewed as a function that takes an input to retrieve one corresponding value in the lookup table (S-Box). The mapping between the input and the output values is defined in Eq.3 for input x of n bits, and of m Boolean function.

$$s[\chi] \in \mathbb{F}_2^n \rightarrow y \in \mathbb{F}_2^m \qquad (3)$$

The S-Box construction is a difficult process since its main purpose is to be resistible for algebraic attacks and able to break the linearity of the LFSRs outputs.

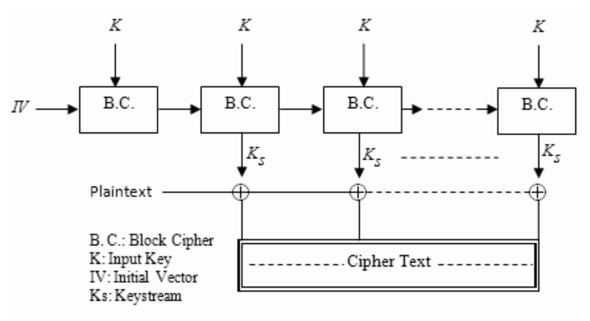


Fig. 3: Block Cipher-based Stream Cipher Scheme

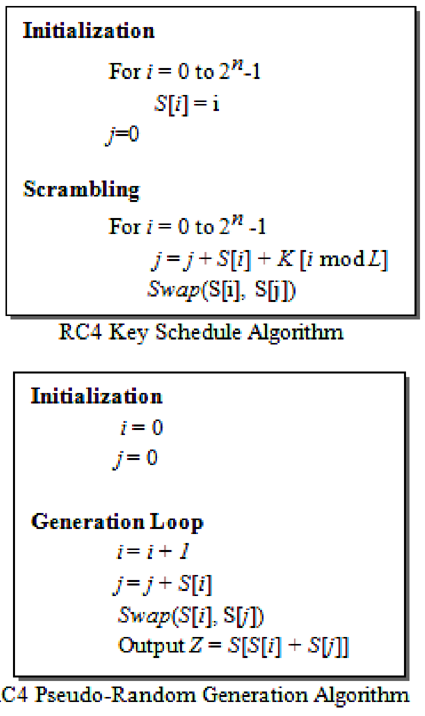
Examples of NLFSR-based stream ciphers are Dragon[9] DECIMv2[10] and An NLFSR-based stream cipher [11].

Block Cipher-Based Stream Cipher: Another technique used in stream cipher is by using a block cipher as a core of the keystream generator of the corresponding stream cipher. The construction of the stream ciphers that belong to this category can be directed depending on known block cipher such as using AES in LEX[12]. Other design philosophy of stream ciphers that belong to this category is based on the Substitution-Permutation Network (SPN) of block cipher as appeared in Hermes8[13]. The general structure of block cipher-based stream cipher is portrayed in Fig. 3.

The security of such design relies on the security of the underlying block cipher that resides at the core of the stream cipher that belongs to this category. In this context, one of the most important and secure block ciphers currently used widely in many cryptographic applications is the Advances Encryption Standard (AES). Up to this day, among the three submitted block cipher-based stream ciphers, LEX and Sosemanuk [14] are the two ciphers which have moved to the third phase of evaluation of the eSTREAM project[15].

T-function-based stream cipher: T-Functions were first introduced in 2002[16], forming a new class of invertible mapping. This class of functions has been discussed intensively in[17] and[18]. The T-function F(χ)works as a mapping function from n-bit input to n-bit output without propagating information from the Least Significant Bit (LSB) “bit 0” to the Most Significant Bit (MSB) “bit n-1”, and depend only on bits j = 0,1,…,i of the input. Examples on such T-functions of single variable are the mapping functions appeared in Eq. 4-6:

$$\chi \rightarrow \chi + 2\chi^{2} \qquad (4)$$



```
Initialization
    For $i = 0$ to $2^n-1$
        $S[i] = i$
    $j=0$

Scrambling
    For $i = 0$ to $2^n -1$
        $j = j + S[i] + K[i \bmod L]$
        $Swap(S[i], S[j])$
RC4 Key Schedule Algorithm

Initialization
    $i = 0$
    $j = 0$

Generation Loop
    $i = i + 1$
    $j = j + S[i]$
    $Swap(S[i], S[j])$
    Output $Z = S[S[i] + S[j]]$
RC4 Pseudo-Random Generation Algorithm
```

Fig. 4: RC4 algorithm

$$\chi \rightarrow \chi + (\chi^{2} \vee 1) \qquad (5)$$

$$\chi \rightarrow \chi \oplus (\chi^{2} \vee 1) \qquad (6)$$

The above T-functions require only three primitive operations over n-bit words and they are invertible for any *n*. Multivariate mappings shown in Eq. 7-8:

$$\chi \rightarrow \chi \oplus 2(\chi \vee y) \qquad (7)$$

$$y \rightarrow (y + 3\chi^{3}) \oplus \chi \qquad (8)$$

Are also invertible for any *n*. The T-function makes one cycle of maximal length. However, single-word T-functions are infeasible due to its limitation in bit size. Therefore, a multivariate-word T-function is proposed to be used as an alternative method for stream ciphers. Different T-function-based stream ciphers have been presented recently such as VEST[19] and ABC[20] stream ciphers.

Another stream cipher design (RC4): There are several stream ciphers which do not fit into the mentioned categories above (e.g. RC4). The well known stream cipher RC4 is widely used in many security protocols such us SSL and WEB protocols as mentioned in this paper so far. It was designed by Ron Rivest in 1987[21] for RSA Data Security, Inc. RC4 is a variable key-size cipher with compact code size and it is suitable for byte-oriented processors. The encryption of a message in RC4 is achieved by generating a keystream to be XOR'ed with a stream of plaintext to produce a stream of ciphertext. The RC4 algorithm is portrayed in Fig. 4.

Generating the keystream in RC4 requires a permutation array *S* of all 256 possible byte using RC4 Key Scheduling Algorithm (KSA). At the present time, RC4 is not recommended for use in new applications. Several weaknesses of the KSA algorithm of RC4[22] can be summarized in two points. The first weakness is the existence of massive classes of weak keys. These classes enable the attackers to determine a large number of bits of KSA output by using a small part of the secret key. Thus, the initial outputs of the weak keys are disproportionally affected by a small portion of key bits. The second weakness rests on a related key vulnerability.

Brute Force attack on RC4 is possible by implementing exhaustive key-searches on Field Programmable Gate Array (FPGAs) using a Network on Chip (NoC) architecture[23]. The idea of this attack relies on two components: Key-Checker Unit and the Controller. The latter is responsible for distributing the key space. Key-Checker Unit is used to check each key independently. Therefore, using more than one Checker in a network will provide an adjustable level of parallelism. The research's results shows that RC4 is quite vulnerable to brute-force attack and able to crack it in minutes with a very large FPGA of 500 Checker units in a network.

However, other kinds of attacks on RC4 have been presented recently. Results[24] showed a statistical bias of the digraphs distribution of the generated stream of RC4. Furthermore, a distinguishing attack clarified and developed based on the statistical bias found in the output sequences [25]. This bias is used along with the first two words of a keystream associated with around 2^{30} secret keys.

Due to the wide adaption and the high performance of RC4, this paper will consider RC4 as the main stream cipher for its comparisons with DSP-128 stream cipher. The next section will describe the structure of DSP-128 followed by an analysis section to analyze the security and the performance property of the proposed stream cipher in this paper.

THE DESIGN OF DSP-128

The use of Discrete Logarithm Problem in our proposed cipher is promising since these problems are widely used in different cryptographic primitives (e.g. Key Exchange, Digital Signatures, etc) and they showed great resistance against all types of attacks. The security of the proposed DSP-128 rests on the intractability of DLP, which is generally defined as follow:

Definition: For a given z, y$\in$G, find the integer x such that $z^X = y$.

This definition in its simplest form consists of three important parts: the field G, the field's elements z and y, and the exponential integer x. The field can be any finite field of characteristic two or above. In DSP-128, the considered finite field is of characteristic two and the representation of the field's elements is founded on polynomial representation. In other words, the elements of G which is $\mathbb{F}_{2^{128}}$ in DSP-128 are in polynomial representation with binary coefficients[0,1]. The degree of the polynomial elements is varied and can rise up to 128 degrees. All the operations will be performed modulo an irreducible (prime) polynomial of degree 128. In this section and subsequently, we refer to the polynomial by β^r where r is the degree of the polynomial. For the irreducible polynomial of degree r, the symbol is δ^r Generally, polynomials of degree r come in the following form (Eq.9):

$$\beta^r = a_m\chi^r + a_{m-1}\chi^{r-1} + \ldots + a_2\chi^2 + a_1\chi + a, \text{ where } a_m \in \mathbb{F}_q \qquad (9)$$

The proposed design works as follows: an input key of 128-bit length generates a keystream. The keystream length is in 128-bit block. The keystream is divided into 4-word (sub-keystream). These sub-keystream bits are XOR'ed with the plaintext bits (one word of plaintext at a time) to generate a stream of ciphertext. The design of DSP-128 cipher is divided into three main stages: Initialization Stage (IS), Keystream Generation Stage (KGS) and Encryption Stage (ES).

Initialization Stage (IS): At this stage, the input key ℓ (128-bit) determines the initial value of the secrete value *c* which is an integer value of 128-bit used as a counter. The second value generated from ℓ is the Polynomial Generator Ω^r of degree r. Regardless of the input key, an Irreducible Polynomial $\delta^{128}$is chosen and tested to reduce all the polynomial resulted by executing different arithmetic operations over $\mathbb{F}_{2^{128}}$. Fig. 5 demonstrates the functions implemented in IS.

The Generator is responsible for generating a primitive polynomial cover $\mathbb{F}_{2^{128}}$ which will be used in the next stage as a base polynomial of the DLP. The

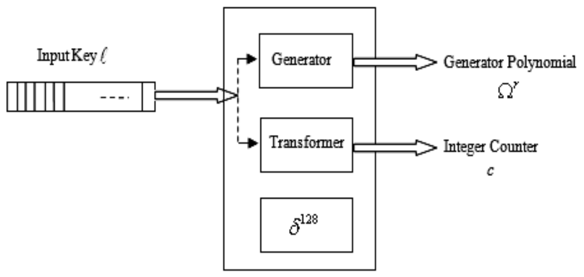


Fig. 5: Initialization Stage (IS)

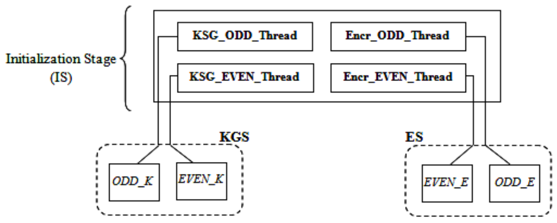


Fig. 6: Threads Creation

purpose of generating the primitive polynomial is to ensure generating pseudorandom output. The Transformer uses ℓ to transform it into a big integer c used as counter in the KGS. The value of the c is incremented by four bytes and the degree of the polynomial Ω^r is limited up to degree 128. It is also unchangeable during the execution of the whole stream cipher.

In term of multithreading, the IS functions as a threads creator which creates the threads responsible for generating keystream bits and encrypting these bits with its analogue stream of plaintext in multithread-fashion. IS will generate four threads as shown in Fig.6. *KSG_ODD/KSG_EVEN_Thread* threads are used for generating keystream in KGS, meanwhile *Encr_ODD/Encr_EVEN_Thread* are used to encrypt the plaintext with its corresponding keystream as discussed later in this section.

The *ODD_K/E* and *EVEN_K/E* threads appearing in the two stages KGS and ES of the above figure refer to the processing fashion. The generated keystream from *ODD_K* thread of KGS will be used for encrypting its corresponding plaintext in the *ODD_E* thread of ES. The *EVEN_K/E* threads in KGS and ES are working in the same way and concurrently with the *ODD_K/E* threads.

Keystream Generation Stage (KGS): The core of DSP-128 is located at this stage. The process of generating keystream relies on the hard mathematical

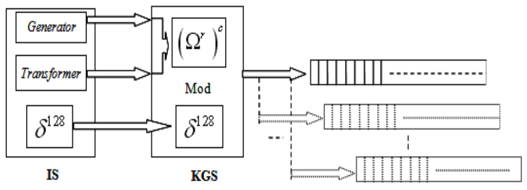


Fig. 7: Keystream Generation Stage (KGS)

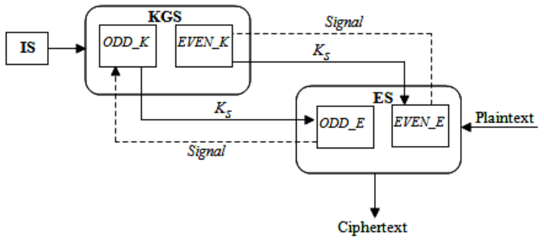


Fig. 8: Thread Synchronization between KGS and ES

problem DLP founded in polynomial arithmetic. The generated and initialized variables and polynomials at the first stage IS are processed here to generate pseudorandom keystream bits. The incremental integer c is the exponential of the polynomial generator$(\Omega^r)^c$. For the given polynomial Ω^r of degree r, the generated keystream is the transformation of the polynomial β^r resulting from Eq.10:

$$K_s = \beta^r = \left(\Omega^r\right)^c \bmod \delta^{128} \tag{10}$$

Each time KGS is required to generate a keystream, the value of the counter c is incremented by four bytes and the same formula above is executed again. The generator polynomial Ω^r is able to generate a pseudorandom output within the field defined by δ^{128} The degree of polynomials β and Ω can be different and we refer to it as r_1 and r_2 respectively. Figure 7 graphically illustrates the considered method of generating a keystream using the variables provided in IS.

Due to the use of multithreading, the process of generating new keystream and encrypting it is done in *Producer-Consumer* fashion. The *ODD_K* thread of the KGS is the producer thread which produces the keystream and triggers the *ODD_E* thread to perform the encryption on the plaintext. The KGS is divided into two parts. The first part is generating keystreams based on the odd incremented values of the counter c, while the second part is generating keystream based on the even incremented values of the same counter c. The

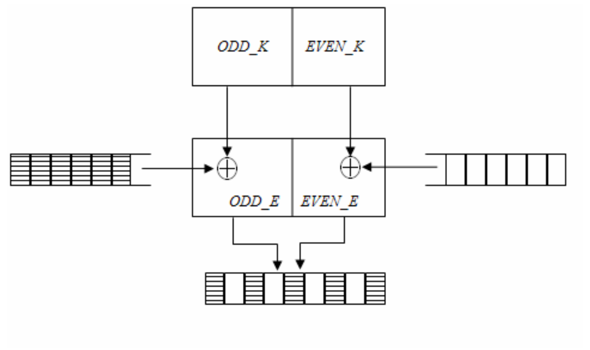


Fig. 9: Encryption Stage (ES)

synchronization between the corresponding threads located in the two stages KGS and ES is portrayed in Fig.8.

The synchronization is done by generating K_S in KGS, then sending a signal to its analogue thread in ES to do the encryption. In turn, the corresponding thread in ES will check whether any stream of plaintext still needs to be encrypted. If so, that thread will send a signal to its corresponding thread in KGS. Therefore, the DLP used to generate keystream in both threads will have the forms in Eq.11-12 based on the corresponding thread:

$$\beta_1^{r_1} = \left(\Omega^{r_2}\right)^{c_i} \bmod \delta^{128} \tag{11}$$

$$\beta_2^{r_1} = \left(\Omega^{r_2}\right)^{c_j} \bmod \delta^{128} \tag{12}$$

Where c_i and c_j are the incrementing of counter c by odd and even values respectively. The result of this stage is two different polynomials $\beta_1^{r_1}$ and $\beta_2^{r_2}$ from ODD_K and EVEN_K threads at the same time. These polynomials are later transformed into two different keystream K_{S1} and K_{S2} that are used for encryption in the next stage.

Encryption Stage (ES): The encryption process is done like any other stream cipher, that is by applying exclusive-OR (XOR) operation between the keystream bits and the plaintext bits. In DSP-128, the encryption is done by XOR one word (32-bit) of the generated K_S from the previous stage with one word of the plaintext. The encryption operations are accomplished by dividing the activities into two threads synchronized with their analogue threads in KGS (total of four threads). *ODD_E* thread is responsible for encrypting the odd orders of the given plaintext with the keystream generated by *ODD_K*. Concurrently, the keystream generated by EVEN_K is used to encrypt the even orders of the same given plaintext as shown in Fig. 9.

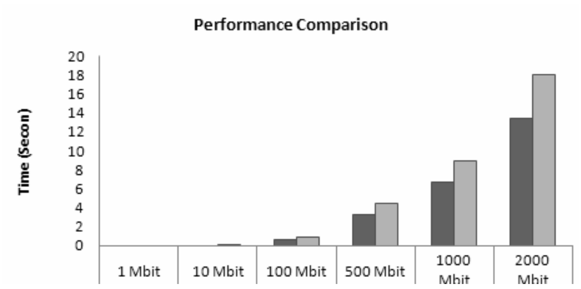


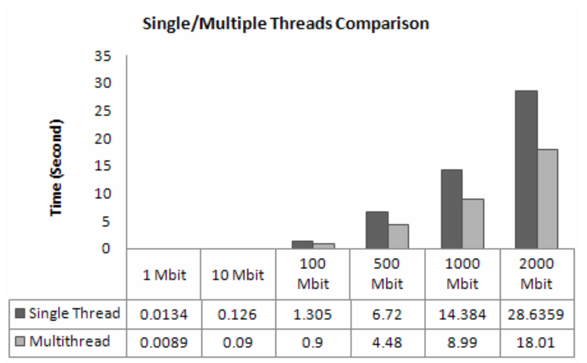
Fig. 10: Performance Evaluation and Comparison between DSP-128 and RC4

DSP-128: IMPLEMENTATION AND PERFORMANCE EVALUATION

We compare the performance of our proposed stream cipher, which is based on DLP, against the widely adapted RC4 stream cipher, which is based on simple substitutions and shift operations. Both ciphers were coded in C++ using MinGW-2.05. NTL library is used to handle the polynomial arithmetic found in DSP-128. For testing purposes, we ran both ciphers on a workstation with Core 2 Duo® 2.13GHz, 2.13 GHz processor, and memory RAM of 2GB.

In term of the performance, DSP-128 is about one time slower than RC4. Therefore, DSP-128 shows a good alternative stream cipher with regards to the high security provided compared to the broken RC4. The encryption rate of our proposed cipher is 112.903 MBits/Second compared to RC4 encryption rate of 152.173 MBits/Second on the same machine. Fig.10 illustrates the performance comparison between the multithreaded DSP-128 and RC4 with different plaintext sizes. The figure shows that DSP-128 required more time than RC4 to encrypt the same size of plaintext. At the same time, this difference is considered negligible due to the security level provided by our proposed cipher.

Two factors play an important role in speeding up the keystream generation process. The first factor is the representation of the underlying field elements defined in this cipher. The representation is founded on fast polynomial arithmetic over $\mathbb{F}_{2^{128}}$. The coefficients of these polynomials are either 1 or 0, which reduce the time consumed for executing arithmetic operations compared to the field of characteristic greater than 2.



	1 Mbit	10 Mbit	100 Mbit	500 Mbit	1000 Mbit	2000 Mbit
■ Single Thread	0.0134	0.126	1.305	6.72	14.384	28.6359
□ Multithread	0.0089	0.09	0.9	4.48	8.99	18.01

Fig.11: Performance Comparison between Single-thread and Multi-thread DSP-128

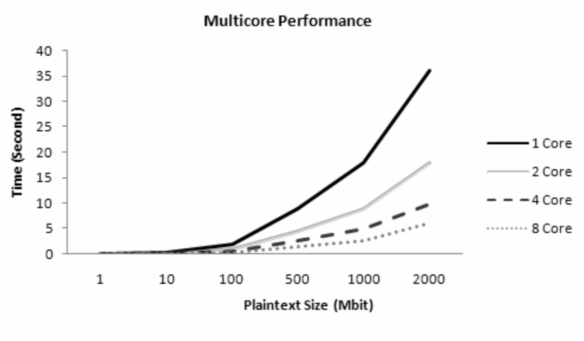


Fig.12: Multi-core Performance Evaluations and Expectations

The second factor affecting the performance of DSP-128 is the use of multithreading techniques. Results show magnificent performance improvement compared to the use of a single-thread model. Figure 11 shows that a single-thread model is approximately one time slower than the multithreaded model.

In fact, current processors tend to use multi-core technology for performance enhancement. DSP-128 makes use of this technology and results presented in Fig. 12 show the improved performance by implementing our cipher on two cores compared to a single-core platform. The two discrete lines appearing in the figure refer to the expected performance with more than two cores. However, the increasing number of cores seems promising in terms of performance due to the use of multithreading techniques.

SECURITY ANALYSIS

The security of our proposed cipher is based on the intractability of solving the DLP, which is an NP-hard problem. At present, no algorithm can solve this

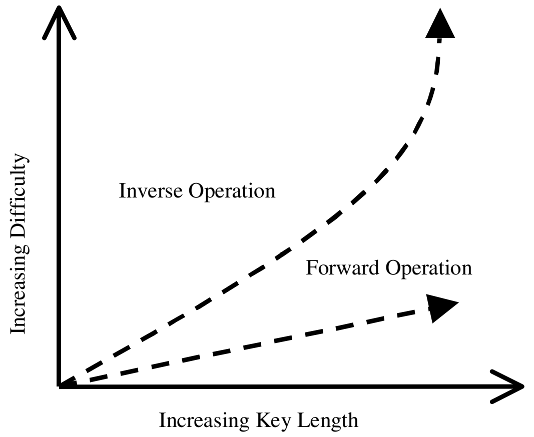


Fig. 13: Difficulty of forward, inverse operation against key length

problem in polynomial time. Therefore, algebraic attacks on the DLP implemented to generate the keystream are infeasible. Another attack that can be discussed here is Brute-Force attack. It is one of the most important attacks on any cryptographic primitive. This attack is avoided in our cipher by using 128-bit input key ℓ. The size of the input key, providing us huge key space 2^{128} and making brute-force attack infeasible.

The length of the key used in DSP-128 was chosen to achieve the expected security against brute force search attack and to satisfy the required balance of the difficulty between the forward and the inverse operations. The forward operation of DSP-128 is the polynomial exponential $(\Omega^r)^c$ modulo irreducible polynomial over $\mathbb{F}_{2^{128}}$, which is fast enough compared to its inverse operation (retrieve c given Ω^r and β^r). In general, the difficulty of the inverse operation against the key length must be exponential for secure cryptographic primitives as portrayed in Fig. 13.

On the other hand, attacking the polynomial representation and the underlying finite field is also avoided. The polynomial used for representing the DLP is of degree 128, which gives extra security level for DSP-128. In addition, the finite field $\mathbb{F}_{2^{128}}$ consists of a huge number of elements (2^{128}), which increase the level of the security compared to other cryptographic primitives that use smaller field size.

In fact, our proposed stream cipher is considered a good alternative for current stream ciphers. It provides a new direction for implementing stream ciphers. The performance provided by single-threaded DSP-128 is slightly less than the well know and fast stream cipher RC4 (but with a sequential design). At the same time, the security level of DSP-128 is higher than what is provided in RC4 and many other stream ciphers that belong to the categories discussed so far.

CONCLUSION

In this paper we present a new stream cipher known as DSP-128, based on the intractability of the Discrete Logarithm Problem (DLP). DSP-128 is divided into three main stages: Initialization Stage (IS), Keystream Generation Stage (KGS) and the Encryption Stage (ES). The performance, security evaluation and analysis showed that DSP-128 is a reliable and good alternative stream cipher. We presented a new design which is not fit for other stream cipher categories discussed in this paper. Moreover, the architecture of our cipher is based on multithreading techniques implemented on multi-core technology, forming a new class of stream ciphers that make use of currently available technologies. The multithreaded design showed great improvement in performance compared to the single-thread (sequential) model. In addition, complete descriptions of DSP-128 stages, implementation and an overview of possible attacks have been discussed.

ACKNOWLEDGMENT

The authors would like to express their thanks to Universiti Sains Malaysia for supporting this study.

REFERENCES

1. Afzal, M., F. Kausar and A. Masood, 2006. Comparative Analysis of the Structures of eSTREAM Submitted Stream Ciphers, Proceeding of International Conference on Emerging Technologies ICET '06, pp: 245-250.
2. Odlyzko, A., Discrete Logarithms in Finite Fields and their Cryptographic Significance, 1984. Advances in Cryptology - Eurocrypt '84, Springer, Verlag, pp: 224-314.
3. Diffie W. and M. Hellman, 1976. New Directions in Cryptography, IEEE Trans. Inform. Theory, IT-vol. 22, pp: 644-654.
4. ElGamal, T., 1985. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Transactions on Information Theory CRYPTO 84, Springer-Verlag, Berlin, pp: 10-18.
5. Du, W. and K. Chen, 2006. Construction of Digital Signature Based on Discrete Logarithm Problem, Progress on Cryptography, Springer, Netherland, vol. 769, pp: 67-71.
6. Freier, A. and P. Karlton, P. Kocher, 1996. The SSL Protocol Version 3.0 <http://wp.netscape.com/eng/ssl3/ssl-toc.html>.
7. Ekdahl, P., 2003. On LFSR based Stream Ciphers Analysis and Design, PhD Thesis, School of Information Technology, Lund University, Lund. See Wikipedia. Algebraic Normal Form. <http://en.wikipedia.org/wiki/Algebraic_normal_form>.
8. Chen, K., M. Henricksen, W. Millan, J. Fuller, L. Simpson, E. Dawson, H. Lee and S. Moon, 2005. Dragon: A Fast Word Based Stream Cipher, Lecture Notes in Comp. Sci. Springer-Verlag, Berlin, pp: 33-50.
9. Berbain, C., O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin and H. Sibert, 2005. Decim-A New Stream Cipher for Hardware Applications, ECRYPT Stream Cipher Project Report 2005/004. <http://www.ecrypt.eu.org/stream/>
10. Gammel, B., R. Gottfert, O. Kniffler, 2006. An NLFSR-Based Stream Cipher, In Proc. IEEE International Symposium on Circuits and Systems (ISCAS 2006), Greece, pp: 4-8.
11. Biryukov, A., 2005. A New 128 bit Key Stream Cipher: LEX, eSTREAM Report 2005/013. <http://www.ecrypt.eu.org/stream/p3ciphers/lex/lex_p3.zip>
12. Kaiser, U., 2006. Hermes8: A Low-Complexity Low-Power Stream Cipher, Cryptology ePrint Archive, Report 2006/019. <http://eprint.iacr.org/2006/019.pdf>.
13. Berbain, C., O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin and H. Sibert, 2005. SOSEMANUK: A Fast Software-Oriented Stream Cipher, eSTREAM Project. http://www.ecrypt.eu.org/stream/p3ciphers/sosemanuk/sosemanuk_p3.pdf ECRYPT, eSTREAM Phase 3 Candidates, eSTREAM Project. <http://www.ecrypt.eu.org/stream/phase3list.html
14. Klimov, A. and A. Shamir, 2002. A New Class of Invertible Mappings, Workshop on Cryptographic HW and Embedded Systems (CHES), Springer-Verlag, Berlin, vol. 942 of LNCS, pp: 470-483.

15. Klimov, A. and A. Shamir, 2003. Cryptographic Applications of T-Functions, Selected Areas in Cryptography (SAC), Springer-Verlag, Berlin, vol. 3006 of LNCS, pp: 234-247.
16. Klimov, A., 2004. Applications of T-Functions in Cryptography, PhD Thesis, Weizmann Institute of Science.
17. Anashin, V., A. Bogdanov and I. Kizhvatov, 2005. ABC: A New Fast Flexible Stream Cipher, eSTREAM Project. <http://www.ecrypt.eu.org /stream/ciphers/abc/abc.pdf>.
18. O'Neil, S., B. Gittins and H. Landman, 2005. VEST Ciphers. eSTREAM Project. <http:// www.ecrypt.eu.org/stream/p2ciphers/vest/vest_p2. pdf>.
19. Rivest, R., 1992. The RC4 Encryption Algorithm. RSA Data Security Inc. Document No, 003-013005-100-000000.
20. Fluhrer, S., I. Mantin and A. Shamir, 2001. Weaknesses in the Key Scheduling Algorithm of RC4, In Proc. 8th Workshop on Selected Areas in Cryptography, Springer-Verlag, Berlin, vol. 2259, pp: 1-24.
21. Couture, N. and K. Kent, 2004. The Effectiveness of Brute Force on RC4, cnsr, IEEE Computer Society, CA, USA, pp: 333-336.
22. Itsik Mantin, 2007. Predicting and Distinguishing Attacks on RC4 Keystream Generator, Advances in Cryptology, Springer-Verlag, Berlin, vol. 3494, pp: 491-506,.
23. Tsunoo Y., H. Kubo and T. Suzaki, 2007. A distinguishing Attack on a Fast Software-Implemented RC4-Like Stream Cipher, IEEE Trans. on Information Theory, vol. 53, pp: 3250-3255.