CHAPTER 4

Circuits Developed for Complex Codes

4.1 Introduction

Most of the cryptographic techniques and protocols require a random or pseudorandom number source. The quality of this source is very crucial for security of schemes or protocols to be implemented. Perfect security is achieved when a cryptosystem is absolutely unpredictable for an external observer, i.e. all possible outcomes (states, sub-trajectories) are equiprobable and do not depend on the previous states. In other words, the state sequence has uniform probability distribution and no correlation, which is similar to white noise. This is very difficult to implement practically. Therefore, pseudorandom generators are being used for practical realization of random numbers. Various schemes of encryption had been proposed based on PN sequence generators [4.1-4.2].

Here, few schemes are presented for LFSR based complex code generators with a brief introduction of all the schemes. Most of the schemes designed and discussed in this thesis are simulated and studied through Simulink in MATLAB. Some schemes are also implemented with the help of ICs and other hardware components. The results taken from the circuits, implemented in lab using hardware and software with the help of MATLAB, showing interesting results. These results are in full agreement with the theoretical results and are shown in Chapter 6. In the end of this chapter, some complex code generator circuits are also presented using LFSR and programmable logic arrays (PLAs).

Linear feedback shift registers are commonly used to generate these pseudorandom sequences. Due to simple, cheap and low complexity hardware, LFSR has been proposed for encryption of packet payloads in Bluetooth communication. A5/1 encryption scheme based on LFSR is a GSM standard in the air transmissions [4.3].

Many keystream generator designs are based on the shift registers. To make use of the good keystream properties avoiding the inherent linear predictability of LFSR sequences, many constructions introduce nonlinearity by applying a nonlinear function to the outputs of regularly clocked LFSRs or by irregular clocking of the LFSRs. Because of the weakness of PN sequences, given in section 3.9.3, many new circuits are being reported in this chapter using LFSRs and FCSRs for the generation of the complex codes [4.4].
4.2 Gold Code Generator

An LFSR is a circuit, used to produce a sequence of symbols with good statistical properties. However, this sequence is very easy to predict due to the linear property of the circuit. Gold code generators using two LFSRs are very simple circuits for generating large number of pseudorandom codes with good correlation characteristics. Using two LFSRs (of n bit shift register each), \(2^{n-1} + 2\) Gold codes can be generated. These codes can be easily used to generate lengthy complex sequences through random and continuous shifting of bit positions added for the Gold code generation.

![Diagram of Gold code generator using 5-bit shift registers 7496.](image)

Fig. 4.1 Gold code generator using 5-bit shift registers 7496.

Fig. 4.1 shows a circuit for Gold code generation using 5-bit shift registers. Possible feedback taps of 5-bit shift register for maximal sequences are \([5,3]\), \([5,2]\), \([5,4,3,2]\), \([5,3,2,1]\), \([5,4,3,1]\) and \([5,4,2,1]\). This circuit uses two LFSRs with feedback tapings \([5,3]\), \([5,4,3,2]\) and third shift register SRIII. The output of the LFSR with \([5,3]\) feedback taping is giving the PN sequence which we shall call mother code. All the registers were clocked with the same clock. All LFSRs are loaded by a data other than all zero with the help of preset enable. Inputs X and Y (shown in Fig. 4.1) are obtained from LFSRII or SRIII drawn arbitrarily. Gold codes are generated by modulo-2 addition of the mother code bits and the output bits of the other LFSR of the same length with an arbitrary bit shifting.
Using 5-bit shift register, PN sequences of 31-bit length are generated and 31 Gold codes can be generated with serial modulo-2 addition of the mother code and the shifted versions of the code of LFSR II. Therefore, in total 33 different maximal sequences can be generated using two PN sequences. The details are shown in Fig. 4.1, in which only 10 shifted versions of LFSR I are considered, to avoid the complexity of the circuit [4.4]. In Chapter 5, Fig. 5.2 shows the arrangement used for the generation of the key using some of these codes.

4.3 A Modified Code Generator using LFSR and FCSR

This generator uses LFSR, SR, FCSR and XOR function to combine them. The XOR operation destroys the algebraic properties of FCSRs, so it makes sense to use it to combine them [4.5]. In the Fig. 4.2, instead of using LFSR I, 3-bit FCSR is used and rest of the circuit remains the same except the feedback taping of LFSR which is taken [5.3]. This will increase the period of the codes MC1 and MC2 and the length of the sequence generated in this case comes out to be 31×10 = 310 bits, which is 10 times larger than the length of the simple Gold codes generated using two 5-bit LFSRs as described in Fig. 4.1. For more complex and lengthy codes, one can increase the lengths of LFSR and FCSR.

We have designed codes using 3-bit FCSR, 5-bit LFSR and 5-bit SR, for the sake of simplicity.

3-bit FCSR

![Diagram of 3-bit FCSR](image)

Fig. 4.2 Modified codes generator using 5-bit shift registers and 3-bit FCSR.
4.4 LFSRs, FCSRs and Gold Code Based Modified Geffe Generators

In this section, a Geffe generator circuit (shown in Fig. 4.3) is being proposed, which is based on LFSRs and FCSR. This keystream generator uses two LFSRs, combined in a nonlinear manner. Here, LFSRs are the inputs into a multiplexer and FCSR controls the output of the multiplexer.

![Fig. 4.3 Modified circuit for Geffe generator.](image)

In the simple Geffe Generator (shown in Fig. 3.9) instead of using LFSR for the input, select lines of the 2×1 multiplexer. It is proposed to use Gold code, which can be generated by XORing the outputs of LFSR1 and LFSR2 as shown in Fig. 4.4.

![Fig. 4.4 Proposed circuit for Geffe generator.](image)

Gold codes have low crosscorrelation and hence low interference. If \( a_i \) and \( a_j \) are the outputs of the LFSR1 and LFSR2 then the output of the Geffe generator is \( a_i \) or \( a_j \) depending upon \( a_i \oplus a_j \) is 1 or 0. Here, Geffe generator uses only two LFSRs, which simplifies the circuit. Length and complexity of the codes generated in this case are increased as described below. In this scheme, output of the Geffe generator can be used as keystream for the encryption of any message. If length of shift register is taken large in
order to increase the period of keystream 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 keystream keeping the length of LFSR small.

4.5 Clock Control Geffe Generator

To introduce non-linearity into the generation of the keystream, one may use the idea of varying the rate at which a register is clocked. If some arrangement is made so that the clocking of one register is, in some way, dependent on another register, then it seems reasonable to suppose that more complex sequences will be generated.

A proposed circuit for Geffe generator is shown in Fig. 4.5. If the clocks of LFSR1 and LFSR2 are taken as shown in Fig. 4.6, where $a_1, ..., a_s$ are the outputs of LFSR1. CLK2 will be inhibited to appear whenever all the outputs of LFSR1 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 Gold codes (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. 4.5 Another circuit for Geffe generator.](image)

![Fig. 4.6 Circuit for clock CLK2, where $a_1, ..., a_s$ are the outputs of LFSR2.](image)
In Fig. 4.7 another keystream generator circuit is proposed. If the clock of LFSR2 is taken as shown in Fig. 4.6, where $a_1, ..., a_5$ are the outputs of LFSR1 or LFSR2. CLK2 will be inhibited to appear whenever all the outputs of LFSR1/LFSR2 are 1. As a result, one bit-shifted versions of the outputs of LFSR2 will appear for onward selection after every 32 bits of the LFSR1. FCSR controls the output of the multiplexer.

![Fig. 4.7 Another modified circuit for Geffe generator.](image)

### 4.6 Parallel Codes for the Encryption of Time Division Multiplexed Signals

Most of the available systems are intended for operation with only single data channel and cannot provide high performance data processing for the multichannel data encryption systems [4.6]. Various techniques have been reported for parallel and secure data transmission. For the encryption of multiple channels of communications in a single box, one can use a different pseudorandom sequence generator for each stream but use of a different pseudorandom sequence generator for each stream requires more hardware and all the different generators have to be synchronized [4.1, 4.7-4.8]. Therefore, it would be simpler to use a single generator for this purpose. Multiple streams can be generated with the help of a single generator by clocking it multiple times. Three independent streams can be generated by clocking the generator three times faster and sending one bit into each stream. This technique works, but for $N$ independent streams, it may have trouble in clocking the generator $N$ times faster. Another technique is to use the same sequence for each channel with a variable time delay, but this method is insecure. A scheme patented by national security agency (NSA) [4.5], which seems to be promising is shown in Fig. 4.8. In this scheme, output of a pseudorandom generator is passed through an $m$-bit serial input parallel out shift register. If there are $n$ messages to be encrypted in parallel $n$ different control vectors are used to generate $n$ different streams (as shown in Fig. 4.8) to encrypt these messages separately. Here also, the circuit is relatively complicated and a simpler scheme may be found preferable. Sometimes cryptographically secure
pseudorandom numbers are not good enough. If an adversary gets a copy of that chosen
generator and the master key, the adversary can create the same keys and break
cryptosystem.

Fig. 4.8 Multiple bit generator.

In Fig. 4.9, a parallel Gold code generator circuit is shown, which gives 5 different
GCs in parallel. These parallel GCs can be used for the generation of five different
parallel keystreams, for the encryption of five different message signals.

Fig. 4.9 Generator of Gold codes (G1-G5).
4.7 Proposed LFSR based Complex Code Generator

Randomness is very important because it destroys statistical properties in the message. Some generators have LFSR clocked at different rates; sometimes the clocking of one generator depends on the output of another [4.9-4.10]. Simple LFSRs generate PN sequences depending upon the feedback tapings 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 tapings. For an LFSR of n-bit if \(2^n - 1\) is a prime number, then number of different m-sequences, which can be generated using different feedback taps, are

\[
\phi = \frac{2^n - 2}{n} \quad (4.1)
\]

This comes out to be a very large number for large values of \(n\). For example, for \(n=5\), \(\phi\) comes out to be 6. Similarly, for \(n=61\), \(\phi\) comes out to be 31,800,705,069,076,960. However, if \(2n+1\) bits out of an m-sequence of \(2^n - 1\) bits are known, the feedback tap 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. In this section, LFSR based complex code generator circuit is given.

4.7.1 Circuit Description

In this scheme (shown in Fig. 4.10), 8-bit shift register is used, which gives \(2^8 - 1 = 255\) bit sequence of 1s and 0s. There are 16 possible feedback taps, which can give maximal sequences. Possible maximal feedback taps for 8-bit shift register are [8,7,6,5,3,2], [8,7,4,3], [8,7,6,3,2,1], [8,6,5,4], [8,6,5,3], [8,7,6,1], [8,6,4,3,2,1], [8,4,3,2], [8,5,3,2], [8,6,3,2], [8,7,3,2], [8,7,6,5,2,1], [8,6,5,1], [8,5,3,1], [8,7,2,1], [8,6,5,2]. Using the circuit shown in Fig. 4.10, one may get a complex code for the encryption of any message. Anyone of the above feedback connections can be selected at a time to generate the corresponding code-sequence in part or in full depending upon the time for which the selected feedback remains connected. If these feedback connections are changed in a synchronous manner, the output sequence is also changed correspondingly. This can be implemented as shown in Fig. 4.10.
Fig. 4.10 The proposed sequence generator: (a) Complete circuit; (b) Details of the feedback network.

For simplicity of the scheme, only seven feedback connections are chosen, which are \([8,6,3,2], [8,5,3,2], [8,4,3,2], [8,6,5,4], [8,6,5,3], [8,6,5,2]\) and \([8,6,5,1]\). One of these connections can be selected at a time with the help of an 8-1 multiplexer (MUX) controlled by a 3-bit word changing in a pseudorandom manner as shown in the above figure. If 000 is to be avoided in the control word (as may be the case in PN-sequence), only 7-inputs of the MUX will be chosen depending upon the control word. Depending
upon the value of $N$ in the circuit, initial state of the code generator and initial state of the multiplexer a complex sequence can be generated, which seems to be very interesting for stream enciphering/deciphering. $N$ decides the length of a particular sequence and the starting state of the next sequence (which will naturally depend upon the starting state of the present sequence also). A lengthier PN-sequence can also be used for providing the control word for the multiplexer and this will advance the complexity of the code words. In such a case, 3-bits of the sequence will appear sequentially as the control word. One more set of taping can be chosen to be connected to the input line corresponding to 000 control word of the MUX.

Output of the shift register as a keystream can be used for the encryption of the message. This keystream is very complex because it is not dependent on contents of the shift register but also on the different feedback taps, which can be selected according to the contents of the PN sequence generator as well as the counter [4.11].

4.8 Proposed Programmable Logic Arrays (PLAs) based Complex Code Generator

Keeping the length of the shift register small one may introduce some logics to increase the length of the keystream as well as the degree of secrecy. In this section, PLA based complex code generator circuit is proposed which can be taken as a Generic PN generator circuit [4.12]. The feedback tapings in this scheme are kept on changing randomly or in a predefined manner, which makes the code generated, quite complex. Logic for taping selection can be implemented by using PLAs as discussed in Fig. 4.11. Simplicity of the circuit along with the complexity of the code generated makes the circuit attractive for low cost secure communication applications.

A possible implementation of the proposed scheme using an LFSR of $n$-bit length is shown in Fig. 4.11. At the time of the starting sequence, the sequence will depend upon the first seed selected. The feedback sequence can be changed at the last bits of the m-sequence. In this case, the seed will remain the same but the m-sequences will depend upon the feedback sequences. Since there are $\phi$ possible feedback sequences, all the sequences can be selected one by one through the logic input for tap selection. However, feedback sequence can also be changed at any moment before the end point of the m-sequence in that case the next sequence will depend upon the seed (output vector of shift register) at the time of the feedback change and therefore, the sequence generator will be quite complex depending upon the instant of the feedback change and the bit sequence of the control input, which can also be controlled through the control bit sequence of the
logic selector. So, in Fig. 4.11, key length is increased by changing different valid feedback tapings with the help of a digital control in a simple or pseudorandom manner. The signal thus generated at the output of LFSR can be made quite complex and suitable for stream enciphering and for other secure communication applications.

![Fig. 4.11 General implementation of the proposed scheme.](image)

Logic for tap selection can be implemented by using PLAs. The number of feedback taps \( k \) decides the number of control bit for PLA and is given as

\[
k = \log_2(\phi)
\]

(4.2)

For high bit rates, delay of the processing circuit will be a bottleneck. If the LFSR is synchronous, the clock duration \( T \) must satisfy the following condition given by

\[
T > (t_a + 3t_x + t_A)
\]

(4.3)

where \( t_a \) is the delay of AND gate, \( t_x \) is the delay of the XOR gate and \( t_A \) is the processing delay of the PLA.

The proposed scheme is tested using 8-bit shift register, which gives \( 2^8 - 1 = 255 \) bit sequence of 1’s and 0’s. The sixteen possible maximal length feedback taps selected for 8-bit shift register are shown in Table 4.1. The implementation of the feedback taping selection logic is carried out with the help of PLAs. If a 4-bit control word is taken, then any 16 tapings can be chosen (a choice of the designer) according to the truth table shown in Table 4.1.
Output of the shift register can be used for the encryption of the message. This key is very complex because it is not dependent on contents of the shift register but also on the different feedback taps, which can be selected according to PLAs truth table with the help of select lines. The control inputs of the PLA for taping selection have been taken from a four-bit counter.

Various randomness tests were also performed which showed random nature of the sequence generated. Further, the sequence generated was applied to encrypt an image and the encrypted image showed white noise characteristics.
References


57