CHAPTER-6

DESFB- DYNAMIC ENCRYPTION SCHEME TO ACHIEVE FORWARD AND BACKWARD SECRECY

Security is a key issue in wireless sensor networks (WSNs). Usually a single key and a static encryption function are used for secure WSNs. This chapter presents a 'Dynamic encryption scheme to achieve forward and backward secrecy (DESFB) in WSNs. The scheme is based on lightweight dynamic encryption function to achieve secrecy by using the concept of forward and backward secrecy in WSNs. The encryption function used in this scheme is dynamic in nature. This function produces different outputs irrespective of the input for different runs. Analytical results shows that presented scheme is more secure than other existing encryption schemes in terms of forward and backward secrecy

Rest of the chapter is organized as follows:

Key issues in secrecy are defined in Section 6.1. Basic difference between key generation v/s key distributions has been presented in Section 6.2. Existing Forward & Backward Secrecy Scheme used in WSNs is describe in Section 6.3. System model used for key generation is explained in Section 6.4 followed by simulation results in Section 6.5. Section 6.6 describes analytical results and performance analysis of the presented system with reference to Forward and Backward secrecy. Finally, chapter has been summarized in Section 6.7.

6.1. KEY ISSUES IN SECRECY

Symmetric key is used to secure wireless sensor networks (WSNs). To avoid the risk of key compromising, this key is updated after a regular internal of time to maintain long time secrecy in the network. Dynamic key is useful as if at some stage an intruder cracks or knows the key, it is compromised up to the next key updation phase only. Once the key is updated, it is no longer useful for hacker. If the key used is updated in every round by the base station then the scheme is known as key distribution. In key distribution scheme, network congestion increases at a very high rate as sensors are very large in number; therefore there is a need for an alternative mechanism to update the key. This alternative

approach is known as key generation. In key generation scheme, key is generated instead of distribution. A secret is used in every round that is helpful in key generation process. This secret is updated by applying a function in each round to produce a new secret. The function for secret updation is static in nature. Drawback of this scheme is that if some intruder knows the secret and the function for key updation in any round then it is very easy for him to hack the network by generating all the secrets used in future or past Function used to encrypt the data is also static in nature, i.e. same function is used every time to encrypt the data. Security is based only on the key that is used in the encryption function. Once this encryption key is compromised, the use of encryption function is meaningless as the cipher text message generated by the authentic sensor is easily converted to plaintext message by the intruder with the help of key.

6.2. KEY GENERATION v/s KEY DISTRIBUTION

Cryptography is used to increase security in WSNs. A key is provided in each cluster. All the cluster members encrypt the sensed data with this key and forward this sensed data to the cluster head for onward transmissions to the base station. Since the key used in a cluster is fixed throughout the lifetime of the network and in case this key is compromised, all the data sent by the members of this cluster gets compromised. Therefore a fresh key is given to every group member to avoid the possible risk of compromised key. Two different techniques are being used to replace this group key in every round, i.e. key distribution and key generation.

In case of key distribution, a fresh key is replaced for every cluster member by the base station. This scheme is not very efficient as the communication cost is very high compared to the processing cost. Therefore, instead of key distribution, a reverse technique is used for key generation. In case of key generation, the key is not distributed, instead it is generated. All the cluster members share a secret and use this secret as a key to update secret using a technique which is shared and known to all the members. This secret is used in next round as a key and again updated using the same technique. The technique of key generation is shown in Figure 6.1 where the second secret is generated with the help of first secret and similarly i^{th} secret is generated by applying a function on $i-1^{th}$ secret.

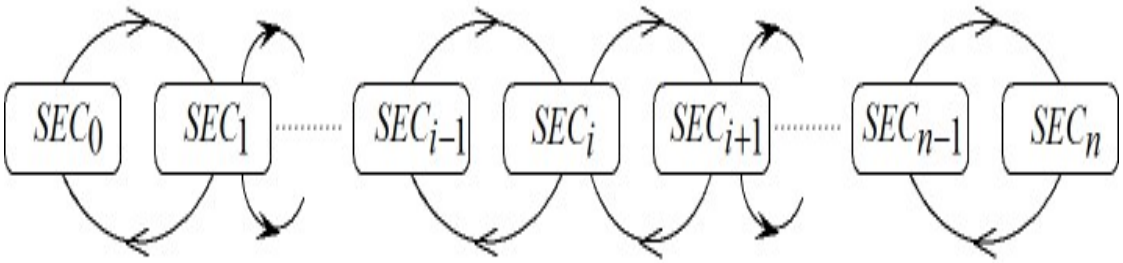

Figure 6.1: Key Generation

6.2.1. Issues in Key Generation

Once the secret of key generation function of any round is compromised, all rounds get compromised. Figure 6.2 shows that if the secret of i^{th} round is compromised, it results in the compromising all the secrets used throughout the lifetime of the network.

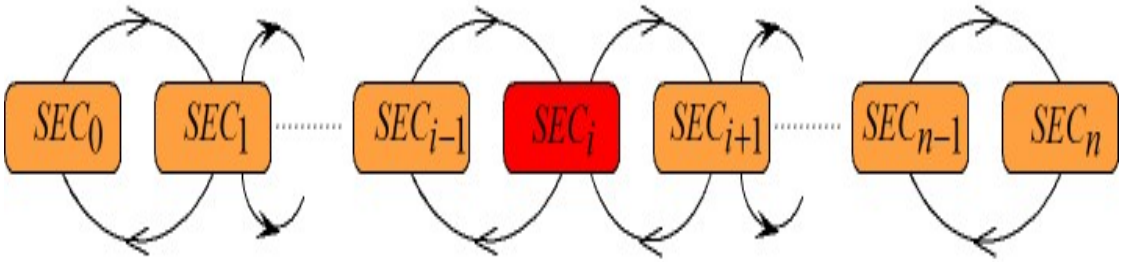

Figure 6.2: Affect of Secrete Compromising in Key Generation

6.2.1.1.Forward Secrecy

The scheme of forward secrecy is used in the process of key generation. According to the principle of Forward Secrecy, the process of key generation is applied in such a way that once any secret is compromised, all the secrets generated in the future also gets compromised. The scheme is shown in Figure 6.3 where the result of i^{th} compromised secret is not disturbing the future secrets but past secrets are unsafe in this principle.

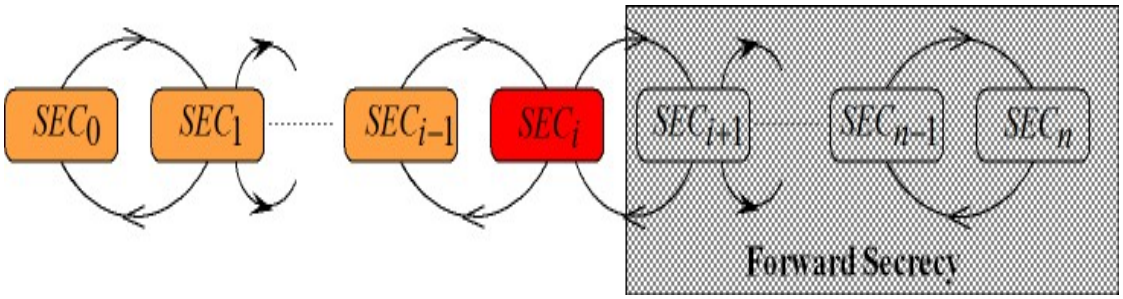

Figure 6.3: Forward Secrecy

6.2.1.2.Backward Secrecy

According to the principle of Backward Secrecy, the process of key generation is applied in such a way that once any secret is compromised, all the secrets that are generated in the past remain uncompromised. The scheme is shown in Figure 6.4 where the result of i^{th} compromised secret is not disturbing the past secrets but future secrets are not safe with this principle.

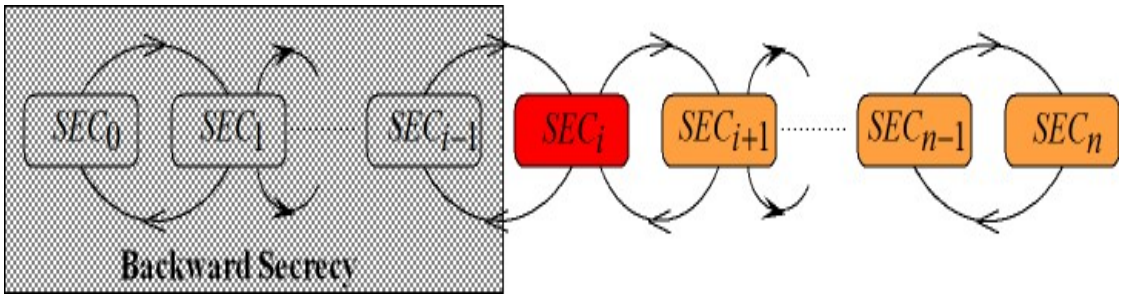

Figure 6.4: Backward Secrecy

6.2.1.3.Forward and Backward Secrecy

To achieve the complete security, Forward and Backward Secrecy is used to avoid the effect of current compromised secret on its past history or on future as shown in Figure 6.5.

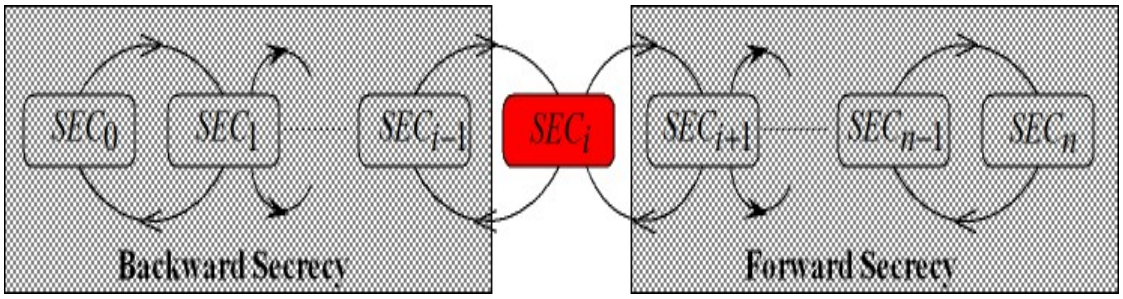

Figure 6.5: Forward and Backward Secrecy

6.3. EXISTING FORWARD & BACKWARD SECRECY SCHEMES

Existing forward and backward secrecy scheme (FBSS) is illustrated with the help of the given example. Consider two sensors A and B with two secrets RA_1 and RB_1 shared with each other. In first round, both generates a key KEY_1 by applying a one way hash function on both of these secrets (RA_1 and RB_1) as shown in Eq. (6.1).

$$KEY_1 = \int_H (RA_1, RB_1) \quad \ldots \quad (6.1)$$

In next round, sensor A generates a random number RA_2 and similarly sensor B generates another random number RB_2. Now sensor A encrypts RA_2 with RB_1 and sends it to sensor B as shown in Eq. (6.2) and similarly sensor B encrypts RB_2 with RA_1 and sends it to sensor A as shown in Eq. (6.3). The complete scenario is shown in Figure 6.6 and Figure 6.7.

$$A \rightarrow B : E(RA_2, RB_1) \quad \ldots \quad (6.2)$$

$$B \rightarrow A : E(RB_2, RA_1) \quad \ldots \quad (6.3)$$

Now both the sensors generate a key KEY_2 to be used in second round by applying one way hash function on both of these secrets (RA_2, RB_2) as shown in Eq. (6.4).

$$KEY_2 = \int_H (RA_2, RB_2) \quad \ldots \quad (6.4)$$

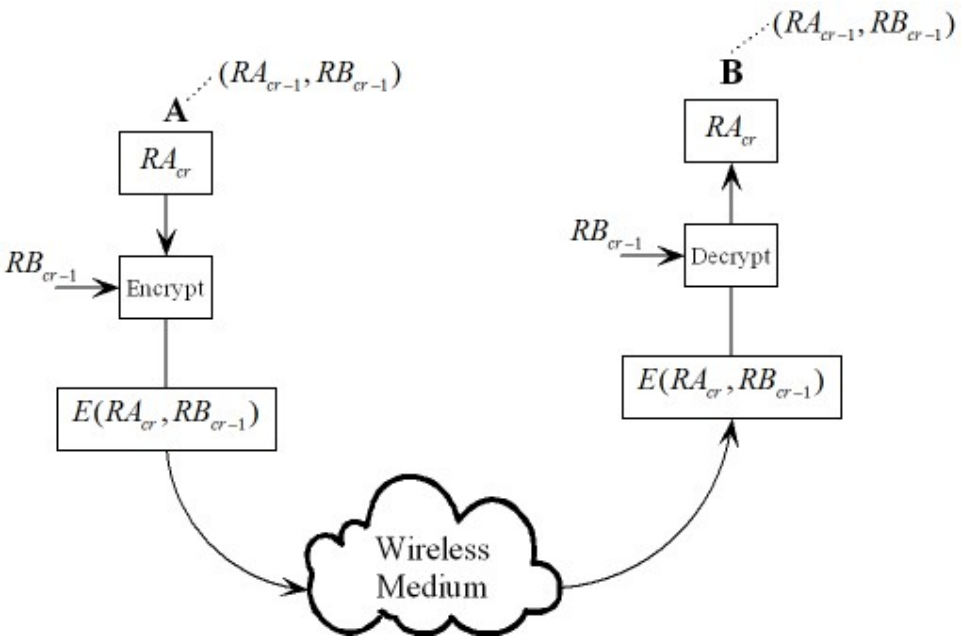

Figure 6.6: Transfer of Key Generation Parameters from Sensor A to Sensor B

In this way both the sensors generate a key used in the current round (KEY_{cr}) with the help of a random number generated in the current round (RA_{cr}, RB_{cr}). The random numbers of current round are exchanged by encrypting these numbers with the random numbers of previous round (RA_{cr-1}, RB_{cr-1}) as a key shown in equation Eq. (6.5) and Eq. (6). Figure 6.6 shows the exchange of random number generated by A sent to B and similarly Figure 6.7 shows the exchange of a random numbers generated for current round between A and B.

$$A \rightarrow B : E(RA_{cr}, RB_{cr-1}) \quad \ldots \quad (6.5)$$

$$A \rightarrow B : E(RB_{cr}, RA_{cr-1}) \quad \ldots \quad (6.6)$$

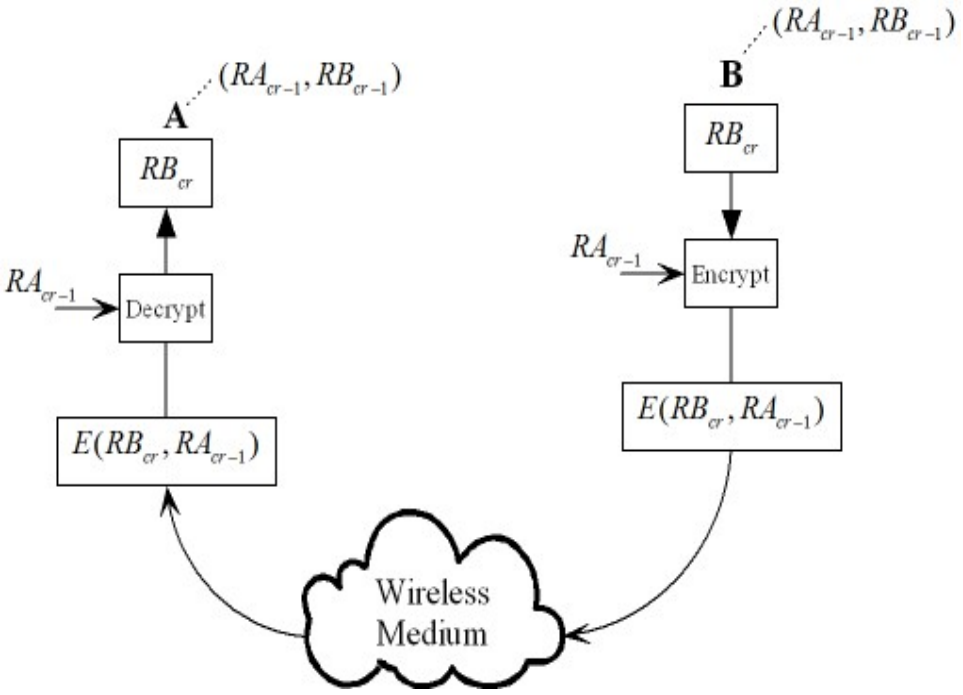

Figure 6.7: Transfer of Key Generation Parameters from Sensor B to Sensor A

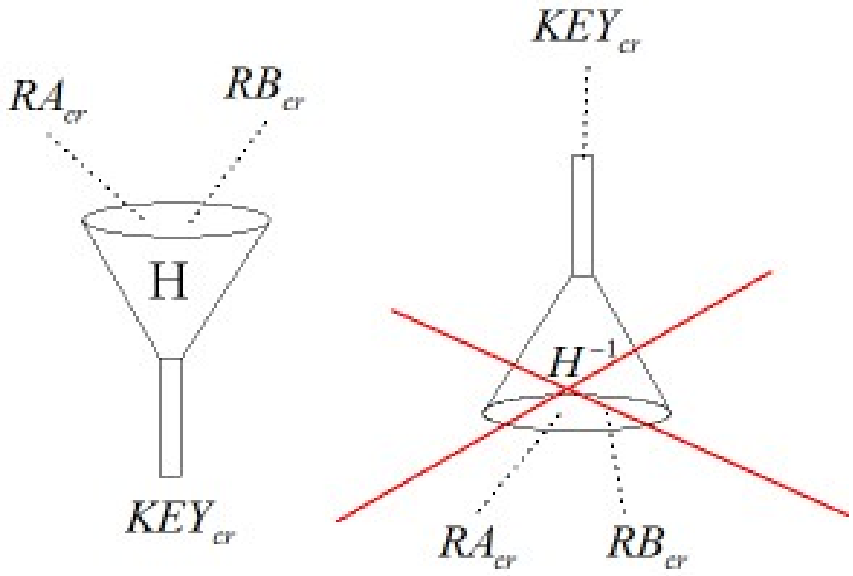

Figure 6.8: One Way Hash Function

The advantage of this scheme is the key generation in each round with the help of a one way hash function. Hash function generates a key with the help of random numbers but it is impossible to generate the random numbers back from the key because the key generation function is one way in nature as shown in Figure 6.8. If some intruder knows the key of current round, then it is impossible to generate the random numbers RA and

RB from this key as the key has been generated using one way hash function. The key is calculated only by the knowledge of key generation parameters i.e. RA and RB.

The principle of forward and backward secrecy is successfully implemented in this work. But the drawback of this scheme is that once the random numbers of any round are compromised, the key of all the rounds are also compromised resulting in failure the principle of backward and forward secrecy. Another drawback of this scheme is that implementation for more than two sensors, the complexity of this system is increased drastically and the complexity of the system for n numbers of sensors (n*n-1) become shown in Figure 6.9.

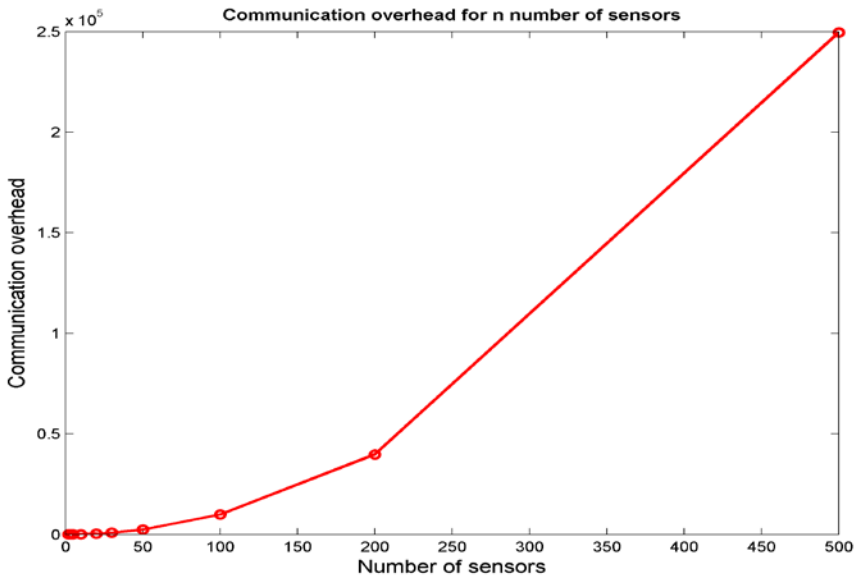

Figure 6.9: Communication Overheads for ‘n’ Sensors in existing System

6.4. SYSTEM MODEL

Presented model attempts to make wireless sensor networks secure enough to ensure the chain secrecy maintained for achieving forward and backward secrecy. This model is based on dynamic encryption scheme (DES) which produces different outputs irrespective of inputs. During the initial phase, numbers of instructions are used for the encryption function. This sequence is known as Initial sequence (IS). All the instructions used in DES are lightweight in nature in the sense that these instructions performs only logical operations or change the sequence of bits stored in the input data. A different round key is generated in each round with the help of VLKM scheme given in Chapter 4. The management of round key is out of scope of this chapter. The round key of current round is used for two purposes: First purpose of this key to generate the instruction

sequence (IS) for current round by applying Sequence Generator (SG) on IS of previous round. The second purpose of this key is to use this in the dynamic encryption function to encrypt the packet used for current round as shown in Figure 6.10.

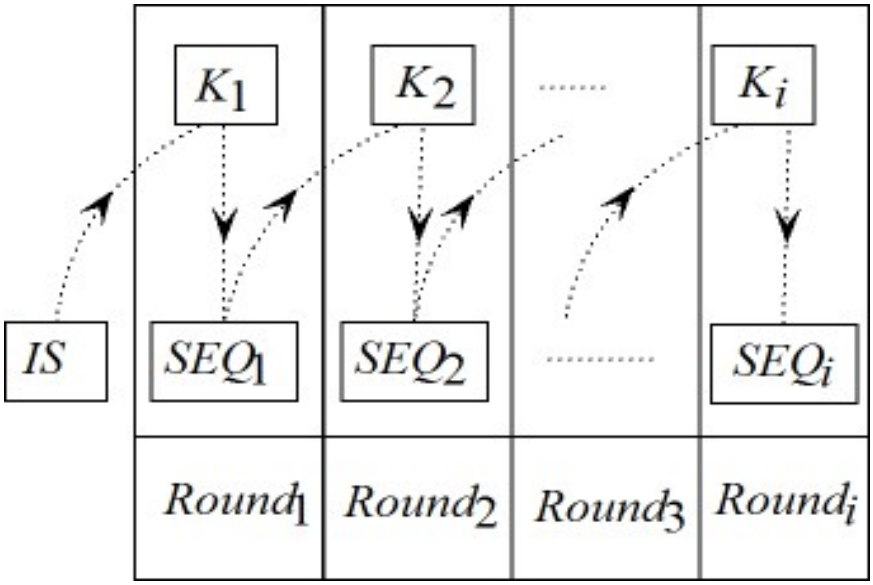

Figure 6.10: Instruction Sequence Generation for Each Round in WSNs

The benefit of this technique is that if in any round a key gets compromised, it is completely impossible to generate the IS of that round without knowing the IS of previous round. Similarly if the IS of the current round is compromised, it is impossible to generate the IS of next round without knowledge of key of next round. Even if all the parameters of security in current round are compromised, still it is not possible to generate the security parameters of next round unless the key used in next round is hacked.

6.4.1. The Architecture

The Dynamic Encryption Function based Scheme (DESFB) to achieve forward and backward Secrecy in WSNs is mainly composed of three modules namely Sequence Generator (SG), Function Builder (FB) and Dynamic Encryption Function (DEF) as shown in Figure 6.11. The SG module generates the sequence of instructions that are executed in the current round. The FB module is used to build the dynamic function from the instruction sequence generated by SG. The DEF module is used to encrypt the packet in each round. A round key which is different in each round and used in DEF to produce the cipher text (CT) in each round. This round key is also used by SG to generate the sequence that is used in each round. This sequence is generated with the help of a key and

the sequence of previous round. In first round the sequence which is used is Initial Sequence (IS).

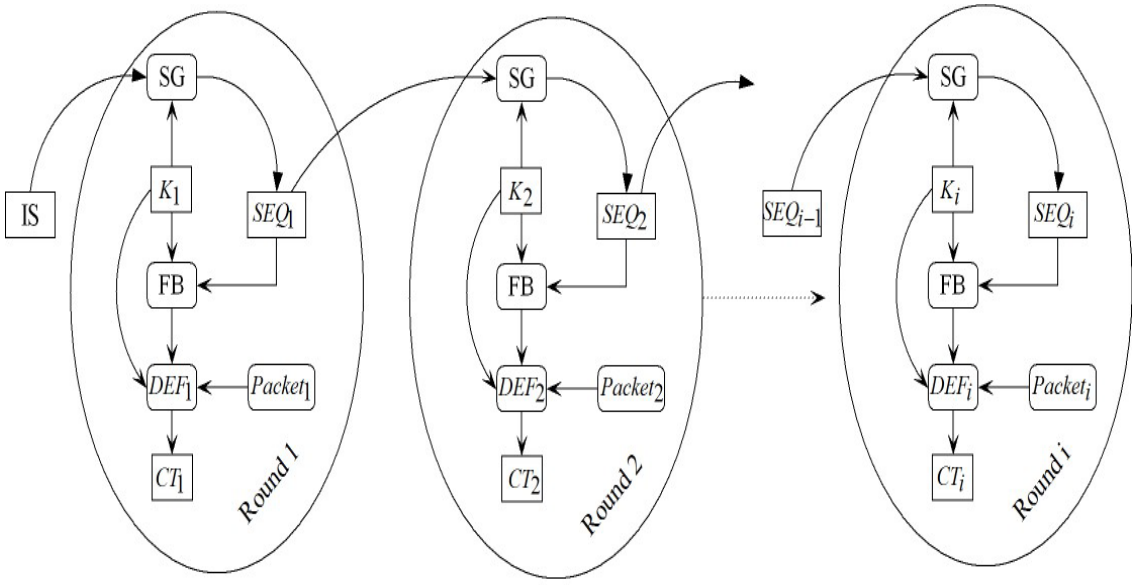

Figure 6.11: A System Overview of Dynamic Function Based Forward and Backward Secrecy Scheme

6.4.1.1.Sequence Generator (SG)

This module takes the instruction sequence of previous round and round key of the current round as inputs and produces the instruction sequence of current round. The number of instructions that are produced by SG are same as that of number of instructions given as input to SG. All the instructions in instruction sequence are unique in nature, i.e. no instruction is repeated in the instruction sequence. The SG generates a temporary sequence (TS) which is flipped to produce the current sequence (CS) for current round, i.e. the instruction at the first position in TS is stored at the last position in CS, second instruction in TS is the second from last position in CS and the instruction at last position in TS is the first instruction in CS.

The instruction sequence in TS depends on the position of bit '1' in round key. The position of round key bit is '1', the corresponding instruction from previous (cr-1) round is a particular instruction in current round (cr). , i.e. let K_q is the q^{th} key bit which is '1' in the round key of current round and $Inst_d$ is the d^{th} instruction in TS that is to be set for current round. The procedure to generate TS and CS are given below in the form of algorithm:

Algorithm 6.1: Sequence Generator (SEQ_{cr-1}, KEY_{cr})

Input: SEQ_{cr-1}, KEY_{cr}
Output: SEQ_{cr}
Where KEY_{cr} is a round key that is used in the current round.
The size of this key is k bits in length
Cr is the current round
TS is the temporary sequence which is initially set to zero

```
1) SEQ (0)=[1 2 3 4 5 6 7 8 9]//Total number of instructions that are used
2) n=length(SEQ(0)); N is the total number of instructions produced by SG (it is fixed in
   each round).
3) k=length(KEY(cr));
4) TS=0;
5) ISloc=1;
6) for j=1:k
   a) if(KEY(cr,j)=='1')//Check the location of bit '1' in key
      i) loc=rem(j,n);
      ii) if(loc==0)
          ◦ loc=n;
          end
      iii) flag=0;//Flag to check weather instruction from previous round at loc is used
      iv) if(TS~=0)
          ◦ for d=1:length(TS)
            • if(TS(d)==SEQ(cr-1,loc))
              (i) flag=1;
              (ii) break;
              end
            end for
          end if (6.a.iv)
      v) if(flag==0)//If this instruction is not used in TS then use it
          ◦ TS(ISloc)=SEQ(cr-1,loc);
          ◦ ISloc = ISloc+1;
          end if (6.a.v)
      endif(6.a)
   end for (6)
7) if(length(TS)<n)//No instruction from previous round should be left without use
   a) a=1:n;
   b) a(TS)=[];
   c) TS=[TS a];
   Endif(7)
8) CS = TS(end:-1:1);//Flip the TS to produce final CS
9) SEQ(cr)=CS;
```

6.4.1.2.Function Builder (FB)

This module takes the instruction sequence (SEQ) and round key of current round as input and produces sequence of instructions (STEPS) to be used in the Dynamic Encryption Function (DEF).

When the instructions given in the algorithm are not executed in a given sequence, the types of functions are known as dynamic encryption functions. The algorithm sequence in the algorithm is shuffled by the FB. FB uses the instruction sequence of previous round and then it performs its operation with the help of a key to produce the instruction sequence for the current round by scanning all the key bits given in the round key one by one through FB. If the scanned key bit at location 'loc' is '1' in the current round key, then the instruction at location $rem(loc, n)$ from the SEQ is added to STEPS otherwise instruction at $rem(loc, n)$ in SEQ is used as dummy instruction. The concept of dummy instruction is used to increase the security of the system.

Dummy instructions are those instructions that are given in the instruction set SEQ but are not executed by the encryption function. In case, some intruder decodes the execution sequence of current round, the calculated results are different than the actual results.

The locations of dummy instructions are not fixed in the algorithm because the execution sequence in the algorithm is shuffled in every round. An instruction which is not executed in the current round appears a dummy instruction for current round and behaves as activated one in next round when it is executed as the usual instruction.

To increase the size of encryption function, all the instructions given in the algorithm are executed more than once in a round robin manner. All the instructions given in the algorithm become as an integral part of an instruction set. Collection of more than one such instruction sets is a complete function to encrypt the data.

In this way, the algorithm is executed more than once depending upon the position of bit '1' in the key bits. Any instruction which is executed in one instruction set may or may not be executed in the next instruction set. An instruction which is executed in all the instruction sets is known as active instructions (AI) of that round. On the other hand, an instruction that is not executed in any of the instruction set is known as dummy instruction (DI) for that round. Rest of the instructions that are executed in some of the instruction set and not in other instruction set are known as Active-Dummy (ADI) instructions of that round.

The total number of instructions produced by FB is equal to the number of key bits in the round key. The total number of active instructions in the DEF are equal to the number of bit '1' in the current round key and total number of dummy instructions in the DEF are equal to the number of bit '0' in the round key. So the number of active

instructions produced by the FB depends upon the number of bit '1' in the round key and similarly location of a particular instruction for DEF taken from SEQ depends upon the locations of bit '1' in the current round key. This feature enables the size of DEF as stretchable, i.e. the size of DEF grows or shrinks with respect to the shape of the current round key. The algorithm of FB to produce shuffled instructions for DEF is given in Algorithm 6.2 below:

Algorithm 6.2: Function Builder (SEQ_{cr}, KEY_{cr})

Input: SEQ_{cr}, KEY_{cr}
Output: Encrypr ($STEPS_{cr}$)

1) j=1
2) n=length(SEQ(cr));
3) for i=1:length(KEY(cr))
   a) if(KEY(cr,i)=='1')
      i) t=rem(i,9);
      ii) if(t==0)
         ◦ t=n;
         end if (2.a.ii)
   b) STEPS(a,j)=SEQ(cr,t);
   c) j=j+1
   end if (3.a)
end for(3)

6.4.1.3.Dynamic Encryption Function (DEF)

This module takes the packet sent by a particular sensor to the cluster head or to the base station and round key of current round as input and produces the encrypted packet ready for onwards transmission.

The function used in this system is dynamic in nature and hence has the capability to generate different outputs for the same inputs as shown in Figure 6.12. The dynamic nature of the function is achieved by changing the execution sequence of instructions stored in the function through FB. The sequence of instructions to be passed over the function is shuffled in each round. Figure 6.12 shows the difference between static and dynamic function. The function is run for three different rounds to encrypt the same plain text p1 encrypted with same key k1. In case of static function, each time the output in the form of cipher text C1 is produced but in case of dynamic function, each time, a different output is produced, i.e. C1 in first run, C2 in second run and C3 in third run.

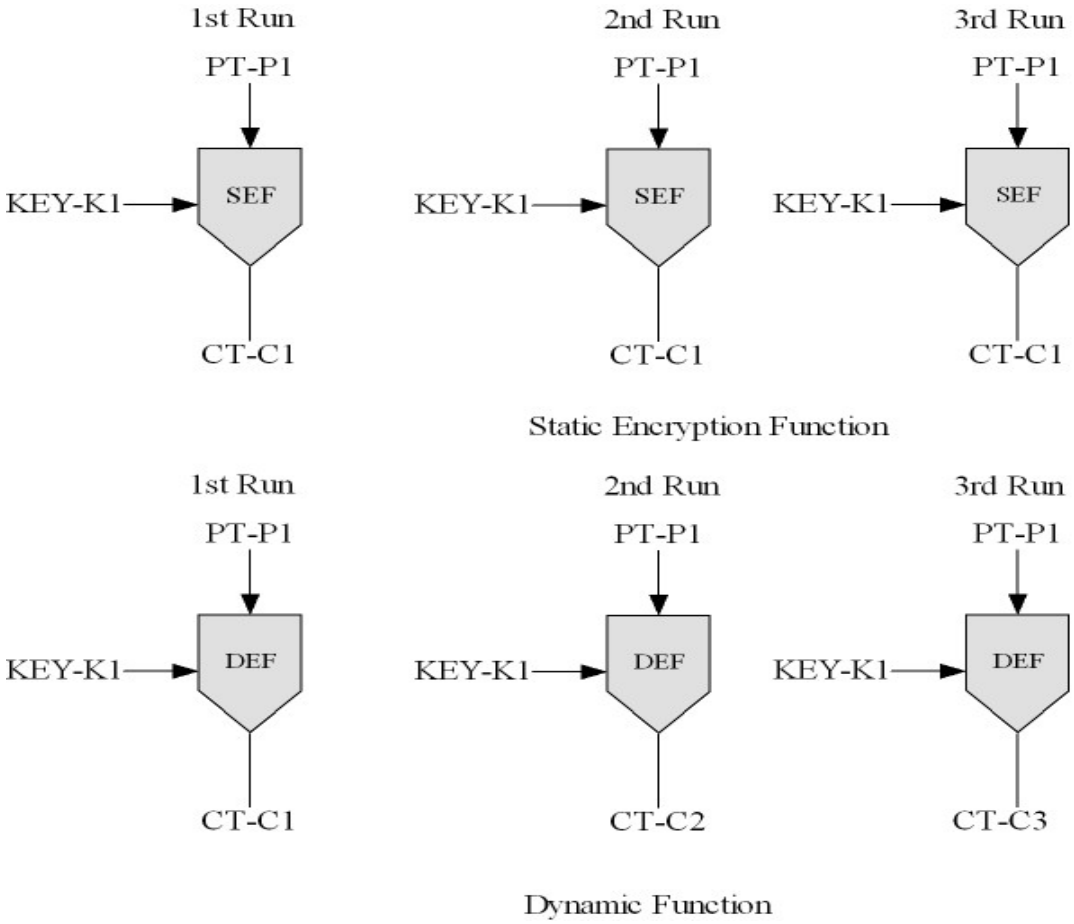

Figure 6.12: Static V/S Dynamic function

The main characteristic of this function is that the instructions used in this function are lightweight in nature because all the instructions have to perform either logical operation or packet bit shifting. All those instructions using power function or multiplication & division operations are known as heavy instructions because these operations requires several more operations to calculate the final results. For example, the power function requires several addition operations and as we know that sensors are resources constraint especially in terms of memory, energy and processing power; therefore the use of power functions need to be avoided as far as possible in wireless sensor networks.

In this function, only those instructions are used that performs logical functions such as one's complement, XOR, NOR, CLS, CRS, two's complement etc. and these instructions are reversible also because decryption is also need to be performed at the receiving end. An example for encryption function with lightweight instructions is given below in Algorithm 6.3.

Algorithm 6.3: One stage of Dynamic Encryption Function

Input: $STEPS_{cr}$, KEY_{cr}, $Packet_{cr}$
Output: Encrypt ($Packet_{cr}$, Key_{cr})

STEP1: change the packet byte order
STEP2: calculate once complement of the entire packet
STEP3: circular left shift the packet bits
STEP4: Exchange pair digits in packet
STEP5: Cross every byte (First four bits in a byte are exchanged with last four bits in the same byte)
STEP6: Circular left shift every byte of the packet
STEP7: XOR current packet bits with original packet bit// If role of round key is included then XOR with key
STEP8: circular right shift the packet bits
STEP9: Complement the skip bits

Dynamic Encryption Function has been described with the help of an example to explain the steps in detail. The example is given in 2 byte packet with the content '1011000101101110'. The steps are executed in sequence, i.e. the output of first step is given to second step, and the output of second step is given to third step and so on.

STEP 1. In this step, the packet byte order is changed. The packet is read from least significant bit (LSB) to most significant bit (MSB). After performing the operation the new content of the packet becomes '0111011010001101'.

STEP 2. In this step, the packet bits are complemented, i.e. each 1 is replaced by 0 and similarly each 0 in packet is replaced by 1. If this step is performed on the output of first step, then the content of packet after this step becomes '1000100101110010'.

STEP 3. In this step, the operation performed on packet is circular left shift (CLS). In this step, the packet bits are rotated circularly, i.e. if MSB is marked as first bit and LSB is marked as last bit then first bit is shifted to the last bit and all the remaining bits are shifted to left of its left position, i.e. bit at i^{th} position is shifted to $i-1^{th}$ position except MSB and MSB is shifted to LSB. After performing the operation on output of step 2, the new content of the packet becomes '0001001011100101'.

STEP 4. In this step, the operation performed on exchanged pair digits. In this step, the packet bits are divided into pairs. The first bit is paired with second bit; third

bit is paired with fourth bit and so on. Therefore, every i^{th} bit is paired with $i+1^{th}$ bit if i^{th} bit is at odd position otherwise it is paired with $i-1^{th}$ bit. Now both the pair bits are exchanged. After performing the operation on output of step 3, the new content of the packet becomes '0010000111011010'.

STEP 5. In this step, the operation performed on packet is cross every byte. In this step, the packet is divided into group of 8 bits, i.e. this operation is performed on every byte of the packet. Now first four bits in every byte are exchanged with last four bits in that byte. After performing the operation on the output of step 4, the new content of the packet becomes '0001001010101101'.

STEP 6. In this step, the operation performed on packet is circular left shift (CLS). In this step, the packet is divided into group of 8 bits, i.e. this operation is performed on every byte of the packet. Now the group bits are rotated circularly, i.e. if MSB is marked as first bit and LSB is marked as last bit then first bit is shifted at last bit position and all the remaining bits are shifted to its left position, i.e. bit at i^{th} position is shifted to $i-1^{th}$ position except MSB and MSB is shifted to LSB at last position. After performing the operation on the output of step 5, the new content of the packet becomes '0010010001011011'.

STEP 7. In this step, the operation performed on packet is XORing the current packet bits with original packet bits. If role of round key is included in the function then XORing is performed on current packet bits with round key. After performing the operation on the output of step 6 with XOR to original packet bits, the new content of the packet becomes '1001010100110101'.

STEP 8. In this step, the operation performed on packet is circular right shift (CRS). In this step, the packet bits are rotated circularly, i.e. if MSB is marked as first bit and LSB is marked as last bit then last bit is shifted at first position and all the remaining bits are shifted to its right position, i.e. bit at i^{th} position is shifted to $i+1^{th}$ position except LSB and LSB is shifted to MSB at first position. After performing the operation on output of step 7, the new content of the packet becomes '1100101010011010'.

STEP 9. In this step, the operation performed on packet is skip packet bits. All the packet bits at odd positions are complemented, i.e. each 1 at all the odd positions is replaced by 0 and similarly each 0 at all the odd positions are

replaced by 1. If this step is performed on the output of step8, the new content of the packet becomes ‘0110000000110000’.

The function used is dynamic in nature as the sequence of instructions in function depends on the previous execution sequence. The new order of sequence for the function is generated for the current round on the basis of previous execution sequence and finally, this new sequence is shuffled.

Even if some intruder knows the SEQ of previous round, it does not generate the SEQ of current round without decoding or hacking the round key of current round. Therefore, to hack the system for one round, it is necessary to hack the SEQ of one round along with the round key of next consecutive round. To hack the data of each round, it is necessary to hack the SEQ and round key of each round. Otherwise it is completely impossible to break the security channel of the presented system.

6.5. SIMULATION RESULTS

To understand the behavior of dynamic function, 56 bit packet is used as an input to produce the outputs in different rounds. The sequence is generated for each round with the help of a sequence of previous round and a key of current round. Then according to the key bit, steps for dynamic encryption function (DEF) are generated for each round and these steps are executed in sequence. The final output is produced for each round as a cipher text. In each round same packet is used as an input to the DEF. Two scenarios are created as given below

a) Same packet but different key
b) Same packet and same key

a) Same packet with different key

In this scenario, in each round the same packet is used as an input but a random key of 56 bits is used in each round. This key is different in each round. To include the role of round key in step 7, the packet at that stage is XORed with the key bits of round key. Simulation results of this scheme are shown in Table 6.4. Results prove that the produced cipher text is different in each round.

PACKET='10010101000111100111011001011001110000111110101110000001'

Table 6.1: Simulation Results for Different Random Key and Same Packet Input in Each Round

KEY	ROUND	SEQUENCE	STEPS	CIPHER TEXT
0100111000011001000101100110100111111011100111001110101 0	1.	1,2,3,4,5,6,7,8,9	2,5,6,7,3,4,7,2,4,5,8,9,2,5,6,8,7,8,9,1,3,4,5,8,9,1,4,5,6,8,1	00101100100110000010100100001111001111111111001010011001
10000001101110111110101010011100011011000101010101100110	2.	5,7,6,4,3,2,9,8,1	5,8,1,7,6,4,2,9,8,1,5,6,3,9,5,7,6,9,8,5,7,2,8,5,6,3,2,1,5	00000011000010011101110110110000011100001100110111110000
10011011011010001101100110000110010100010000011010001010	3.	2,6,1,7,8,9,3,4,5	2,7,8,3,4,2,6,7,4,5,6,1,9,3,1,7,3,5,7,2,6,7,4,2	00010010100001101001001001011010000101010000011010101101
10110001100001101010110110010111001010001110101101001110	4.	6,3,9,8,5,4,7,1,2	6,9,8,1,2,5,4,1,6,9,8,4,7,6,9,8,5,1,6,5,4,7,2,3,9,5,1,2,6	01000110111110001000010011001100000111111001001100110111
10101110001110111001001101101100001100101110101000000111	5.	2,1,8,3,7,4,5,9,6	2,8,7,4,5,1,8,3,4,5,9,1,7,4,9,6,1,8,9,6,8,7,4,5,6,1,6,2,1	10010110101111011101101011101111011011010010011010111100
10011101111111000000101000010001111110111101001010110000 0	6.	5,8,1,6,9,4,7,3,2	5,6,9,4,3,2,5,8,1,6,8,6,2,6,9,4,7,3,2,8,1,6,9,7,5,1,9,4	10110100111001011010101111010111001001111100011101101100
01100100110100111110101101111010010010001100100000000000 10	7.	6,9,3,7,5,2,4,1,8	9,3,2,8,6,3,2,4,1,8,6,3,5,2,1,8,6,3,2,8,7,5,1,6	00110111110000101101101001100111111000000010100001000100 11
10010111011111011101001010110100010010111110000010111100	8.	8,5,3,9,1,4,2,7,6	8,9,4,2,7,8,5,3,9,1,2,7,6,5,1,2,6,8,3,2,8,3,9,1,4,2,9,4,2,7,6	00001010011111001011010000101101000010101111010110001011
01111001101100111110101110101100110001111010110011001110	9.	8,2,4,6,7,1,9,3,5	2,4,6,7,3,5,2,4,1,9,3,5,8,4,7,1,9,5,2,4,1,9,2,4,6,7,9,5,8,6,7,3,5,8	01100101011000010000011001101100011000010000010111011001
00011100100100000101010100010000000111111001110111100001 1	10.	9,3,8,2,4,5,1,7,6	2,4,5,6,8,6,3,2,5,9,7,6,9,3,8,2,1,7,6,3,8,2,4,9,3	10100000010011000010101010001000001110100011100100011100 10
00100010011010000000001101111101101010011001011110000101 10	11.	5,6,7,4,2,3,9,1,8	7,9,5,6,4,4,2,9,1,8,5,7,4,3,1,6,7,3,1,8,5,6,9,8,5	10101010001010101100110101111101011001100001010110011111
00101010101011100110001101110111011111101110011110111101	12.	1,5,3,4,6,8,9,2,7	3,6,9,7,5,4,6,8,7,1,6,8,2,7,1,3,4,6,9,2,7,1,5,3,6,8,9,1,5,3,4,8,9,2,7,5	10011000110110110011100110100101101111011111101100101110
00011010101111110100110110001111111111000100000000111101 10	13.	1,2,8,3,5,7,9,6,4	3,5,9,4,2,8,3,5,7,9,4,8,3,7,9,2,8,3,5,7,9,6,4,1,2,7,3,5,7,9,4,1	00100011100100101011001000100100001111000011001100011000
00010101101100100111110101010010000111000010011110101010	14.	5,9,1,8,2,4,6,7,3	8,4,7,3,9,1,4,3,5,9,1,8,4,7,5,8,3,5,9,6,5,9,1,8,4,7,5	10000010001000010100000010110101000111010001010011111101
10010110110101100101111011100011110010110110011011011000	15.	7,9,2,1,3,6,4,8,5	7,1,6,4,5,7,2,3,6,5,9,2,1,3,4,8,5,1,3,6,4,7,2,1,6,4,7,9,1,3,4,8	01011011001110010011101100110011111111010101111100001001
10011100110111010101011111001100011111001110010000001001 1	16.	8,9,4,2,5,6,3,1,7	8,2,5,6,7,8,4,2,5,3,7,9,2,5,6,3,8,9,6,3,1,7,8,2,5,6,7,3,8,9	11110010110010110000101001111010011001100001111100111110
10100010011100010111110111110101010100010100100101011110 1	17.	5,1,6,2,7,9,3,4,8	5,6,3,5,1,6,3,8,5,1,6,2,9,3,4,8,1,2,9,4,6,7,4,1,2,9,3,4,8,1	01001100111100100111011101010100011000001001100100000111
00000011101100100101 1	18.	5,2,7,9,6,	8,4,3,2,7,1,3,2,7,6,1,7,9,	111001011101000011110 1

011000001111001010010 00110011101011		1,8,4,3	6,1,3,2,6,3,5,9,6,1,4,5,2	101100101000000100001 0001100000000
110100100000111010100 100100000000010110101 11001101011001	19.	3,7,4,1,6, 8,9,2,5	3,7,1,9,1,6,8,2,3,1,9,2,3, 7,1,8,9,2,7,4,6,9,2,7	000100001111010101101 1101100101001011010100 001001110100
101000110010110010000 111000011011111100010 10000001100110	20.	5,8,6,1,7, 2,9,4,3	5,6,9,4,8,1,7,4,1,7,2,8,6, 7,2,9,4,3,5,7,9,7,2,3,5	100001110111010110100 0 1101010100111101001111 100101110010
000111001111100111001 000010010101001001011 10010100101000	21.	4,9,6,8,5, 3,2,7,1	8,5,3,1,4,9,6,8,2,7,1,6,7, 9,8,3,1,6,5,3,2,4,6,3,7	000111110011110110001 1 001011011010111001000 0 101011111111
011111001010100110111 101111011010111100001 01110001100011	22.	4,7,2,1,3, 5,8,6,9	7,2,1,3,5,9,7,1,8,6,4,7,2, 1,5,8,6,9,7,2,3,8,6,9,4,5, 6,9,4,3,5,4,7	111001000111000111111 1 011010000001111111111 0 001110111011
111000100101001110001 010000110110000110101 00111010010110	23.	9,1,3,6,5, 8,2,7,4	9,1,3,2,9,3,8,2,7,3,5,9,1, 6,5,9,1,6,8,4,9,1,6,2,4,9	001100000100100110010 1 010111011011111100111 1 111100000000
010011111111100100001 110011101000011001100 01101001111010	24.	6,3,9,4,7, 2,8,5,1	3,7,2,8,5,1,6,3,9,4,8,9,4, 7,5,1,6,9,5,1,9,4,5,1,3,7, 2,8,5,6	100100000110101010101 0 100101100100001010101 0 110100110010
111110001011100100101 000010100001011101000 01000010001111	25.	2,5,8,1,7, 4,9,3,6	2,5,8,1,7,6,5,8,1,9,2,8,3, 2,4,3,6,2,8,3,1,3,6,2,5	001010101100101111010 0 101111011100011101011 0 011101011111
100011101001000100011 010100100010010101001 11100111000010	26.	1,3,5,8,6, 9,4,7,2	1,6,9,4,2,5,4,3,5,6,4,1,6, 7,1,5,9,4,7,2,5,8,6,1	101111000000010001101 0 000011100010110011000 1 111101000110
000010100001110011100 001011110110101110100 00010110011000	27.	3,9,1,2,7, 8,5,4,6	7,5,1,2,7,4,6,3,8,4,6,3,9, 2,7,5,6,3,9,2,3,1,2,5,4	010101100001111111001 1 1000010100000100001001 1 10101001111
111111011100111110100 111001101110001110110 00011101010110	28.	5,6,4,8,7, 2,1,9,3	5,6,4,8,7,2,9,3,5,8,7,2,1, 9,5,8,7,2,3,5,4,8,7,3,5,6, 8,7,5,6,4,7,1,3,5	000100011011101110000 1 000011101010111001011 1 10101100010
111111001011001000010 011111111011010011100 11000110011100	29.	9,1,3,2,7, 8,4,6,5	9,1,3,2,7,8,5,1,3,8,1,7,8, 4,6,5,9,1,3,7,8,6,1,3,2,4, 6,3,2,4,6,5	100101100000111101110 1 010100101001111111110 0 10110011001
011111100001100100010 110011010011111101110 01110011101010	30.	9,5,6,4,8, 7,2,3,1	5,6,4,8,7,2,6,4,2,5,4,8,3, 1,5,8,7,2,3,1,9,6,4,8,3,1, 9,4,8,7,3,9	100111000110000100100 0 110000110110010001001 0 100010011001
100000101100011101111 010010011110101010101 10111001111100	31.	4,3,6,5,7, 8,1,2,9	4,1,9,4,7,8,1,9,4,3,6,7,2, 3,6,5,7,1,9,3,5,8,1,9,4,3, 7,8,1,2,9	101111110000100000000 1 011010111000001011001 1 10011011100
111011100000000111000 101010110100111011100 01101000001010	32.	5,9,2,1,8, 7,6,3,4	5,9,2,8,7,6,6,3,4,1,7,3,5, 9,1,6,3,4,9,2,1,3,4,9,3,5	010101001110011011001 0 111110101000010011010 1 00010010010
001100001111110110111 111010110101101100101 11101111011111	33.	3,8,6,7,9, 5,4,1,2	6,7,2,3,8,6,7,5,4,2,3,8,6, 7,9,5,1,3,8,7,5,4,2,3,7,5, 4,1,2,8,6,7,9,4,1,2,3,8	101101010110100001000 1 110110111000000011010 1 001111011101
101110111010001100101 011100001111111100100 10110100001100	34.	5,8,2,1,4, 9,7,6,3	5,2,1,4,7,6,3,8,9,7,5,2,4, 9,7,2,1,4,9,7,6,3,5,1,7,3, 5,2,6,3	010011100001100100000 1 101101100001000110011 1 00110101101
011110010001000111001 000001011110111010000 00010010100011	35.	9,5,3,7,6, 4,1,2,8	5,3,7,6,2,3,1,2,8,3,8,5,3, 7,6,1,2,8,9,3,9,7,4,9,5	100011110001011011110 1 000100110001100110000 1 101111110010
101010100010011010001 111100111000001111101 01000001110000	36.	8,7,2,4,5, 1,6,3,9	8,2,5,6,7,5,1,3,2,4,5,1,6, 8,7,2,9,8,7,2,4,1,3,5,1,6	001111011000101100111 1 101000010011000101111 1 011101010001
111000111010110101010 000101011000001101010 1 00111111111100	37.	1,5,4,9,3, 6,2,7,8	1,5,4,2,7,8,5,9,3,2,8,5,2, 8,5,4,7,8,5,9,6,8,1,5,4,9, 3,6,2,7,8	010111101000100011000 0 101001101111010110100 0 101000101101

000101000010100111111 101001101111011100000 00101000101101	38.	3,4,1,8,7, 2,5,6,9	8,2,4,8,5,6,9,3,4,1,8,2,9, 3,1,8,7,2,6,9,3,9,4,2,6,9, 4	101110100110110000011 100000100101110100110 1000011001000
111110100101101111001 101001000110011111001 10111100010111	39.	9,6,2,5,7, 8,1,4,3	9,6,2,5,7,1,9,2,5,8,1,4,3, 2,5,8,3,5,7,4,3,9,6,2,8,1, 3,9,6,2,1,3,9,6	000110111100110111011 011011111111101001111 0110111001011
100011011101010000100 101010101001101001100 00101100111011	40.	6,1,5,2,3, 4,8,7,9	6,3,4,7,9,6,5,3,6,2,4,7,6, 5,4,8,9,5,2,9,1,5,4,8,7,6, 1	100100111011100100001 100011001010001100110 011010001001
000100000001100110001 011100001010111001101 10110001101001	41.	1,6,9,4,3, 7,8,5,2	4,9,4,8,5,9,3,7,8,9,3,8,5, 2,9,4,7,8,2,1,3,7,5,6	101100110110110010011 010011011010001011100 001101000101
000110001000011000010 010011111100110000001 10111111011101	42.	8,9,1,5,6, 7,2,3,4	5,6,4,6,7,9,6,3,4,8,9,1,5, 2,3,7,2,4,8,9,1,5,6,2,3,4, 9	011101011011001000110 101001111100100100010 0111011011100
000010010101110000011 100111000010001101001 10100011000100	43.	7,4,2,9,5, 1,8,3,6	5,3,7,2,9,5,4,2,9,8,3,6,5, 6,7,2,1,8,6,9,5,6	110111011111111011011 011111000110111000101 1000011100111
101001111100101011010 111000110001010111100 11111011010000	44.	5,4,9,6,3, 8,1,2,7	5,9,8,1,2,7,5,6,8,2,7,4,6, 3,8,5,4,8,2,5,4,9,6,1,2,7, 5,4,6,3,1	101111100101100000001 100111111010101010000 0001011000000
101010111011100000000 010010110001111000110 00000000100010	45.	8,6,4,7,2, 1,3,9,5	8,4,2,3,9,5,6,4,7,2,9,8,6, 1,3,9,5,7,2,1,8	101010100000110001110 100111100001101101101 0001011101 01
011100101110110110111 100011111111000100001 01001001000011	46.	1,9,2,8,5, 3,7,4,6	9,2,8,7,6,1,9,8,5,7,4,1,9, 2,8,4,6,1,9,2,8,5,3,1,3,4, 9,5,1,9	011110101100100111011 100010101000001001001 1000000010000
110010010000000111011 110111100000000010101 11011101111011	47.	3,8,2,6,7, 4,5,9,1	3,8,7,9,5,9,1,8,2,6,7,5,9, 1,3,8,6,4,5,9,3,8,2,7,4,5, 9,3,8	110111011100011111010 0 111010010111110100010 0 0001101011 00
101000110010100000010 110100000100111101101 00000011011011	48.	4,1,7,6,8, 9,5,2,3	4,7,5,2,1,6,1,6,8,5,6,5,2, 3,4,7,6,9,6,8,5,2,4,1	111011110101111000011 000101000011100000100 0 011110110100
000110001000011110110 110101011010010000110 01111100001000	49.	7,1,4,2,5, 9,3,8,6	2,5,6,5,9,3,8,7,1,2,5,3,6, 1,4,5,8,2,5,8,6,7,1,4,8	111011001111011111010 01 100100000111011010000 0 100110000110
101010100000011011100 111110011010011000111 10011101001100	50.	1,2,6,8,9, 3,5,4,7	1,6,9,5,9,3,4,7,1,8,9,3,5, 4,2,6,9,4,7,8,9,3,5,1,2,6, 9,4,7	010000000011010001101 1 010111001001100101110 0 101110011011
001110111001011100100 110111110110100011011 00000000101011	51.	2,1,3,7,4, 5,9,8,6	3,7,4,9,8,6,3,4,5,9,2,7,4, 9,8,6,2,1,7,4,9,1,3,4,5,5, 8,2,1	001111110000000000100 01 000101101100111110101 0 010011111001
010110001110000001010 101100000111001000101 01110001101100	52.	3,8,9,5,2, 6,4,7,1	8,5,2,1,3,8,1,8,5,6,4,5,2, 6,1,5,6,7,1,3,2,6,7,1	001001110010011110101 0 101011101000101010110 1 001010001010
011111011000101111110 010001001011111001000 11110101110100	53.	3,4,1,7,6, 2,5,9,8	4,1,7,6,2,9,8,7,2,5,9,8,3, 4,6,8,1,6,2,5,9,8,1,5,9,8, 3,1,6,2,5,8	011100110010011011110 1 100101110110101101010 0 111100111010
001111011010010010001 100000001100010101111 10000111100110	54.	5,3,4,8,9, 2,6,7,1	4,8,9,2,7,1,3,9,7,4,8,4,8, 7,5,4,8,9,2,6,4,8,9,2,1,5	001101110010110010001 1 001010011000001001010 0 101111101100
010111011011011110111 101010111000011100000 00001101000010	55.	5,6,4,1,7, 2,9,8,3	6,1,7,2,8,3,6,4,7,2,9,8,5, 6,4,1,2,8,5,6,4,8,3,5,6,4, 7,5	100100111111000101101 0 011001101001101101110 0 100001100111
100101010000101001001 010111001111101010111 11100010110111	56.	6,9,7,4,3, 8,2,1,5	6,4,8,1,4,8,5,7,3,2,1,5,7, 4,3,8,2,5,9,4,3,8,2,1,5,4, 8,2,5,6,9	110001110101000110000 1 111000111111111001010 1 011011101001
010000110100011011001 011011110000010111101	57.	4,7,5,8,3, 6,1,2,9	7,1,2,4,3,6,2,9,5,3,6,2,9, 4,7,2,4,7,5,8,6,1,2,9,8,3,	010001000011010100000 0 100110000111011100000

11100011000111			9,4,7	101000000110
101111011001000010101 111000111001101010110 11011111000111	58.	1,7,9,2,6, 3,8,5,4	1,9,2,6,3,5,4,9,5,1,9,2,6, 3,1,7,9,3,8,4,7,2,6,8,5,1, 7,9,2,6,4,1,7	000111101011011100000 100000100110011111100 111100011001
000101101110000111001 010001000100110111100 00001111001000	59.	6,9,5,7,1, 4,8,3,2	7,4,8,2,6,9,8,3,2,5,1,2,7, 8,3,6,9,5,7,9,5,7,1,3	100011001100100100110 000101110100111000011 0101000101110
100011000100100100011 110110100010100100100 01111111101000	60.	2,3,5,9,8, 7,4,1,6	2,8,7,2,9,4,3,5,9,8,4,1,2, 8,4,2,9,1,6,2,3,5,9,8,7,1	011110011101011001100 0011001110011101100000 000001011000
010010110011111110100 000100000000001110100 01011001100110	61.	6,2,7,9,5, 1,4,8,3	2,5,4,8,2,7,9,5,1,4,8,6,4, 3,6,2,9,8,6,2,5,1,3,6	111000101110111110100 1001000011001001111111 110001001010
000101100111111000101 1 000010101000111001001 00010011000101	62.	8,3,5,7,2, 6,4,1,9	7,6,4,8,3,5,7,2,9,3,5,1,8, 5,4,1,9,5,6,8,7,2,9,3	001110001100011111010 1 110101111101011110110 100111011010
010011111111111110011 001010110101101111110 01110100011010	63.	7,5,8,9,1, 4,6,2,3	5,1,4,6,2,3,7,5,8,9,1,4,6, 2,5,8,4,2,7,5,9,4,6,3,7,5, 8,9,1,2,3,7,8,6,2,7	001001010010101010011 0 100110111001111011001 1 100110000000
010111000110100101000 110001101111000001000 10101010100000	64.	2,8,3,6,7, 4,1,9,5	8,6,7,4,2,8,6,1,5,6,7,5,2, 3,6,7,4,3,1,5,8,6,4	101000111101001101100 1 101011110101011001111 0 011110110010
011101110001100011101 001101110111000000111 11100110010010	65.	7,2,5,9,1, 4,6,3,8	2,5,9,4,6,3,5,9,3,8,7,5,4, 6,8,7,2,9,1,4,9,1,4,6,3,8, 5,9,6,7	110100000101011000010 1 110100100110101001111 0 010101110110
101111100100100011100 111100111110111010101 01010100001110	66.	2,8,3,6,4, 1,9,5,7	2,3,6,4,1,9,2,6,5,7,2,6,4, 1,9,2,8,3,6,4,9,5,7,8,6,1, 5,2,3,5,7,2	011101111000010110101 0 110000000101110100101 1 010101000010
001100001100011101000 010000100101000011100 11011000110010	67.	5,8,9,1,4, 2,7,6,3	9,1,3,5,4,2,7,3,4,5,1,2,8, 9,1,7,6,5,8,2,7,5	011110100100010101010 0 111111101011001101010 1 01011011101
111010010100111010000 110110011001101110110 00111110101011	68.	3,7,2,1,6, 4,9,8,5	3,7,2,6,8,3,1,6,4,8,1,6,9, 8,7,2,4,9,5,3,7,1,6,5,3,7, 2,1,4,8,3,7	000100101100100001001 0 011001110011101001100 1 001110101110
101000110010111000101 010101011110111010011 10100101000111	69.	5,4,6,1,7, 8,9,2,3	5,6,9,2,4,1,7,8,5,6,7,9,3, 4,6,1,7,9,2,3,4,7,8,9,3,6, 7,3,5,4	000000111100011001001 1 111010010100110111011 0 010010110101
000111001101011101010 100111010111101111011 10011010111000	70.	2,4,9,6,5, 3,8,7,1	6,5,3,1,2,9,5,3,8,1,4,6,8, 7,1,4,6,5,3,8,1,2,4,9,5,3, 8,2,4,6,3,8,7	011100101010111111111 0 001001000010010000110 0 10011101010
111000001011011001011 111111011100100101011 01100011111001	71.	7,8,6,3,5, 1,9,4,2	7,8,6,2,8,6,5,1,2,8,6,3,5, 1,9,4,2,8,6,3,9,7,6,5,1,4, 2,3,5,1,9,4,8	101111000011000111011 0 000000001111101101111 0 00010100101
011011101111100100010 000100111001000101000 11011000011101	72.	4,3,7,2,9, 1,5,6,8	3,7,9,1,5,8,4,3,7,2,5,3,5, 4,3,7,1,4,7,5,6,4,3,5,6,8, 3	111100010001000011011 0 011101100001011111111 1 01011110010
100011101011110100000 101000111111100100100 00110011111101	73.	6,2,7,3,8, 5,1,9,4	6,8,5,1,4,2,7,3,8,1,3,5,6, 2,7,3,8,5,1,6,3,4,6,3,8,5, 1,9,4,2	111111110100010001011 0 000001100001001011001 0 10011100001
111011011110011100111 101100000011001101000 01001011000101	74.	3,1,4,9,5, 8,7,2,6	3,1,4,5,8,2,6,3,1,5,8,7,3, 1,4,9,8,7,5,8,6,3,4,2,1,9, 5,6,1	000101011000111001101 0 100101001010010011000 0 00101010000
001101001011011110100 000111100000100001000 00111010100111	75.	3,2,7,5,1, 6,8,9,4	7,5,6,4,2,7,1,6,8,9,3,8,9, 4,3,8,7,4,3,2,5,6,4,3,2	011000001000100110011 1 110001100010101000011 0 00111010000
111001101100010101111 001111111010011011110 11001100000000	76.	5,9,1,4,8, 6,7,2,3	5,9,1,6,7,3,5,8,7,3,5,9,1, 6,7,2,3,5,9,1,8,2,3,9,1,4, 8,7,2,9,1	100100101010011001101 1 101100100001000011111 1 00111111001
001010011011000000001	77.	4,5,7,6,9,	7,9,8,1,5,7,7,9,3,2,1,5,7,	010011011001001010110 0

011101011011101100011 11101111111011		3,2,8,1	9,3,2,1,4,9,3,2,8,1,5,7,6, 9,3,2,8,4,5	000110001101010000111 1101101001010
000011111001100000100 11000000111101101010 10000100101010	78.	5,4,6,7,1, 8,2,3,9	1,8,2,3,9,6,7,5,7,1,7,1,8, 2,9,5,6,1,2,6,8,3,5	110111000000001000001 10011110101100101110000 110110000000
001001001011001100111 01111000111010111011 00100010011000	79.	7,3,1,5,2, 4,9,8,6	1,4,6,3,1,4,9,7,3,1,2,4,9, 8,1,5,2,9,6,7,3,5,2,6,5,9, 8	001001010000111101010 000101111101011010010 1101001011101
010111001101110101001 10000000000100011110 1 00110011001111	80.	8,9,1,7,6, 4,2,5,3	9,7,6,4,3,8,1,7,6,2,3,1,7, 4,8,9,1,7,4,3,8,7,6,5,3,8, 9	001100000111001010000 011101110101000001001 01110011110111
000011011100110001100 10100000011010110000 0 10100010001001	81.	1,9,2,7,8, 3,5,4,6	8,3,4,6,1,7,8,6,1,7,3,7,8, 5,6,1,5,6,7,4,9	100110000001101110100 1 1001010010110000010010 010100110110
110000111111110100110 11101011100100111010 0 01001000011100	82.	3,8,7,2,6, 4,5,9,1	3,8,5,9,1,3,8,7,2,6,5,3,8, 2,6,4,9,3,8,7,4,1,3,8,2,9, 8,5,9,1	100001011001111010000 1 000110100110111111001 0 000001100010
100011001011011110001 10011100111011001100 11100011101010	83.	2,9,5,7,8, 1,4,6,3	2,8,1,3,9,5,8,1,4,6,5,7,4, 6,3,5,7,8,4,6,3,5,7,4,6,3, 7,8,1,6,2	111110101011000000001 1 100101000011001000100 100111101101
100011001110010110011 01101010011010011101 1 10010011011110	84.	7,5,6,4,9, 3,1,8,2	7,9,3,2,7,5,9,1,8,5,6,9,3, 8,7,4,9,1,7,5,6,9,3,1,7,4, 9,1,8,2,7	111101011010110101111 0 001110100110110100101 0 011100011100
010110000101100011001 10101100110101001010 0 10110001101010	85.	1,3,2,8,6, 7,9,4,5	3,8,6,1,2,8,4,5,2,8,7,4,5, 2,8,7,4,3,8,9,5,1,6,7,4,1	101101110101110001000 1 001101110111110100110 1 011001101010
000100100101110001001 10011101111000100011 0 10000000011101	86.	7,3,4,5,6, 2,1,9,8	5,1,7,4,5,6,8,4,5,1,9,8,3, 4,5,6,8,5,6,1,1,9,8,3	101011100010111000010 1 000010110000010111110 1 111000010100
111110111100110000001 01111011000000000000 11001111010000	87.	2,8,9,1,6, 5,4,3,7	2,8,9,1,6,4,3,7,2,1,6,9,6, 5,4,3,2,8,4,3,8,9,1,6,4	111011110010011100100 1 000011001111100101011 0 11110001111
000001111000110000011 01110101100010001011 0 11110110001100	88.	2,9,8,6,1, 7,3,4,5	7,3,4,5,6,1,9,8,1,7,3,5,9, 8,3,9,6,1,3,4,5,2,8,6,4,5	101011001000100111100 0 001110001000100110100 1 10100110111
101011011111001010001 00101111111111000101 0 11101010011111	89.	3,6,9,5,4, 7,1,8,2	3,9,4,7,8,2,3,6,9,7,8,9,7, 8,2,3,6,9,5,4,7,1,8,9,4,1, 8,2,6,5,1,8,2,3,6	000010100100100011101 0 111110001110101010010 1 101111001111
001011100011000011011 00110010100000110000 1 10110000110111	90.	5,3,2,8,6, 1,7,4,9	2,6,1,7,3,2,4,9,3,2,1,7,5, 2,9,5,1,7,9,5,1,7,9,5,3	011000110011010101101 0 111000011001101100001 0 001011000000
111000101000111110100 11101111010001100101 0 10100101110001	91.	4,1,6,8,9, 7,2,3,5	4,1,6,2,5,8,9,7,2,3,4,8,9, 7,3,5,4,1,8,3,5,6,9,2,5,6, 9,7,2,1	110100101000111110100 1 000100101000101001001 1 111110110010
011110111011111001101 00101011011010101101 0 11111110111010	92.	4,7,5,3,2, 9,8,6,1	7,5,3,2,8,6,1,7,5,3,2,9,1, 4,5,9,6,4,7,3,2,8,1,7,5,2, 8,6,1,4,7,5,3,9,8,6,4	100010100101001101011 1 100110010101110010110 0 010000010010
011010101001011100101 10101010110100111100 0 10101101110000	93.	6,3,4,9,1, 8,2,5,7	3,4,1,2,7,4,1,8,2,6,4,9,8, 5,6,4,9,8,7,6,3,4,9,2,7,3, 4,1,8,2	001101010000101101010 1 110000101011110100010 0 010000101111
000001100101011001010 01110101000101110100 1 01000001010001	94.	9,5,3,7,1, 4,6,2,8	4,6,9,3,1,4,8,5,1,4,6,8,5, 4,2,8,9,3,4,2,1,6,5	001101101110101011100 1 011011110101110101100 1 000100100010
001001010110000111111 00001111010000000101 0 10001100000101	95.	1,7,8,6,5, 9,2,4,3	8,9,4,1,7,2,4,3,1,7,8,4,3, 1,7,6,6,9,2,7,8,3,7	111000101010110110101 1 101000001000010101001 1 110001101111
010100011100011110000 10101010000101010100 0 11101010111110	96.	8,2,9,5,1, 3,4,6,7	2,5,6,7,8,1,3,4,6,5,3,6,8, 3,6,8,9,4,6,7,2,5,3,4,6,7, 8	110110001010010010000 1 010110101111011000011 1 110100100100

010001000010101100001 010011001010111110000 01110001010000	97.	8,7,6,1,9, 4,5,3,2	7,4,7,1,4,5,6,9,3,2,6,9,5, 3,2,8,7,3,2,8,9,5	101001010011111111011 0 011111000100000110101 0 110111010000
000111110011000011100 111111101010111011110 01101100100110	98.	8,2,6,7,3, 5,4,9,1	7,3,5,4,9,2,6,9,1,8,7,3,5, 4,9,1,8,6,3,4,9,1,2,6,7,3, 9,1,2,6,5,1,8	110101010011100000000 0 011110111001001100010 1 001000000110
101100001011000000000 111100011101101000101 11111011111011	99.	9,4,5,3,2, 1,7,6,8	9,5,3,8,4,5,3,2,1,7,4,5,3, 1,7,8,3,1,7,6,8,9,4,3,2,1, 7,6,9,4	111000111111101101000 0 001101100101101011111 1 011110011000
101011001001101001101 011110001011011100111 10001111000111	100.	4,6,7,3,8, 1,2,5,9	4,7,8,1,9,7,3,1,9,4,7,8,1, 2,5,7,8,1,5,9,4,3,8,1,2,6, 7,3,8,9,4,6	101110010011001010100 1 100000001101111001100 0 000010100011

b) Same packet and same key

In this scenario, in each round the same packet and a same key of 56 bit is used as an input in each round. Simulation results of this scheme are shown in Table 5. Results prove that the produced cipher text is still different in each round. If this scenario is created for a static function, then the produced output is same in each round.

Algorithm 6.4: Same packet same key Algorithm

STEP1: change the packet byte order

STEP2: calculate once complement of the entire packet

STEP3: circular left shift the packet bits

STEP4: Exchange pair digits in packet

STEP5: Cross every byte (First four bits in a byte are exchanged with last four bits in the same byte)

STEP6: Rotate Bits in every byte of the packet

STEP7: XOR current packet bits with original packet bits

STEP8: circular right shift the packet bits

STEP9: Complement bits at only even or only odd positions

Packet used in each round : 00100000101001000010001110011011

Round key in each round : 11100101101110011100001100101111100

Table 6.2: Simulation Results for Same Round Key and Same Packet Input in Each Round

Round No.	**Algorithm sequence for different rounds**	**Steps executed in every round based on the algorithm sequence given in that round**	**CT produced in every round**
1.	9, 8, 7, 6, 5, 4, 3, 2, 1	9, 8, 7, 4, 2, 1, 8, 7, 6, 3, 2, 1, 5, 4, 1, 8, 7, 6, 5, 4	011011010100110001001 00000011110
2.	9, 8, 7, 6, 5, 4, 3, 1, 2	9, 8, 7, 4, 1, 2, 8, 7, 6, 3, 1, 2, 5, 4, 2, 8, 7, 6, 5, 4	011001111110001010111 10101001111
3.	9, 8, 7, 6, 5, 4, 2, 3, 1	9, 8, 7, 4, 3, 1, 8, 7, 6, 2, 3, 1, 5, 4, 1, 8, 7, 6, 5, 4	101101101110111111011 00001101111
4.	9, 8, 7, 6, 5, 4, 2, 1, 3	9, 8, 7, 4, 1, 3, 8, 7, 6, 2, 1, 3, 5, 4, 3, 8, 7, 6, 5, 4	000000011110101001010 01111101101
5.	9, 8, 7, 6, 5, 4, 1, 2, 3	9, 8, 7, 4, 2, 3, 8, 7, 6, 1, 2, 3, 5, 4, 3, 8, 7, 6, 5, 4	111100111101000010000 10000111110
6.	9, 8, 7, 6, 5, 4, 1, 3, 2	9, 8, 7, 4, 3, 2, 8, 7, 6, 1, 3, 2, 5, 4, 2, 8, 7, 6, 5, 4	111001001100010001110 11100010011
7.	9, 8, 7, 6, 5, 3, 4, 2, 1	9, 8, 7, 3, 2, 1, 8, 7, 6, 4, 2, 1, 5, 3, 1, 8, 7, 6, 5, 3	011110000100100000001 10101001010
8.	9, 8, 7, 6, 5, 3, 4, 1, 2	9, 8, 7, 3, 1, 2, 8, 7, 6, 4, 1, 2, 5, 3, 2, 8, 7, 6, 5, 3	100111111010100001110 10100110101
9.	9, 8, 7, 6, 5, 3, 2, 4, 1	9, 8, 7, 3, 4, 1, 8, 7, 6, 2, 4, 1, 5, 3, 1, 8, 7, 6, 5, 3	101001110001001001100 11110010110
10.	9, 8, 7, 6, 5, 3, 2, 1, 4	9, 8, 7, 3, 1, 4, 8, 7, 6, 2, 1, 4, 5, 3, 4, 8, 7, 6, 5, 3	000000110101111101110 10010010011
11.	9, 8, 7, 6, 5, 3, 1, 2, 4	9, 8, 7, 3, 2, 4, 8, 7, 6, 1, 2, 4, 5, 3, 4, 8, 7, 6, 5, 3	111010000001001001100 11110111111
12.	9, 8, 7, 6, 5, 3, 1, 4, 2	9, 8, 7, 3, 4, 2, 8, 7, 6, 1, 4, 2, 5, 3, 2, 8, 7, 6, 5, 3	111010000000001010001 01010111101
13.	9, 8, 7, 6, 5, 2, 3, 4, 1	9, 8, 7, 2, 4, 1, 8, 7, 6, 3, 4, 1, 5, 2, 1, 8, 7, 6, 5, 2	011000010111001101111 01111010010
14.	9, 8, 7, 6, 5, 2, 3, 1, 4	9, 8, 7, 2, 1, 4, 8, 7, 6, 3, 1, 4, 5, 2, 4, 8, 7, 6, 5, 2	111001011100111100000 00010010001
15.	9, 8, 7, 6, 5, 2, 4, 3, 1	9, 8, 7, 2, 3, 1, 8, 7, 6, 4, 3, 1, 5, 2, 1, 8, 7, 6, 5, 2	101100000011110100101 10101111101

16.	9, 8, 7, 6, 5, 2, 4, 1, 3	9, 8, 7, 2, 1, 3, 8, 7, 6, 4, 1, 3, 5, 2, 3, 8, 7, 6, 5, 2	1101111010011111100011001011000
17.	9, 8, 7, 6, 5, 2, 1, 4, 3	9, 8, 7, 2, 4, 3, 8, 7, 6, 1, 4, 3, 5, 2, 3, 8, 7, 6, 5, 2	0101001111000101101000111100101 0
18.	9, 8, 7, 6, 5, 2, 1, 3, 4	9, 8, 7, 2, 3, 4, 8, 7, 6, 1, 3, 4, 5, 2, 4, 8, 7, 6, 5, 2	0010100111100000010010110101011 11
19.	9, 8, 7, 6, 5, 1, 3, 2, 4	9, 8, 7, 1, 2, 4, 8, 7, 6, 3, 2, 4, 5, 1, 4, 8, 7, 6, 5, 1	0111011011111111000011000101100 0
20.	9, 8, 7, 6, 5, 1, 3, 4, 2	9, 8, 7, 1, 4, 2, 8, 7, 6, 3, 4, 2, 5, 1, 2, 8, 7, 6, 5, 1	1111000101111110100010111101100 1
21.	9, 8, 7, 6, 5, 1, 2, 3, 4	9, 8, 7, 1, 3, 4, 8, 7, 6, 2, 3, 4, 5, 1, 4, 8, 7, 6, 5, 1	1101100110000010010010100101010 1
22.	9, 8, 7, 6, 5, 1, 2, 4, 3	9, 8, 7, 1, 4, 3, 8, 7, 6, 2, 4, 3, 5, 1, 3, 8, 7, 6, 5, 1	1101000101101101001000010110001 0
23.	9, 8, 7, 6, 5, 1, 4, 2, 3	9, 8, 7, 1, 2, 3, 8, 7, 6, 4, 2, 3, 5, 1, 3, 8, 7, 6, 5, 1	1010111000001101110110011110001 1
24.	9, 8, 7, 6, 5, 1, 4, 3, 2	9, 8, 7, 1, 3, 2, 8, 7, 6, 4, 3, 2, 5, 1, 2, 8, 7, 6, 5, 1	1100110101010010011101000000010 1
25.	9, 8, 7, 6, 4, 5, 3, 2, 1	9, 8, 7, 5, 2, 1, 8, 7, 6, 3, 2, 1, 4, 5, 1, 8, 7, 6, 4, 5	1011011000010000011100100011101 0
26.	9, 8, 7, 6, 4, 5, 3, 1, 2	9, 8, 7, 5, 1, 2, 8, 7, 6, 3, 1, 2, 4, 5, 2, 8, 7, 6, 4, 5	1100011110110011010001010010000 0
27.	9, 8, 7, 6, 4, 5, 2, 3, 1	9, 8, 7, 5, 3, 1, 8, 7, 6, 2, 3, 1, 4, 5, 1, 8, 7, 6, 4, 5	0100101000101110011010110010110 1
28.	9, 8, 7, 6, 4, 5, 2, 1, 3	9, 8, 7, 5, 1, 3, 8, 7, 6, 2, 1, 3, 4, 5, 3, 8, 7, 6, 4, 5	1100001101011110101001011111010 0
29.	9, 8, 7, 6, 4, 5, 1, 2, 3	9, 8, 7, 5, 2, 3, 8, 7, 6, 1, 2, 3, 4, 5, 3, 8, 7, 6, 4, 5	0001010110001011010011110010111 0
30.	9, 8, 7, 6, 4, 5, 1, 3, 2	9, 8, 7, 5, 3, 2, 8, 7, 6, 1, 3, 2, 4, 5, 2, 8, 7, 6, 4, 5	1011011101101011001110001001001 1
31.	9, 8, 7, 6, 4, 3, 5, 2, 1	9, 8, 7, 3, 2, 1, 8, 7, 6, 5, 2, 1, 4, 3, 1, 8, 7, 6, 4, 3	1000011100000011001000010100101 1
32.	9, 8, 7, 6, 4, 3, 5, 1, 2	9, 8, 7, 3, 1, 2, 8, 7, 6, 5, 1, 2, 4, 3, 2, 8, 7, 6, 4, 3	1111100100011100101001101010110 1
33.	9, 8, 7, 6, 4, 3, 2, 5, 1	9, 8, 7, 3, 5, 1, 8, 7, 6, 2, 5, 1, 4, 3, 1, 8, 7, 6, 4, 3	1100001111101111110011110000110 1

34.	9, 8, 7, 6, 4, 3, 2, 1, 5	9, 8, 7, 3, 1, 5, 8, 7, 6, 2, 1, 5, 4, 3, 5, 8, 7, 6, 4, 3	000100100110111001101 01000000111
35.	9, 8, 7, 6, 4, 3, 1, 2, 5	9, 8, 7, 3, 2, 5, 8, 7, 6, 1, 2, 5, 4, 3, 5, 8, 7, 6, 4, 3	001000001001101010111 00010001010
36.	9, 8, 7, 6, 4, 3, 1, 5, 2	9, 8, 7, 3, 5, 2, 8, 7, 6, 1, 5, 2, 4, 3, 2, 8, 7, 6, 4, 3	101010010110000010111 10000000010
37.	9, 8, 7, 6, 4, 2, 3, 5, 1	9, 8, 7, 2, 5, 1, 8, 7, 6, 3, 5, 1, 4, 2, 1, 8, 7, 6, 4, 2	100101001111111011011 00001011100
38.	9, 8, 7, 6, 4, 2, 3, 1, 5	9, 8, 7, 2, 1, 5, 8, 7, 6, 3, 1, 5, 4, 2, 5, 8, 7, 6, 4, 2	010011110000100000000 01110101010
39.	9, 8, 7, 6, 4, 2, 5, 3, 1	9, 8, 7, 2, 3, 1, 8, 7, 6, 5, 3, 1, 4, 2, 1, 8, 7, 6, 4, 2	011010010110111000101 01000001111
40.	9, 8, 7, 6, 4, 2, 5, 1, 3	9, 8, 7, 2, 1, 3, 8, 7, 6, 5, 1, 3, 4, 2, 3, 8, 7, 6, 4, 2	011010010010000111011 01001011110
41.	9, 8, 7, 6, 4, 2, 1, 5, 3	9, 8, 7, 2, 5, 3, 8, 7, 6, 1, 5, 3, 4, 2, 3, 8, 7, 6, 4, 2	001001101100111100000 01100010101
42.	9, 8, 7, 6, 4, 2, 1, 3, 5	9, 8, 7, 2, 3, 5, 8, 7, 6, 1, 3, 5, 4, 2, 5, 8, 7, 6, 4, 2	000110011111000001111 01000011011
43.	9, 8, 7, 6, 4, 1, 3, 2, 5	9, 8, 7, 1, 2, 5, 8, 7, 6, 3, 2, 5, 4, 1, 5, 8, 7, 6, 4, 1	101010100011111111101 11100001101
44.	9, 8, 7, 6, 4, 1, 3, 5, 2	9, 8, 7, 1, 5, 2, 8, 7, 6, 3, 5, 2, 4, 1, 2, 8, 7, 6, 4, 1	010000000010101011011 10000111110
45.	9, 8, 7, 6, 4, 1, 2, 3, 5	9, 8, 7, 1, 3, 5, 8, 7, 6, 2, 3, 5, 4, 1, 5, 8, 7, 6, 4, 1	010011111100110101100 11111110100
46.	9, 8, 7, 6, 4, 1, 2, 5, 3	9, 8, 7, 1, 5, 3, 8, 7, 6, 2, 5, 3, 4, 1, 3, 8, 7, 6, 4, 1	000001101100010100100 01100011111
47.	9, 8, 7, 6, 4, 1, 5, 2, 3	9, 8, 7, 1, 2, 3, 8, 7, 6, 5, 2, 3, 4, 1, 3, 8, 7, 6, 4, 1	100111111111111000010 00010001110
48.	9, 8, 7, 6, 4, 1, 5, 3, 2	9, 8, 7, 1, 3, 2, 8, 7, 6, 5, 3, 2, 4, 1, 2, 8, 7, 6, 4, 1	001011010111001101100 11011010110
49.	9, 8, 7, 6, 3, 4, 5, 2, 1	9, 8, 7, 4, 2, 1, 8, 7, 6, 5, 2, 1, 3, 4, 1, 8, 7, 6, 3, 4	100110100001101011011 11111010111
50.	9, 8, 7, 6, 3, 4, 5, 1, 2	9, 8, 7, 4, 1, 2, 8, 7, 6, 5, 1, 2, 3, 4, 2, 8, 7, 6, 3, 4	101010101110101000100 11000010001
51.	9, 8, 7, 6, 3, 4, 2, 5, 1	9, 8, 7, 4, 5, 1, 8, 7, 6, 2, 5, 1, 3, 4, 1, 8, 7, 6, 3, 4	010101001111010010111 01110010011

52.	9, 8, 7, 6, 3, 4, 2, 1, 5	9, 8, 7, 4, 1, 5, 8, 7, 6, 2, 1, 5, 3, 4, 5, 8, 7, 6, 3, 4	000000101111100011001 00110111011
53.	9, 8, 7, 6, 3, 4, 1, 2, 5	9, 8, 7, 4, 2, 5, 8, 7, 6, 1, 2, 5, 3, 4, 5, 8, 7, 6, 3, 4	010110011000010101100 00001010010
54.	9, 8, 7, 6, 3, 4, 1, 5, 2	9, 8, 7, 4, 5, 2, 8, 7, 6, 1, 5, 2, 3, 4, 2, 8, 7, 6, 3, 4	000111110111101111001 01110111110
55.	9, 8, 7, 6, 3, 5, 4, 2, 1	9, 8, 7, 5, 2, 1, 8, 7, 6, 4, 2, 1, 3, 5, 1, 8, 7, 6, 3, 5	101001110101000000100 11100111110
56.	9, 8, 7, 6, 3, 5, 4, 1, 2	9, 8, 7, 5, 1, 2, 8, 7, 6, 4, 1, 2, 3, 5, 2, 8, 7, 6, 3, 5	101001010101010000100 10100111010
57.	9, 8, 7, 6, 3, 5, 2, 4, 1	9, 8, 7, 5, 4, 1, 8, 7, 6, 2, 4, 1, 3, 5, 1, 8, 7, 6, 3, 5	010011100010111001101 11100101101
58.	9, 8, 7, 6, 3, 5, 2, 1, 4	9, 8, 7, 5, 1, 4, 8, 7, 6, 2, 1, 4, 3, 5, 4, 8, 7, 6, 3, 5	111110011111100100100 01000000000
59.	9, 8, 7, 6, 3, 5, 1, 2, 4	9, 8, 7, 5, 2, 4, 8, 7, 6, 1, 2, 4, 3, 5, 4, 8, 7, 6, 3, 5	000011001001010111010 00011110010
60.	9, 8, 7, 6, 3, 5, 1, 4, 2	9, 8, 7, 5, 4, 2, 8, 7, 6, 1, 4, 2, 3, 5, 2, 8, 7, 6, 3, 5	001100110110100111100 01100000010
61.	9, 8, 7, 6, 3, 2, 5, 4, 1	9, 8, 7, 2, 4, 1, 8, 7, 6, 5, 4, 1, 3, 2, 1, 8, 7, 6, 3, 2	011011000010101101101 01100001110
62.	9, 8, 7, 6, 3, 2, 5, 1, 4	9, 8, 7, 2, 1, 4, 8, 7, 6, 5, 1, 4, 3, 2, 4, 8, 7, 6, 3, 2	111010001111100100010 00000100011
63.	9, 8, 7, 6, 3, 2, 4, 5, 1	9, 8, 7, 2, 5, 1, 8, 7, 6, 4, 5, 1, 3, 2, 1, 8, 7, 6, 3, 2	100001011111111010001 00100011100
64.	9, 8, 7, 6, 3, 2, 4, 1, 5	9, 8, 7, 2, 1, 5, 8, 7, 6, 4, 1, 5, 3, 2, 5, 8, 7, 6, 3, 2	001111001000101000100 10010000011
65.	9, 8, 7, 6, 3, 2, 1, 4, 5	9, 8, 7, 2, 4, 5, 8, 7, 6, 1, 4, 5, 3, 2, 5, 8, 7, 6, 3, 2	101010111011011011110 01010101011
66.	9, 8, 7, 6, 3, 2, 1, 5, 4	9, 8, 7, 2, 5, 4, 8, 7, 6, 1, 5, 4, 3, 2, 4, 8, 7, 6, 3, 2	001011111011011011100 01111000001
67.	9, 8, 7, 6, 3, 1, 5, 2, 4	9, 8, 7, 1, 2, 4, 8, 7, 6, 5, 2, 4, 3, 1, 4, 8, 7, 6, 3, 1	011100000010100001011 11010001110
68.	9, 8, 7, 6, 3, 1, 5, 4, 2	9, 8, 7, 1, 4, 2, 8, 7, 6, 5, 4, 2, 3, 1, 2, 8, 7, 6, 3, 1	110011000001100000100 00110000010
69.	9, 8, 7, 6, 3, 1, 2, 5, 4	9, 8, 7, 1, 5, 4, 8, 7, 6, 2, 5, 4, 3, 1, 4, 8, 7, 6, 3, 1	001111001100011010111 00011001010

70.	9, 8, 7, 6, 3, 1, 2, 4, 5	9, 8, 7, 1, 4, 5, 8, 7, 6, 2, 4, 5, 3, 1, 5, 8, 7, 6, 3, 1	0110011011101011000011110000 1010
71.	9, 8, 7, 6, 3, 1, 4, 2, 5	9, 8, 7, 1, 2, 5, 8, 7, 6, 4, 2, 5, 3, 1, 5, 8, 7, 6, 3, 1	1101001100000010001011100100 0100
72.	9, 8, 7, 6, 3, 1, 4, 5, 2	9, 8, 7, 1, 5, 2, 8, 7, 6, 4, 5, 2, 3, 1, 2, 8, 7, 6, 3, 1	1001111110010111011011100010 0010
73.	9, 8, 7, 6, 2, 4, 3, 5, 1	9, 8, 7, 4, 5, 1, 8, 7, 6, 3, 5, 1, 2, 4, 1, 8, 7, 6, 2, 4	0100111101001000010010000000 111000
74.	9, 8, 7, 6, 2, 4, 3, 1, 5	9, 8, 7, 4, 1, 5, 8, 7, 6, 3, 1, 5, 2, 4, 5, 8, 7, 6, 2, 4	1101010001001000010010000010 11100
75.	9, 8, 7, 6, 2, 4, 5, 3, 1	9, 8, 7, 4, 3, 1, 8, 7, 6, 5, 3, 1, 2, 4, 1, 8, 7, 6, 2, 4	1001010000000101001100100100 1101
76.	9, 8, 7, 6, 2, 4, 5, 1, 3	9, 8, 7, 4, 1, 3, 8, 7, 6, 5, 1, 3, 2, 4, 3, 8, 7, 6, 2, 4	0110111101011001010000100010 0001
77.	9, 8, 7, 6, 2, 4, 1, 5, 3	9, 8, 7, 4, 5, 3, 8, 7, 6, 1, 5, 3, 2, 4, 3, 8, 7, 6, 2, 4	0101100110101111111111011100 10100
78.	9, 8, 7, 6, 2, 4, 1, 3, 5	9, 8, 7, 4, 3, 5, 8, 7, 6, 1, 3, 5, 2, 4, 5, 8, 7, 6, 2, 4	1110010011010111000100111001 1011
79.	9, 8, 7, 6, 2, 3, 4, 5, 1	9, 8, 7, 3, 5, 1, 8, 7, 6, 4, 5, 1, 2, 3, 1, 8, 7, 6, 2, 3	0101101000001000010010010110 1100
80.	9, 8, 7, 6, 2, 3, 4, 1, 5	9, 8, 7, 3, 1, 5, 8, 7, 6, 4, 1, 5, 2, 3, 5, 8, 7, 6, 2, 3	0111001110001001100010111111 1001
81.	9, 8, 7, 6, 2, 3, 5, 4, 1	9, 8, 7, 3, 4, 1, 8, 7, 6, 5, 4, 1, 2, 3, 1, 8, 7, 6, 2, 3	1001010011001111100110001000 0111
82.	9, 8, 7, 6, 2, 3, 5, 1, 4	9, 8, 7, 3, 1, 4, 8, 7, 6, 5, 1, 4, 2, 3, 4, 8, 7, 6, 2, 3	1100111000001010101010001101 0111
83.	9, 8, 7, 6, 2, 3, 1, 5, 4	9, 8, 7, 3, 5, 4, 8, 7, 6, 1, 5, 4, 2, 3, 4, 8, 7, 6, 2, 3	0100001110101001111011100010 0110
84.	9, 8, 7, 6, 2, 3, 1, 4, 5	9, 8, 7, 3, 4, 5, 8, 7, 6, 1, 4, 5, 2, 3, 5, 8, 7, 6, 2, 3	0101001101110110000000010000 0110
85.	9, 8, 7, 6, 2, 5, 3, 4, 1	9, 8, 7, 5, 4, 1, 8, 7, 6, 3, 4, 1, 2, 5, 1, 8, 7, 6, 2, 5	1111100001001000010010000100 0011
86.	9, 8, 7, 6, 2, 5, 3, 1, 4	9, 8, 7, 5, 1, 4, 8, 7, 6, 3, 1, 4, 2, 5, 4, 8, 7, 6, 2, 5	1010001100000011111011001111 0111
87.	9, 8, 7, 6, 2, 5, 4, 3, 1	9, 8, 7, 5, 3, 1, 8, 7, 6, 4, 3, 1, 2, 5, 1, 8, 7, 6, 2, 5	1111110011010011110010110011 1001

88.	9, 8, 7, 6, 2, 5, 4, 1, 3	9, 8, 7, 5, 1, 3, 8, 7, 6, 4, 1, 3, 2, 5, 3, 8, 7, 6, 2, 5	0001101001010011111000101011111110
89.	9, 8, 7, 6, 2, 5, 1, 4, 3	9, 8, 7, 5, 4, 3, 8, 7, 6, 1, 4, 3, 2, 5, 3, 8, 7, 6, 2, 5	1100101011111111010010000001011011
90.	9, 8, 7, 6, 2, 5, 1, 3, 4	9, 8, 7, 5, 3, 4, 8, 7, 6, 1, 3, 4, 2, 5, 4, 8, 7, 6, 2, 5	0110010100001110101011010100011011
91.	9, 8, 7, 6, 2, 1, 3, 5, 4	9, 8, 7, 1, 5, 4, 8, 7, 6, 3, 5, 4, 2, 1, 4, 8, 7, 6, 2, 1	1111111001110011000011000101011100
92.	9, 8, 7, 6, 2, 1, 3, 4, 5	9, 8, 7, 1, 4, 5, 8, 7, 6, 3, 4, 5, 2, 1, 5, 8, 7, 6, 2, 1	0011010100100001001000010010111
93.	9, 8, 7, 6, 2, 1, 5, 3, 4	9, 8, 7, 1, 3, 4, 8, 7, 6, 5, 3, 4, 2, 1, 4, 8, 7, 6, 2, 1	0110001011010011000110111101110
94.	9, 8, 7, 6, 2, 1, 5, 4, 3	9, 8, 7, 1, 4, 3, 8, 7, 6, 5, 4, 3, 2, 1, 3, 8, 7, 6, 2, 1	1110001000101001011011010101011001
95.	9, 8, 7, 6, 2, 1, 4, 5, 3	9, 8, 7, 1, 5, 3, 8, 7, 6, 4, 5, 3, 2, 1, 3, 8, 7, 6, 2, 1	0000010001110010101001100101001001
96.	9, 8, 7, 6, 2, 1, 4, 3, 5	9, 8, 7, 1, 3, 5, 8, 7, 6, 4, 3, 5, 2, 1, 5, 8, 7, 6, 2, 1	0010001100101101111101001011101011
97.	9, 8, 7, 6, 1, 4, 3, 2, 5	9, 8, 7, 4, 2, 5, 8, 7, 6, 3, 2, 5, 1, 4, 5, 8, 7, 6, 1, 4	1111111110010111000110001010101101
98.	9, 8, 7, 6, 1, 4, 3, 5, 2	9, 8, 7, 4, 5, 2, 8, 7, 6, 3, 5, 2, 1, 4, 2, 8, 7, 6, 1, 4	0101000111100100111001110110111000
99.	9, 8, 7, 6, 1, 4, 2, 3, 5	9, 8, 7, 4, 3, 5, 8, 7, 6, 2, 3, 5, 1, 4, 5, 8, 7, 6, 1, 4	0111011100011010010010110110010
100.	9, 8, 7, 6, 1, 4, 2, 5, 3	9, 8, 7, 4, 5, 3, 8, 7, 6, 2, 5, 3, 1, 4, 3, 8, 7, 6, 1, 4	1000101110011111000011000100101101

6.6. ANALYTICAL RESULTS AND PERFORMANCE

Let n be the number of instructions used in the algorithm and k is the key size. If algorithm sequence is not changed then the power of the system exists in the key only and the complexity of static function becomes ns^k where ns is the number of bits of key. On the other hand, if algorithm sequence used in the algorithm is changed randomly and the system works without any key then the complexity of the system becomes $n!$. In presented system, with both the parameters (key and execution sequence) participate, then the

complexity of the system becomes $n!.j^k$.

Theoretical analysis of the presented model has been shown graphically considering n=9 and ns=2 (binary) and number of key bits are changed from k=1 to 11. Figure 6.13 shows the time to compromise a sensor node. In figure the time complexity between static encryption function (SEF) v/s presented dynamic encryption function (DEF) is shown. In case of SEF, the time complexity for key size (k=11) is 2048 and for DEF it is 743178240 which is very high for such a small key size.

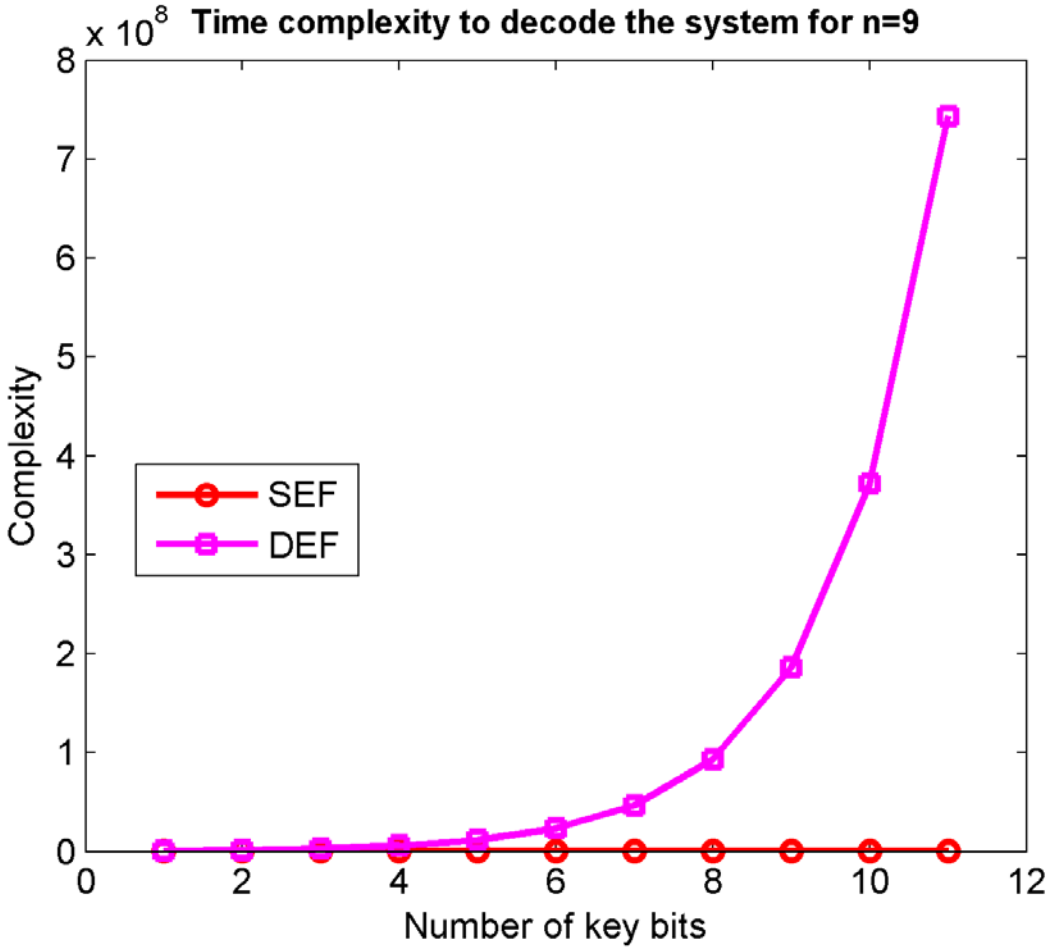

Figure 6.13: Time Complexity to Decode the System

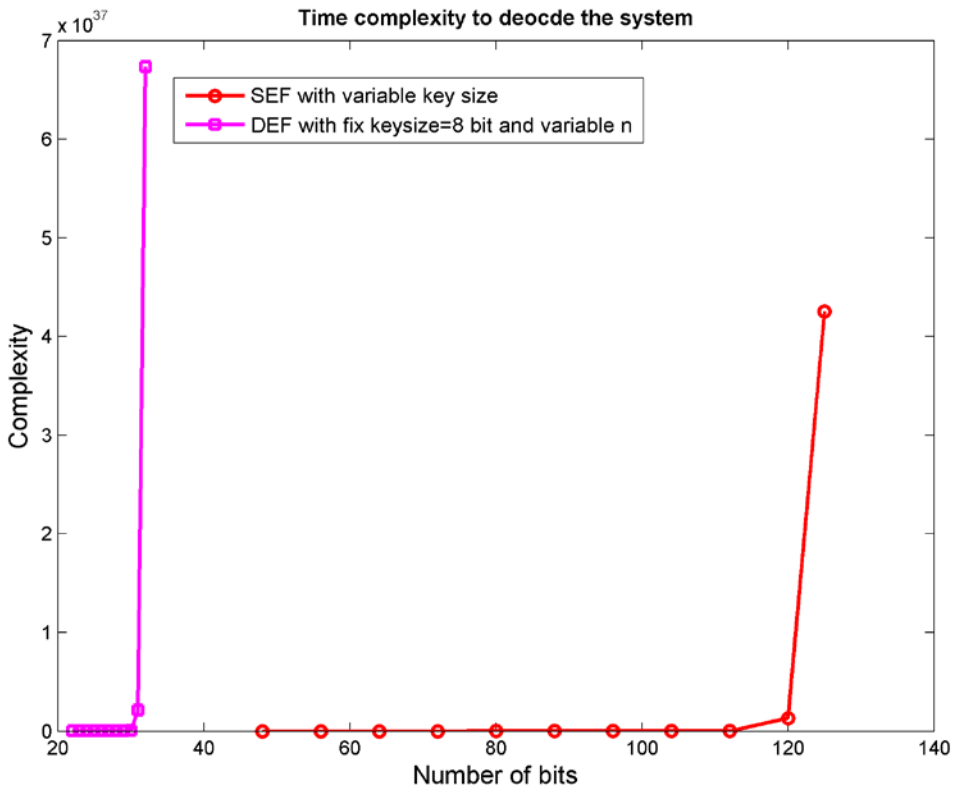

Figure 6.14: Dynamic Encryption Function with 8 Bit Key v/s Static Encryption Function with 128 Bits Key

Figure 6.14 shows the time complexity between static v/s dynamic encryption function. Figure shows the time complexity difference between DEF and SEF with fixed key size (k) and variable number of instructions (n) with variable key size. The number of key bits in case of DEF is fixed to 8 and number of instructions are changed from n=22 to 32. The size of key is increased from 48 bit to 125 bits in case of SEF. Still the performance of the presented system observed better.

The performance of the system is evaluated as follows:

i. Lightweight Encryption algorithm: the algorithm used in this security framework is lightweight in nature. The algorithm uses instructions performing either logical operations or the instructions that changes the sequence of packet bits.

ii. Forward secrecy: The power of this system exists not only in the key but in the dynamic function also. The instructions of dynamic function shuffled with the help of a key in each round. The key is applied on sequence generator (SG) to shuffle the instruction sequence of previous round and finally to produce the instruction sequence of current round. If some hacker or intruder generates the instruction sequence (IS) of current round, it is completely impossible to generate the instruction sequence of next

round without hacking the key of next round. Similarly if some intruder hack the key of current round, but not the execution IS of previous round then it is still impossible to generate the IS of current round. Therefore, in presented security framework, the principle of forward security is easily and efficiently achieved.

iii. Backward secrecy: The principle of backward secrecy is automatically achieved if the principle of forward secrecy is achieved as the same technique is used which works in case of achieving forward secrecy. Even if a new sensor is replaced with the old sensor, backward secrecy is still maintained for the new joining sensor as this sensor never generates any past key due to the fact that new sensor has not a round key of past rounds.

Even if all the confidential parameters of current round are compromised, it is still completely impossible to break the principle of forward and backward secrecy because a new key is used in next round that stops hacking the parameters of next round.

6.7. SUMMARY

This chapter explores the techniques for achieving Forward and Backward secrecy in WSNs. This chapter presents DESFB, a symmetric key-based dynamic encryption scheme to achieve both forward and backward secrecy in WSNs. This scheme is very suitable against chaining attacks resulting from compromised sensor nodes in the network. Simulation results show that the presented scheme performs better in terms of forward and backward secrecy and is also very strong to decode the system in comparison to static encryption function. In chapter-7, Pairing Based Encoding Scheme (PBES) has been presented in detail.