**Available Online at www.ijcsmc.com**

**International Journal of Computer Science and Mobile Computing**

**A Monthly Journal of Computer Science and Information Technology**

**ISSN 2320–088X**
**IMPACT FACTOR: 6.017**

*IJCSMC, Vol. 7, Issue. 11, November 2018, pg.105 – 111*

# Modified Geffe Generator Circuit for Stream Cipher

**Farah Maqsood**
Department of Optometry and Vision Sciences, King Saud University, KSA
farahmaqsood@gmail.com

***Abstract*— Encryption algorithms provide the securely transmitted data over insecure communication channels. In this paper a different stream cipher scheme is presented which is very simple in hardware implementation and provides relatively high level of security. Geffe generator is used for the generation of a complex key in a unique way. In this modified scheme, keystream is dependent not only on the inputs of the Geffe generator but also on the Gold Codes generated with the help of outputs of pseudo noise (PN) sequences of the two inputs of Geffe generator. Further complexity of the key is increased by introducing automatic shift of the contents of one LFSR to generate a new Gold code.***

***Keywords*— "Geffe generator", "Gold Code (GC)", "Maximal Sequence (m-sequence)", "Linear Feedback Shift Register (LFSR)", "Pseudorandom number generator (PRNG)".**

## I. INTRODUCTION

Secure communication is commonly used in commercial as well as domestic applications. The commercial applications require simple hardware at the transmitting and receiving ends to incorporate the data secrecy [1], [2]. Cryptography is the science of writing in secret code. In data and telecommunications, cryptography is necessary when communication is over any untrusted network. Encryption techniques require both communicating parties to share knowledge of a secret key, and those other parties do not have access to this key [3]. Key agreement is the process by which the portable system and the network agree upon the proper key. Authentication is the process which ensures that the portable unit does not access the network using a false identity in order to avoid charges for usage. This paper presents a Geffe generator based key agreement, which maintains privacy of conversation and deter usage fraud. For class demonstration in laboratories of educational institutions these types of encryption methods can be implemented in order to explain the encryption/decryption processes in a better and simpler way.

## II. THEORETICAL BACKGROUND ON STREAM CIPHERS AND PROPOSED CIRCUITS

Encryption can be done via cryptographic hardware or via software. There are two classes of key-based encryption algorithms, symmetric (or secret key) and asymmetric (or public key) algorithms. Symmetric algorithms use the same key for encryption and decryption, whereas asymmetric algorithms use a different key for encryption and decryption, and the decryption key cannot be derived from the encryption key. Stream cipher

 

process the given message (known as plaintext) bit by bit or byte by byte (as a stream). So stream ciphers encrypt each bit/byte of the input data individually (at a certain time) before moving on to the next. Stream ciphers are also known as state ciphers since the encryption of a bit is dependent on the current state. A stream cipher is a type of symmetric encryption algorithm [4].
A key stream generator outputs a stream of bits: $k_1$, $k_2$ ...$k_n$. This keystream is XORed with a stream of plaintext bits, $p_1$, $p_2$ ...$p_n$ in a bit by bit manner in order to produce the stream of ciphertext bits.

$$c_i = p_i \oplus k_i$$

At the decryption end, the ciphertext bits are XORed with an identical key stream to recover the plaintext bits.

$$p_i = c_i \oplus k_i$$

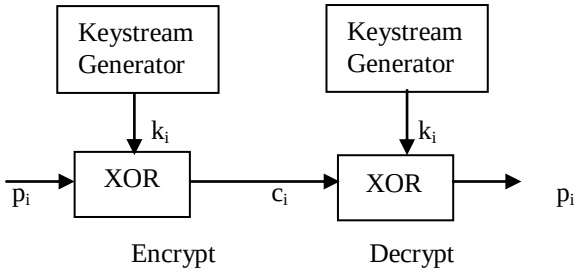


Fig. 1 Synchronous Stream Cipher

The systems security depends entirely on the insides of the key stream generators. The synchronous stream ciphers (Fig. 1) are the ciphers, in which the keystream is generated independently of the plaintext and the ciphertext [5]. They generate bits from a source other than the message itself. The simplest of such cipher extracts bits from a register to use as the key. The contents of the register change on the basis of the current contents of the register. Most stream cipher designs are for synchronous stream cipher. Some ciphers, called self-synchronizing stream ciphers (shown in Fig. 2), use several previous ciphertext bits to compute the keystream. They depend on the data and its encryption and also on the seed. A single bit error will result in a long burst of garble, but the cipher will eventually recover from a lost bit after the damaged and incorrect bit falls off the shift register.

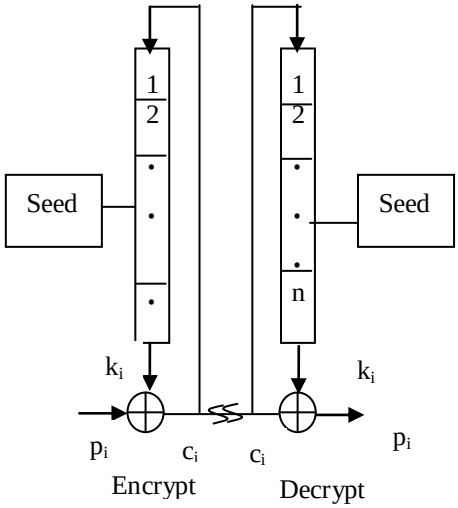


Fig. 2 Self-Synchronizing Stream cipher

In this figure seed is required for the input of the register, which is loaded in the register at the beginning. In Fig. 3. an improved circuit for self synchronizing stream cipher is shown [6]. In this circuit complexity of the key is increased by selecting the register's outputs randomly through Q's as shown in the figure.

 

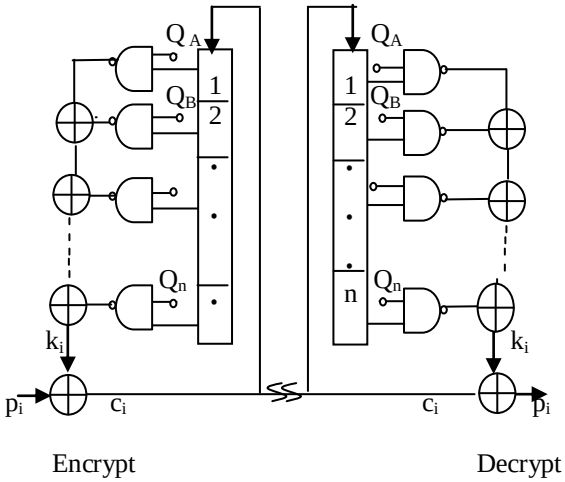


Fig. 3. Proposed Self Synchronizing Stream cipher

The basic approach to designing a key stream generator using linear feedback shift registers (LFSRs) is simple. Key is the initial state of the LFSRs so most stream ciphers consist of a pseudorandom number generator (PRNG). For encryption the PRNG is initialized with a key and provides its output as a sequence of bits known as the (pseudo) random key stream. Randomness is very important because it completely destroys any statistical properties in the message. So complications have been added to the key stream generator. Some generators have LFSR clocked at different rates; sometimes the clocking of one generator depends on the output of another "7". Simple LFSRs generate PN sequences depending upon the feedback tapping and are very simple for hardware implementation. But they are not considered for stream enciphering because the secrecy obtained through them is poor. An n-bit LFSR can generate different m-sequences of $2^n$-1 bits depending upon the feedback tappings. If $2^n$-1 is a prime number then there are $\phi = (2^n\text{-}2)/n$ valid feedback tappings, which will generate the $\phi$ number of different m-sequences. However, if 2n+1 bits out of a m-sequence of $2^n$-1 bits are known the feedback tapping and hence the whole sequence can be easily obtained. This makes deciphering easier for unauthorized receivers. To overcome this weakness of LFSR based m-sequence various circuits are reported which utilize more than one LFSR of same or different lengths to generate a complex code.

Geffe generator (Fig. 4) is simple for hardware implementation and gives codes with good autocorrelation characteristics. A Geffe generator can be used for the generation of the keystream. This Geffe generator uses three LFSRs, combined in a nonlinear manner. Two of the LFSRs provide inputs to multiplexer, and the third LFSR controls the output of the multiplexer "2". A conventional Geffe generator provides a keystream, which has much longer period than a single linear machine [8], [9].

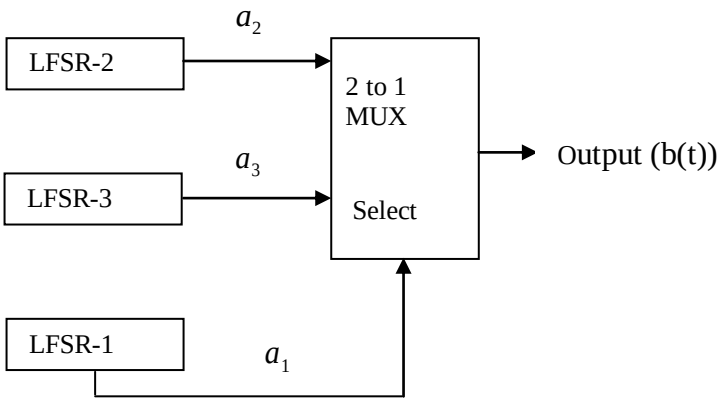


Fig. 4 Geffe generator

 

If $a_1$, $a_2$ and $a_3$ are the outputs of the three LFSRs, the output of the Geffe generator can be described by:

$$b = (a_1 \wedge a_2) \oplus ((\neg a_1) \wedge a_3)$$

Where $\oplus$ is XOR, $\wedge$ is AND and $\neg$ is NOT.

If the LFSRs have lengths $n_1$, $n_2$ and $n_3$ respectively, then the linear complexity of the generator is

$$(n_1 + 1)n_2 + n_1 n_3$$

For a lengthy sequence the circuit becomes complex. Feedback with carry shift register (FCSRs) and scheme using various combinations of LFSRs and / or FCSRs are reported and seem to be attractive but they involve mathematical complexity making the system complex for commercial applications "3". Instead of using LFSR1 for the input select lines of the $2 \times 1$ multiplexer. It is proposed to use Gold code, which can be generated with the help of LFSR2 and LFSR3. One Gold code (GC) generator circuit is shown in Fig. 5 [5]. Gold codes have low cross-correlation and hence low interference. Output of the Geffe generator is $a_2$ or $a_3$ depending upon $a_2 \oplus a_3$ is 1 or 0. Here Geffe generator uses only two LFSRs which simplify the circuit. Length of the LFSR's may also be taken relatively small (which further simplifies the circuit).

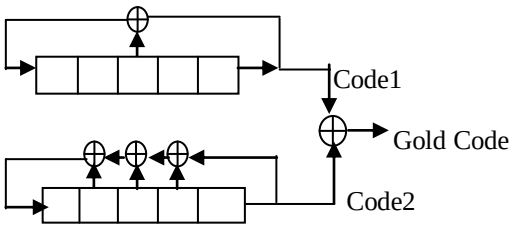


Fig. 5 Gold Code Generator

Length and complexity of the codes generated in this case is increased as described below. In this scheme output of the Geffe generator can be used as key for the encryption of the encryption of any message. If length of shift register is taken large in order to increase the period of key there may be problem of bit-error propagation; one bit error can cause n erroneous bits, so one should try to increase the complexity of the key keeping the length of LFSR small. Since Gold code is generated with the help of LFSR2 and LFSR3 and if these LFSRs length is having 5 stages so 31 bit maximal sequences are generated depending upon the feedback taps. Outputs of these LFSRs are modulo-2 added to generate the corresponding Gold codes. Feedback taps [5, 3] and [5, 4, 3, 2] are used for LFSR2 and LFSR3 respectively [1]. A proposed circuit for Geffe generator is shown in Fig. 6. If the clocks of LFSR3 are taken as shown in Fig. 7, where $a_1$,...$a_5$ are the outputs of LFSR2. $CK_2$ will be inhibited to appear whenever all the outputs of LFSR2 are 1. Thus output of LFSR2 will be one bit delayed after every 31 bits, which is the m-sequence length in this case. With this arrangement all the 31 GCs of 31 bit length each can be generated in series. With the delay bits included the total sequence length of GC will be $31 \times 31 = 961$ bits. After these 961 bits feedback combination of any one LFSR can be changed. This code will also be much more difficult to break as compared to simple m-sequence of similar length.

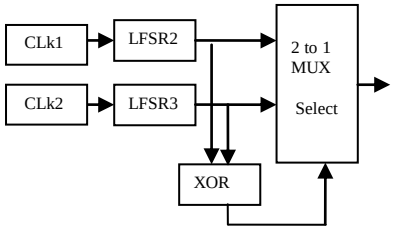


Fig. 6 Proposed circuit for Geffe Generator

 

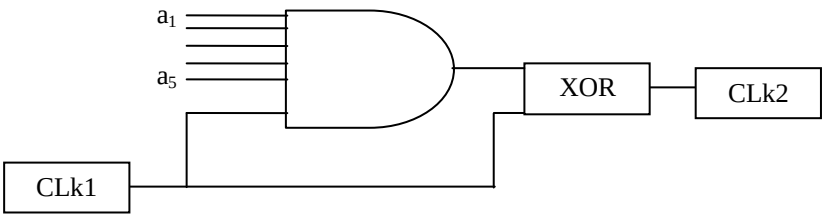


Fig. 7 Circuit for clock CLK2, where $a_1 ... a_5$ are the outputs of LFSR2

## III. RESULTS AND DISCUSSIONS

Output of the simple Geffe generator is shown in Fig. 8. The scheme presented in this paper is tested for the encryption of 31-bit length pseudorandom message signals (Fig. 11) using Simulink in MATLAB. Output of the Geffe generator is used for the encryption of 31-bit message signal. The input lines of Geffe generator are selected with the help of GC, which was generated by modulo-2 addition of outputs of two LFSRs. The results obtained are presented in Fig. 9.

Key length (period) is increased when one bit of LFSR3 is inhibited using the circuit as shown in Fig.7. Using this key (shown in Fig. 10) and message signal encryption is done by simple modulo-2 addition. As the periods of sequences are very large so large numbers of samples are taken in order to show the full lengths of the sequences. Decryption is done in order to extract information from the encrypted message using the same key. The results are shown in Fig. 12 and 13 respectively, and it is clear that scheme presented works satisfactorily. This scheme can be used for the encryption of speech as well as image signals.

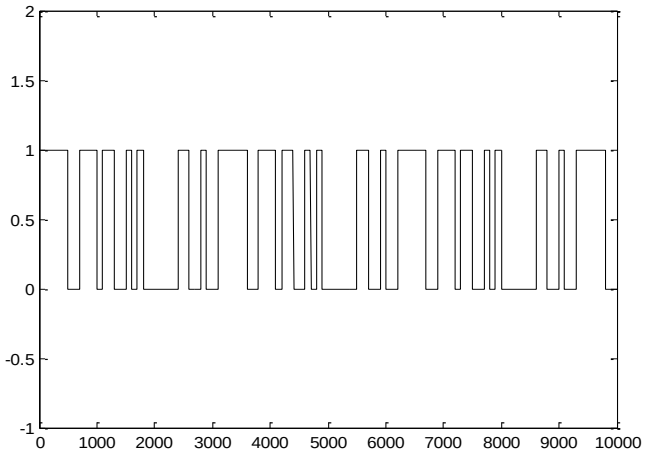


Fig. 8 Output of the Simple Geffe generator

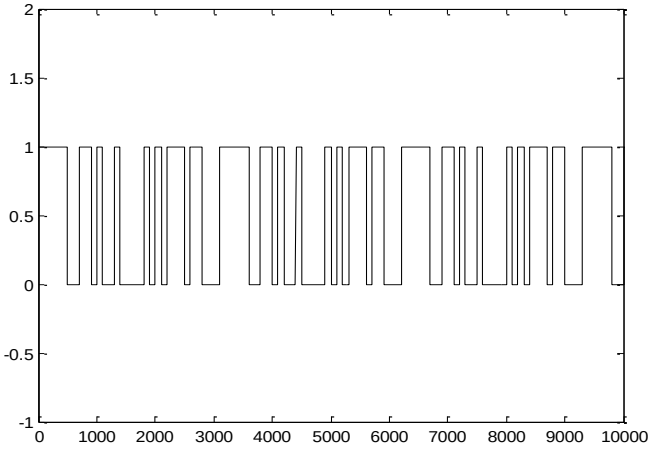


Fig. 9 Output of Geffe generator when input lines are selected with the Gold code

 

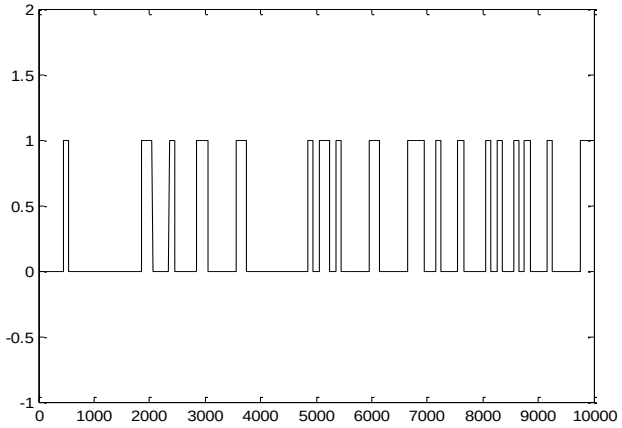


Fig. 10 Output of the Proposed Geffe generator when one bit is inhibited with the help of the circuit shown in Fig. 7

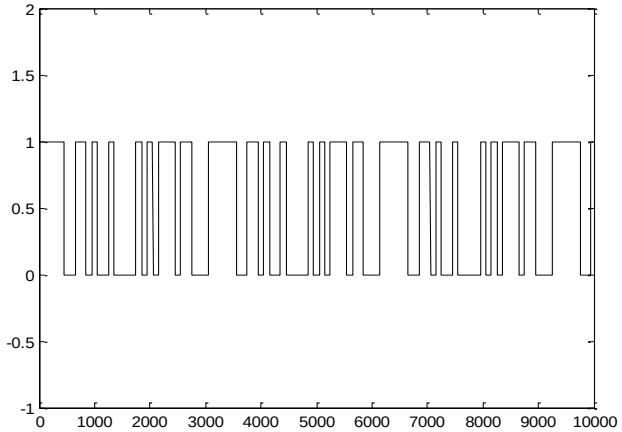


Fig. 11 31 bit Message signal

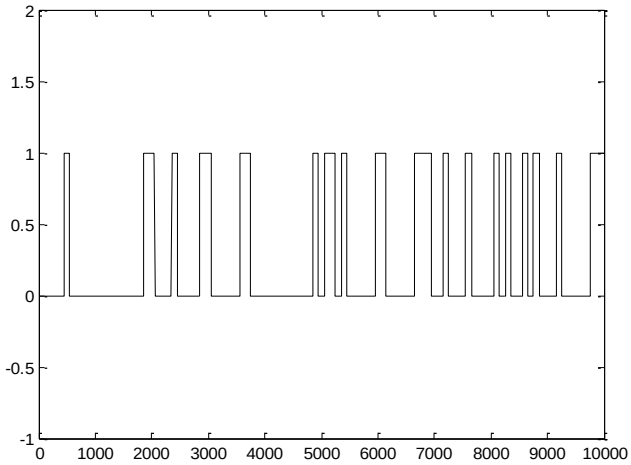


Fig. 12 Encrypted data (when output of the proposed Geffe generator is used as the Key for the encryption)

 

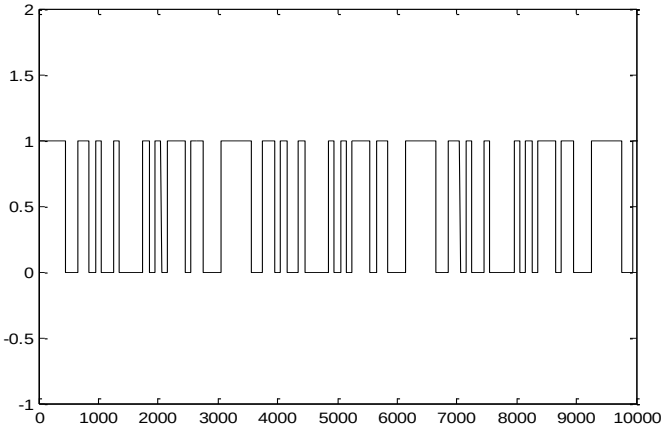


Fig. 13 Output of the Decrypter (when same key is used for decryption process)

## IV. CONCLUSION

During this work the attention was concentrated on the following issues:

- To design a circuit that can generate a keystream of longer period with simpler hardware.
- To increase the complexity and hence to increase the degree of security.
- To decrease the number of hardware.

Scheme presented in this paper generates a keystream of large period using minimum number of hardware. This scheme works very well in stream ciphers. Since this circuit is having low cost and simple for demonstration of encryption techniques, so it can be implemented in laboratory classes of educational institutions.

## ACKNOWLEDGEMENT

This research project was supported by a grant from the Research Center of the Female Scientific and Medical Colleges, Deanship of Scientific Research, King Saud University.

# REFERENCES

[1] R. C. Dixon, *Spread Spectrum Systems with Commercial Applications*, 3rd ed., New York: Wiley, 1994.
[2] M. J. Beller, L. F. Chang, and Yacov Yacobi, 'Privacy and Authentication on a Portable Communications System' *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 6, pp. 821-829, Aug. 1993.
[3] B. Schneier, *Applied Cryptography*, 2nd ed., New York: Wiley, 1996.
[4] M. Galanis, P. Kitsons, G. Kostopoulos, N. Sklavos, and C. Goutis, "Comparison of the hardware implementation of stream ciphers," *Int. Arab J. Infor. Tech*., vol. 2, no. 4, pp. 267-274, Oct. 2005.
[5] E. Zenner, "Stream cipher criteria," CRYPTICO A/S, [Online]. Available: http://www.ecrypt.eu.org/stream/papersdir/2006/032.pdf
[6] F. Maqsood, A. Ahmad and W. Ahmad, 'Scrambler for Increased Data Security and Low bit error Propagation', in *Proc. Of National Conference on Modern Trends in Electronics and Communication Systems*, Aligarh, 2005, paper, p. 91.
[7] Matt Bishop, *Computer Security- Art and Science*, Number ISBN0-201-44099-7, Pearson Education, 2003.
[8] H. T. Khamees, J. A. Kahlf, and A. A. Al-sajee, "Encryption and Decryption of data by using Geffe Algorithm", *IJMER*, vol. 2, no. 3, pp. 1354-1359, 2012.
[9] S. Wei, On generalization of Geffe's Generator, *IJCSNS*, vol. 6, no. 8A, pp. 161-165, 2006

 