



ANCHOR: A New Proposed Stream Cipher for Smart Cards with Crypto Co-Processor

Ahmed O. Elrefai^{1,*}, Khaled A. Shehata¹, Hazem M. Eldeeb¹, Hanady H. Issa²

¹ Department of Electronics and Communication Engineer, Faculty of Engineer, Arab Academy for Science Technology University, Cairo, Egypt

² Department of Electronics and Communication Engineer, Faculty of Engineer, Military Technical College, Cairo, Egypt

ARTICLE INFO

ABSTRACT

Article history:

Received 20 October 2023

Received in revised form 20 December 2023

Accepted 10 February 2024

Available online 22 May 2024

Keywords:

Smart card; Encryption algorithm;
Crypto co-processor; ANCHOR; Stream
cipher; Cryptography

A smart card is a small pocket-sized computer with limited resources used for secure data processing and storage. The card consists of different software and hardware components, including a microprocessor, crypto co-processor, RAM, secure ROM, and operating system. Even though smart cards have a lot of limitations in terms of processing power and small-sized memory, smart cards are widely used in many applications that require a high degree of security such as e-passports, citizen cards, e-banking, etc... Basically, the security of a smart card depends mainly on the security strength of the cipher algorithm implemented inside. This paper presents a new lightweight, high-speed, and cryptographically strong stream cipher algorithm (ANCHOR) suitable for implementation on smart cards. The building blocks of the proposed cipher were carefully built/chosen with high-valued cryptographic properties. The randomness and linear complexity properties of the proposed cipher algorithm have been successfully tested with statistical tests of the NIST suite and the Berlekamp- Massey algorithm respectively. In order to test the performance of the proposed cipher algorithm, the algorithm was developed in C language and executed on a Linux machine.

1. Introduction

Security experts around the world work relentlessly to thwart the ever-growing threats of malicious attacks over computing environments and data transmission. One of the most common methods of data protection is the process of data encryption, which is transforming the data and encoding it in a way that the data becomes unintelligible not only to the naked eye but also to expert hackers as well. Encrypted data can only be read or used when restored to its original form, or when it is decrypted.

* Corresponding author.

E-mail address: ahmadelrefaai@gmail.com

<https://doi.org/10.37934/araset.45.2.227239>

1.1 Smart Card

The concept of the smart card came into being in the early 1970s through research and development in some of the most technologically innovative nations across the world, including Germany, France, and Japan [A1]. A smart Card is a plastic card, which contains an embedded computer, that can store data and a computing system that transacts data via a reader. Smart card readers require the lowest cost of system maintenance as it has a high lifetime rate.

A smart card is considered a small, tiny computer that consists of a contact interface, microprocessor chip, EPROM, ROM, memory chip, cipher chip (crypto co-processor), antenna for a contactless interface, separate magnetic strip, and printing. All these components together in a card provide many efficient capabilities from internal program memory to additional programmable memory and coprocessor security. Nowadays, there are some components that may be added, such as LCD, Fingerprint sensor, integrated tiny battery, and simple pin pad.

1.2 Types of Smart Cards

Smart Cards have three main types depending on the application and its supported communication systems, commonly there are three main types. The first type is Contact Interface Cards, it is the most common type that is, not only supported with ISO7816 T0 or T1 protocols but also supported with electrical contacts for establishing communication and data transactions.

The second type is Contactless Interface Cards, it is the second main type, which is supported with ISO14443 protocol; also, it is supported with RFID to establish communication and data transaction. As the Contact or Contactless Interface cards cannot influence the development of some applications, a Dual Interface card had been established, which allows vendors to use both features of Contact and Contactless Interface cards to fit the market needs. Hence, it could be one of the main types of smart cards. The third main type is Multi-Component Cards that are supported with ISO7816 USB protocol and also supported with USB interface and fingerprint scanners for establishing communication and data transactions.

1.3 Smart Card Communications

Smart cards typically use a serial interface for communication with the card reader. The most common interface standard is the ISO/IEC 7816 standard, which defines the transmission protocols and electrical characteristics for smart cards. Smart cards communicate with the external device using Application Protocol Data Units (APDU) commands. These commands follow a specific structure and are used for various operations, such as data retrieval, storage, authentication, and cryptographic operations.

1.4 Smart Card Security

Smart cards typically Despite the increased, widely spread applications of Smart Cards within various fields due to their reliability and security of data being stored or processed inside, smart cards are subjected to various types of attacks that aim to gain unauthorized access to the data being stored or processed. One of these attacks is side-channel attacks, in which a hacker takes advantage of flaws in the execution of cryptographic algorithms to extract secret keys or other sensitive data.

The security of smart cards depends mainly on the security of the adopted cipher algorithm. A crypto coprocessor on a smart card is normally utilized with proprietary encryption algorithms to

increase the degree of security in which all the cryptographic operations (Encryption and Decryption) are executed inside the card using the dedicated microprocessor without exposing the cipher keys to the outer world. Crypto coprocessors are frequently implemented as independent hardware modules, each with its own memory and processing capabilities, and they are designed specifically for a given cryptographic function.

The ANCHOR algorithm is proposed as a novel stream cipher algorithm designed for data encryption in smart cards. The ANCHOR algorithm is an alternative to commonly used ciphers such as (AES – DES) and modified ciphers could be used such as Hybrid AES-Modified ECC Algorithm [21] in the crypto processor of smart cards. Compared to the existing ciphers, the ANCHOR stream cipher has multiple advantages, for example, its operation speed and small memory usage

1.5 Paper Structure

In this paper, we demonstrate our work in six sections as follows. Section One introduces smart card applications, limitations, and development. Section two illustrates the literature review as the paper demonstrates a survey of proposed encryption algorithms used in smart cards and illustrates their limitations and weaknesses comparing them with the proposed ANCHOR encryption algorithm. Section three description of the main modules for constructing ANCHOR cipher, in this section we will illustrate the general structure, Key schedule, LFSRs modules, nonlinear module, reduction, and Keystream generator initialization and operation. Section four demonstrates cryptographic assessments regarding security measures and statistical tests such as frequency tests, Runs tests, and Binary Derivative tests. Section five introduces the Berlekamp-Massey to assess the generated keystream properties such as linear complexity and describes ANCHOR cipher performance assessments such as time consumption, and memory usage. Section six concludes the work achieved in the paper and elaborates on future research.

2. Literature Review

Security experts around Smart Cards have a lot of limitations such as limited processing and memory capabilities, consequently, smart cards are susceptible to assaults. Commonly used cyphers such as DES and AES which offer significant security due to their robust architecture and complex math. Integrating AES into data systems enhances privacy and security, boosting confidence in digital interactions. Although AES has limitations like block size restrictions, and DES is considered outdated, various cipher algorithms have been developed for smart cards [22]. Hence, many Cipher algorithms of different types were developed to operate on smart cards.

In this section a survey was conducted for most of the developed cipher algorithms, along with their advantages and disadvantages. Several lightweight encryption algorithms, such as TEA and SDES, as well as SNOW algorithms have been proposed to address the need for fast and small encryption algorithms. However, these algorithms also have their limitations. For instance, TEA and SDES have relatively small key sizes, making them susceptible to brute-force attacks. On the other hand, Snow has a large memory footprint, making it unsuitable for resource-constrained devices.

In recent years, several new algorithms, such as Acorn and Future, have been proposed to address the limitations of existing algorithms. Acorn has a larger memory footprint compared to Snow but provides a higher level of security. Future has a small memory footprint and is fast, making it suitable for deployment in resource-constrained devices. However, the memory usage of the Future algorithm is 4 KB.

Kishan Chand [1] introduces a novel lightweight block cipher named FUTURE. The paper emphasizes the significance of lightweight block ciphers in applications that necessitate efficient data encryption on

low-resource devices. However, the FUTURE cipher has certain limitations. It can only encrypt data in fixed blocks of 64 bits, which may not be suitable for a wide range of applications. Additionally, the FUTURE cipher consists of 10 rounds, each comprising four distinct transformations, which can impact the speed of smart card devices. Moreover, the FUTURE algorithm utilizes 4 KB of memory, which should be taken into consideration in resource-constrained environments.

Heru Nurwarsito and Sarah Kusuma Ayu [2] conduct a comparative analysis of the ACORN algorithm and the SNOW algorithm on a smart card platform. The study involves test vector testing as well as encryption and decryption performance testing to ensure the proper functioning of the system. However, certain limitations were identified. The Acorn algorithm key space of 80 bits is relatively small, making it susceptible to exhaustive key search attacks. On the other hand, the main limitation of the SNOW family of ciphers lies in their relatively low speed compared to some other lightweight encryption algorithms. This reduced speed is attributed to the extensive use of XOR and permutation operations, which can negatively impact the cipher's overall performance.

B. Ege, E. B. Kavun, and T. Yalçın [9] focused on smart cards equipped with non-volatile memory. They implemented the AES (Advanced Encryption Standard) and PRESENT algorithms using Counter (CTR) and XOR-Encrypt-XOR (XEX) modes. The objective was to enable memory encryption on smart cards with limited resources, specifically those implemented on less than 6K gates. However, one limitation of their approach was the utilization of the PRESENT algorithm for memory encryption. In this survey, most of the presented algorithm was Block cipher which operates in fixed-size blocks and doesn't require a mechanism of synchronization.

3. The Proposed Stream Cipher (ANCHOR)

The proposed stream cipher ANCHOR is a new lightweight stream cipher suitable for implementation on smart cards. ANCHOR aims to provide a high degree of data security and overcome the problem of limited resources of smart cards as well. Unlike most of the above- presented cipher algorithms for smart cards which adopted the block cipher approach to avoid the synchronization process, ANCHOR adopted the approach of stream cipher with an off-line mechanism for synchronization using stream cipher algorithms on smart cards ensuring high-speed processing for data encryption and avoids using a padding mechanism for fixed-block data encryption. Proposed Stream Cipher (ANCHOR) Block Diagram in Figure 1.

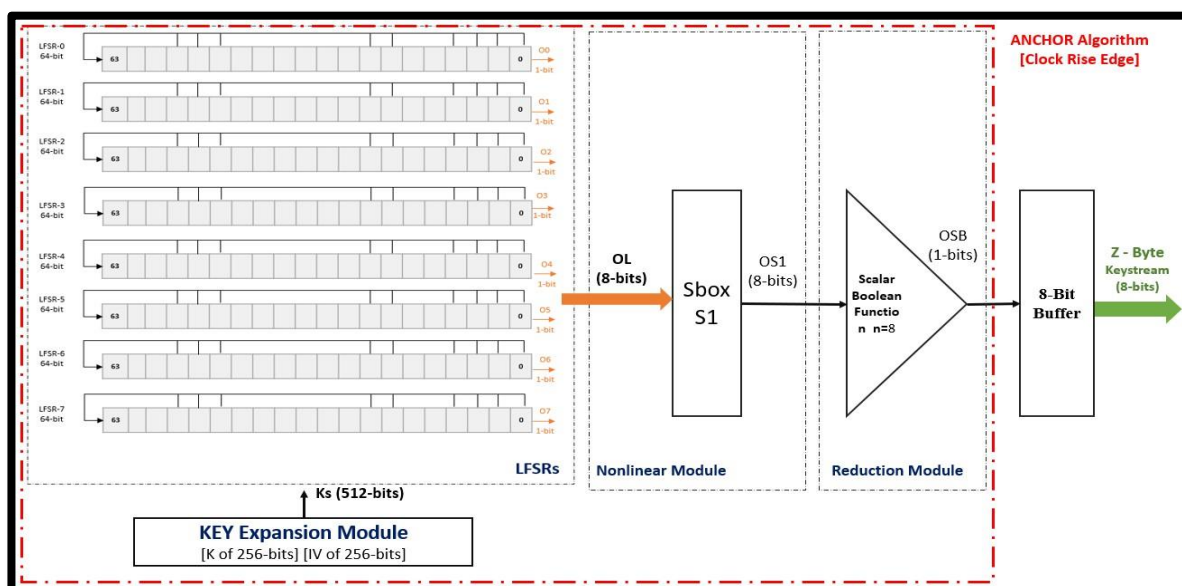


Fig. 1. Proposed Stream Cipher (ANCHOR) Block Diagram

3.1 General Structure

The general structure of the ANCHOR algorithm consists of four building blocks, a set of eight Linear Feedback Shift Registers (LFSRs) each of 64-bit length, a Non-linear module which is the main source of non-linearity, a reduction module, and a key expansion module. The four building blocks are cascaded and integrated together to provide a cryptographically strong cipher algorithm.

The ANCHOR algorithm is used to produce a pseudo-random sequence of bits as a keystream generator. The algorithm requires a 256-bit Initialization Vector (IV) and 256-bit Secret Key (K). A different Initialization Vector is concatenated to each received data frame/ message. Data is ciphered by generating a key stream which is XORed with a byte stream of the plaintext/ciphertext data (see Figure 1).

3.2 Key Expansion Module

The ANCHOR encryption algorithm requires a 512-bit to fill the cells of an eight-bit set of Linear Feedback Shift Registers (Initial State) in order to prepare the cipher algorithm for working. The function of the key expansion module is to set a value to the Initial State. The key expansion module uses the same Sbox (S1) of the ANCHOR algorithm as shown in Figure 2.

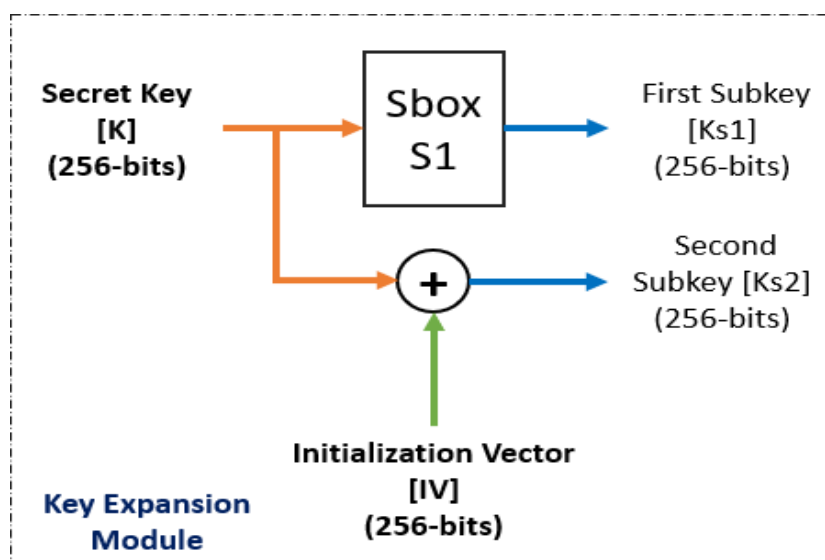


Fig. 2. ANCHOR Key Expansion Module

The secret key (K) of 256-bits length is passed as an input to Sbox (S1) to produce the first half of the Initial State (Ks1). The second half of the Initial State (Ks2) is produced by XORing the Secret key with the received Initialization Vector. In order to fill the eight shift registers, each of Ks1, Ks2 is split into four parts in Eq. (1) and Eq. (2) as follows

$$Ks1 = Kr0 \parallel Kr1 \parallel Kr2 \parallel Kr3 \tag{1}$$

$$Ks2 = Kr4 \parallel Kr5 \parallel Kr6 \parallel Kr7 \tag{2}$$

The values (Kr0, Kr1,....., Kr7) fill the eight shift registers to create the Initial State value. The key expansion process of the ANCHOR cipher is designed to be both straightforward and efficient, while still maintaining a high level of security.

3.3 LFSRs Module

One of the most important characteristics of any strong cipher algorithm is the long cycle. This module is designed to provide the cipher algorithm with a very large cycle as it consists of eight bitwise LFSRs, each of 64-bit length, and equipped with a primitive polynomial that guarantees maximal length.

Figure 1 shows the eight different LFSRs (LFSR-0, LFSR-1, LFSR-2, LFSR-3, LFSR-4, LFSR-5, LFSR-6, and LFSR-7). The internal state of each LFSRs is completely changed with each rising edge of each clock cycle due to the shift operation by the value (V). Furthermore, the feedback polynomials of the LFSRs are selected to be primitive polynomials with dense coefficients. Each LFSR produces a binary sequence period with maximal length sequences in Eq. (3) as follows

$$P = 2^{(n\#)} - 1 \tag{3}$$

The coefficients of the eight primitive polynomials are shown in the following Table 1:

Table 1

LFSRs Primitive Polynomial Tabs

| LFSR# | Primitive Polynomial Feedback Coefficients (U) |
|--------|--|
| LFSR-0 | U0 = [0,1,2,3,4,6,8,10,12,14,15,17,18,20,21,22,23,24,25,26,31,32,34,35,38,39,40,41,44,48,49,50,52,53,55, 58] |
| LFSR-1 | U1 = [0,1,2,3,4,5,8,9,10,11,12,14,15,16,19,21,23,24,25,26,28,29,30,31,32,33,35,36,38,44,45,46,48,50,53,55,56,60,61,63] |
| LFSR-2 | U2 = [0,1,2,3,8,9,12,13,17,21,23,25,26,28,29,31,34,35,38,43,47,49,50,56,57,58,59,60] |
| LFSR-3 | U3 = [0,1,2,4,5,11,13,14,16,19,20,22,23,24,27,28,29,33,36,38,40,41,42,43,44,47,48,49,51,55,57,58,61,63] |
| LFSR-4 | U4 = [0,3,7,8,9,10,11,14,16,21,23,24,26,29,31,32,33,36,41,43,44,45,46,47,50,51,52,55,56,59,60,61,62,63] |
| LFSR-5 | U5 = [0,3,7,8,10,11,13,15,16,19,20,21,23,27,28,30,33,34,36,38,39,41,42,43,45,48,49,50,51,55,60,62] |
| LFSR-6 | U6 = [0,1,2,3,4,5,7,8,9,10,12,19,20,21,23,24,27,30,31,34,35,37,38,39,40,42,45,47,48,51,53,54,56,58,60,61,62,63] |
| LFSR-7 | U7 = [0,2,3,4,5,7,11,13,15,18,20,24,25,33,34,37,38, 39,42,45,46,48,49,51,53,54,55,58,59,62] |

The output of eight LFSRs are concatenated to perform outputs OL of 8-bit to act as input to the next building block (Nonlinear Module) calculated in Eq. (4) as follows:

$$OL = \text{LFSR-0 (0)} \parallel \text{LFSR-1 (0)} \parallel \text{LFSR-2 (0)} \parallel \text{LFSR-3 (0)} \parallel \text{LFSR-4 (0)} \parallel \text{LFSR-5 (0)} \parallel \text{LFSR-6 (0)} \parallel \text{LFSR-7 (0)} \tag{4}$$

3.4 Nonlinear Module

This module provides the cipher algorithm with the required degree of non-linearity which grants strong resistance against known cryptographic (linear and differential) attacks. This module is a substitution Box Sbox (S1). It takes one byte as input and produces one byte (OS1) as an output with no linear relation between them.

The presented Sbox was built based on a power function with exponent ($\text{exp} = 4966$). Figure 3 shows the definition of the power function that is used to generate the Sbox. The Sbox properties are designed to have a maximum Walsh equal to 32, with zero immunity, maximum autocorrelation equal to 32, function PC (0), and 7th algebraic degree. The definition of the Scalar Boolean function is shown in Figure 4 demonstrated in the next section.

| | | | | | | | | |
|-----|------|-----|-----|-----|-----|-----|-----|-----|
| S1= | [197 | 196 | 75 | 49 | 130 | 98 | 191 | 127 |
| 104 | 88 | 24 | 93 | 248 | 111 | 152 | 83 | |
| 29 | 183 | 5 | 157 | 37 | 251 | 137 | 163 | |
| 85 | 27 | 144 | 69 | 101 | 70 | 142 | 239 | |
| 169 | 40 | 252 | 148 | 165 | 147 | 233 | 79 | |
| 181 | 21 | 218 | 143 | 227 | 78 | 246 | 171 | |
| 141 | 76 | 170 | 235 | 97 | 6 | 133 | 155 | |
| 149 | 231 | 10 | 108 | 110 | 201 | 208 | 36 | |
| 243 | 154 | 61 | 16 | 87 | 139 | 99 | 193 | |
| 245 | 77 | 238 | 219 | 211 | 162 | 128 | 86 | |
| 253 | 230 | 173 | 73 | 68 | 223 | 224 | 164 | |
| 214 | 4 | 14 | 166 | 82 | 203 | 242 | 132 | |
| 225 | 146 | 15 | 158 | 124 | 1 | 210 | 136 | |
| 151 | 72 | 42 | 118 | 229 | 41 | 234 | 247 | |
| 237 | 20 | 212 | 28 | 44 | 62 | 31 | 188 | |
| 30 | 178 | 195 | 126 | 65 | 8 | 59 | 57 | |
| 222 | 145 | 100 | 216 | 185 | 9 | 33 | 117 | |
| 140 | 244 | 226 | 232 | 150 | 172 | 199 | 48 | |
| 221 | 26 | 129 | 138 | 94 | 121 | 202 | 153 | |
| 206 | 25 | 120 | 81 | 105 | 204 | 2 | 103 | |
| 217 | 71 | 90 | 3 | 241 | 7 | 131 | 192 | |
| 11 | 254 | 200 | 249 | 89 | 205 | 123 | 114 | |
| 66 | 32 | 43 | 174 | 46 | 55 | 122 | 106 | |
| 0 | 161 | 194 | 190 | 80 | 95 | 107 | 115 | |
| 215 | 156 | 96 | 240 | 160 | 125 | 102 | 91 | |
| 23 | 50 | 167 | 159 | 64 | 184 | 109 | 255 | |
| 236 | 180 | 13 | 51 | 60 | 134 | 18 | 19 | |
| 213 | 182 | 179 | 189 | 92 | 207 | 220 | 84 | |
| 209 | 250 | 35 | 53 | 67 | 116 | 39 | 52 | |
| 63 | 177 | 54 | 113 | 168 | 228 | 119 | 175 | |
| 38 | 34 | 112 | 47 | 198 | 74 | 22 | 12 | |
| 135 | 17 | 45 | 176 | 186 | 58 | 187 | 56] | |

Fig. 3. New Stream Cipher (ANCHOR) Sbox S1

```
int boolean[256] = {1,0,0,1,0,1,
0,0,1,1,0,0,0,1,0,0,0,1,0,1,1,1,
0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1,
1,1,0,0,1,1,0,1,1,0,0,1,1,0,1,0,
1,1,0,1,1,1,0,1,0,1,0,1,0,0,1,0,
1,0,0,0,0,0,1,0,0,1,1,0,0,1,0,0,
1,0,1,0,0,0,1,1,1,1,1,0,0,0,1,1,
1,1,1,1,0,1,1,1,0,1,1,0,0,1,0,1,
1,1,1,1,0,1,0,1,1,1,1,1,0,0,1,1,
0,1,0,1,1,0,0,1,0,1,1,1,0,1,1,0,
1,1,0,0,1,1,1,1,0,0,1,1,1,0,0,0,
0,1,0,0,1,1,0,0,0,0,0,1,1,0,1,1,
0,0,1,0,0,0,1,0,0,1,0,0,1,0,0,1,
0,0,0,1,1,1,1,0,1,0,1,0,1,1,0,0,
0,0,0,0,1,1,1,0,0,0,0,1,1,0,0,0,
0,0,1,1,1,0,1,1,0,1,1,1,1,0,0,0,
1,0,0,1,1,1,1,0,1,1};
```

Fig. 4. New Scalar Boolean Function (SB)

3.5 Reduction Module

The function of the Reduction module is to provide the cipher algorithm with a non-linear reduction stage. The module is constructed from a balanced Scalar Boolean function (SB) with eight-bit input ($n=8$) and one-bit output. This function was constructed with high-valued cryptographic parameters.

The Scalar Boolean function linear characteristics are designed to have a zero-immunity order, Walsh Maximum equals 56, distance to linear function equals $100*(= 2^{**}(n-1) - \text{MaxW}/2)$ and $\text{PrMax}(f(x) = l(x)) = 1/2 (1 + 0.2188)$. Furthermore, the Scalar Boolean function differential characteristics are developed to have Maximum Autocorrelation equal to 72 and propagation criterion (PC) equal to zero. The definition of the Scalar Boolean function is shown in Figure 4. The output of this function is one bit which is considered the output keystream key (Z) of the proposed ANCHOR cipher algorithm.

3.6 Keystream Initialization Mode and Operation Mode

The ANCHOR cipher system's LFSR module is initialized once at the beginning of every new data transmission session or whenever the secret session key (K) is updated. An eight-bit Buffer was

implemented after the ANCHOR algorithm to generate an eight-bit key stream. The ANCHOR algorithm is initialized and operates according to the following steps:

- i. Key Expansion Module operates with the secret session key (K) and the Initialization Vector (IV) of the new plain text message is received to produce the round keys (Kr).
- ii. The eight-bit set of Linear Feedback Shift Registers (LFSRs) of ANCHOR cipher are filled at initialization mode with the eight 64-bit round keys (Kr).
- iii. The main 8-LFSRs are stepped regularly to shift the register one bit to the right after each rising edge of the clock (t). The feedback bit is calculated for each shift operation using the primitive polynomial feedback of each register and its functions.
- iv. The output byte from LFSRs is used as input to the Nonlinear module.
- v. The Sbox (S1) maps the input byte and produces an output byte that acts as input to the Reduction Module as the balanced Scalar Boolean function (SB) reduces the byte input to a one-bit key stream output of the ANCHOR algorithm.

4. Security Assessment

In order to ensure the security strength of the proposed stream cipher, some cryptographic criteria were tested to evaluate the randomness, unpredictability, and resistance against known cryptographic attacks. These tests included statistical tests and equivalent linear complexity. The ANCHOR algorithm was implemented using C language on a Linux machine with 8 GB memory and 8-core processors to produce a sample keystream in data files to be assessed.

4.1 Cryptographic Statistical Tests

For the purpose of testing the randomness property of the proposed cipher algorithm, CRYTX98 package has been used. The package consists of fifteen tests that were applied to the binary sequence taken as an output of the cipher algorithm. Figures 5 and 6 show the statistical test result.

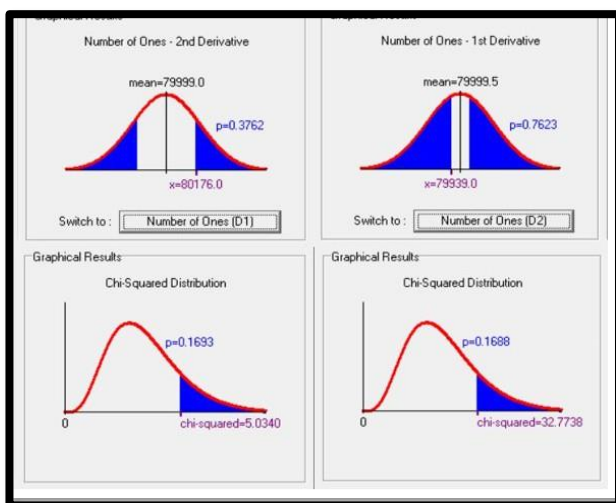


Fig. 5. Cryptographic Statistical Tests Output One

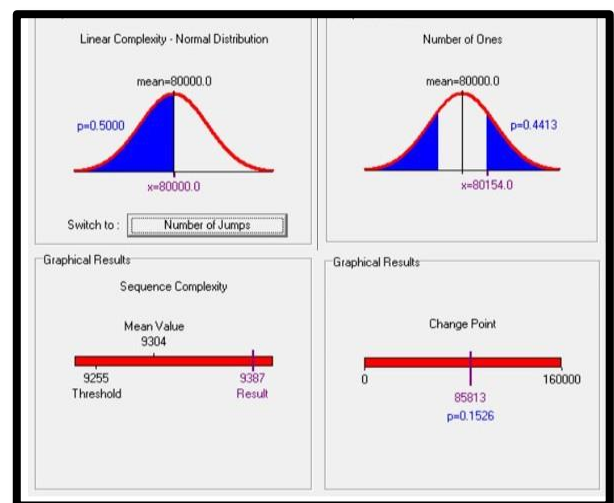


Fig. 6. Cryptographic Statistical Tests Output Two

A file of size 20,000-byte key (160,000-bit) in a binary format has been produced by the proposed cipher algorithm, with a non-zero random value for the initialization vector (IV). Accordingly, ANCHOR's Key stream output had successfully satisfied all statistical tests.

4.2 Linear Complexity Assessment

The Berlekamp-Massey algorithm is a mathematical algorithm used in the field of error correction coding. It was developed by Elwyn R. Berlekamp and James L. Massey in the early 1960s. The primary objective of the algorithm is to find the shortest linear feedback shift register (LFSR) that can generate a given sequence of bits.

The algorithm works by iteratively updating a candidate LFSR polynomial that generates the observed sequence of bits. Starting with an initial candidate polynomial of degree zero, the algorithm checks if the polynomial correctly predicts the next bit in the sequence. If it does, the degree of the polynomial remains the same, and the algorithm moves on to the next bit. If the polynomial fails to predict the next bit, the algorithm updates the polynomial to a new one with a higher degree.

The process continues until the algorithm finds the minimal-degree polynomial that can generate the entire sequence. The Berlekamp-Massey algorithm demonstrates the linear complexity of a sequence by finding the shortest LFSR that can produce the given sequence. Various samples of ANCHOR cipher keystream were generated to measure the equivalent Linear complexity. Table 2 shows the results of testing bit streams of various lengths. The relation between the tested inputs and the output equals to $(n/2)$ and satisfies the long cycle criterion.

Table 2
Linear Complexity Output for
ANCHOR Cipher

| Sample size (bit) | Linear complexity |
|-------------------|-------------------|
| 512 | 256 |
| 1024 | 532 |
| 2048 | 1064 |
| 4096 | 2024 |
| 8192 | 4096 |
| 16384 | 8192 |

5. Performance Assessment

Due to the limitation of the smart card resources, Performance assessment is an essential part of evaluating encryption algorithms' efficiency and suitability to work on the smart card. The two crucial aspects that are evaluated in performance assessment are speed and size. Speed refers to the time taken by an encryption algorithm to execute its encryption and decryption cycles. size refers to the memory, ROM, and storage space needed for a cipher algorithm to run on the smart card.

Encryption algorithms that are fast and require a small memory footprint are generally preferred. This is because fast algorithms can efficiently encrypt and decrypt large amounts of data in a short time, while small memory requirements enable them to be deployed in resource-constrained devices like smart cards and embedded systems.

5.1 Speed Assessment

The speed of processing is evaluated during the ANCHOR encryption and decryption process testing. Two test data of sizes 800 bytes and 1 kilobyte were used in this test to determine how the size of the test data affected the speed of the encryption and decryption processes. In this test, the built-in system is run to determine how long the encryption and decryption processes take, and the system then displays the results shown in Figures 7 and 8.

```

kali@kali:~$ ./anchor
Execution Time: 0.001808 seconds
kali@kali:~$ ./anchor
Execution Time: 0.001807 seconds
kali@kali:~$ ./anchor
Execution Time: 0.001807 seconds
kali@kali:~$ ./anchor
Execution Time: 0.001809 seconds
kali@kali:~$ ./anchor
Execution Time: 0.001807 seconds
kali@kali:~$ ./anchor
Execution Time: 0.001808 seconds
kali@kali:~$ ./anchor
Execution Time: 0.001807 seconds
kali@kali:~$ ./anchor
Execution Time: 0.001874 seconds
kali@kali:~$ ./anchor
Execution Time: 0.001808 seconds
kali@kali:~$ ./anchor
Execution Time: 0.001879 seconds
kali@kali:~$ ./anchor
Execution Time: 0.001808 seconds
    
```

Fig. 7. ANCOR Cipher Speed Assessment 0.8 kb Output

```

kali@kali:~$ ./anchor
Size of keys generated: 1 KB
Execution Time: 0.002317 seconds
kali@kali:~$ ./anchor
Size of keys generated: 1 KB
Execution Time: 0.002536 seconds
kali@kali:~$ ./anchor
Size of keys generated: 1 KB
Execution Time: 0.002768 seconds
kali@kali:~$ ./anchor
Size of keys generated: 1 KB
Execution Time: 0.002374 seconds
kali@kali:~$ ./anchor
Size of keys generated: 1 KB
Execution Time: 0.002539 seconds
kali@kali:~$ ./anchor
Size of keys generated: 1 KB
Execution Time: 0.002620 seconds
kali@kali:~$ ./anchor
Size of keys generated: 1 KB
Execution Time: 0.002589 seconds
kali@kali:~$ ./anchor
Size of keys generated: 1 KB
Execution Time: 0.002774 seconds
kali@kali:~$ ./anchor
Size of keys generated: 1 KB
Execution Time: 0.002546 seconds
kali@kali:~$ ./anchor
Size of keys generated: 1 KB
Execution Time: 0.002650 seconds
    
```

Fig. 8. ANCOR Cipher Speed Assessment 1.0 kb Output

The same data was subjected to ten experiments to determine the average processing time for the ANCHOR encryption and decryption procedure regarding speed assessment results as Table 3.

Table 3
 Speed Assessment Results as Experiments in
 Milliseconds (ms)

| | Number/Size | 800 bytes | 1 kilobyte |
|--------------|-------------|-----------|------------|
| | 1 | 1.8ms | 2.3ms |
| | 2 | 1.8ms | 2.5ms |
| | 3 | 1.8ms | 2.7ms |
| | 4 | 1.8ms | 2.3ms |
| | 5 | 1.8ms | 2.5ms |
| | 6 | 1.8ms | 2.6ms |
| | 7 | 1.8ms | 2.5ms |
| | 8 | 1.8ms | 2.7ms |
| | 9 | 1.8ms | 2.5ms |
| | 10 | 1.8ms | 2.6ms |
| Average Time | | 1.8ms | 2.36ms |

The ANCHOR encryption and decryption procedure on test data with a total size of 0.8 KB takes an average of 1.8ms and for 1 KB takes an average of 2.3ms. The outcomes of this test demonstrate that ANCHOR can perform the encryption and decryption procedure in a very brief amount of time, making it suitable for usage with smart cards with an average speed rate of Eq. (5) as follows

$$KB/s = 1 \text{ KB} / 0.00236 \text{ seconds} = 423.72881 \text{ KB/s} \tag{5}$$

5.2 Memory Usage

The ANCHOR cipher underwent comprehensive testing to evaluate its memory consumption. The cipher consists of three modules: eight LFSRs (Linear Feedback Shift Registers), each with a size of 64 bits, a Sbox, and a Scalar Boolean component. To assess the total memory usage of the ANCHOR algorithm, it was implemented in the C programming language on a Linux machine.

The results of the testing, as depicted in Figure 9, revealed that the ANCHOR algorithm consumes a mere 2.692 KB of memory on the Linux machine. This finding highlights the remarkable efficiency of the ANCHOR cipher in terms of memory utilization. In fact, it stands out as one of the cryptographic ciphers with the smallest memory footprints.

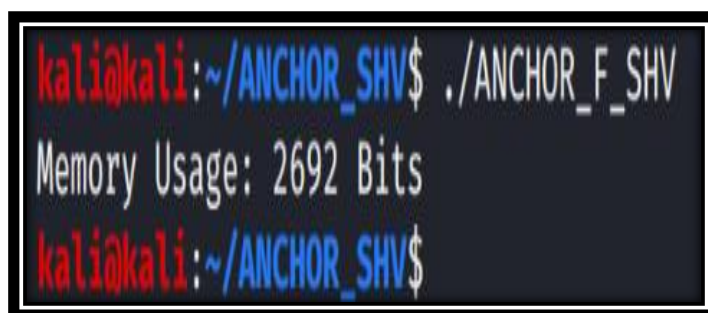
A terminal window screenshot showing the execution of a script. The prompt is 'kali@kali:~/ANCHOR_SHV\$./ANCHOR_F_SHV'. The output is 'Memory Usage: 2692 Bits'. The prompt then returns to 'kali@kali:~/ANCHOR_SHV\$'.

Fig. 9. ANCOR Cipher Memory Usage Assessment Output

The compact memory fingerprint of the ANCHOR cipher is a significant advantage in various applications. In resource-constrained environments, such as Smart Cards which are devices with limited memory capacities, the algorithm's low memory consumption is highly desirable. It allows for efficient utilization of system resources without compromising the security or performance of cryptographic operations.

The ANCHOR cipher's small memory footprint not only contributes to reduced resource requirements but also facilitates its integration into diverse computing environments. The algorithm can be seamlessly implemented in systems with restricted memory availability, ensuring its compatibility and usability across a wide range of platforms and devices. Consequently, the ANCHOR cipher offers an excellent balance between strong cryptographic security and efficient memory utilization, making it a compelling choice for applications where memory efficiency is a crucial consideration.

6. Conclusions

This paper introduces a new encryption algorithm, ANCHOR. The purpose of ANCHOR cipher is not only to encrypt and decrypt the data stored and used in smart cards but also to improve the overall security of information transmitted and stored on smart cards, making it difficult for intruders to exploit the data. The simplicity and efficiency of this algorithm make it better suited to be applied to smart cards.

In conclusion, the ANCHOR cipher has demonstrated its strength as a competitive algorithm through its successful completion of NIST cryptographic tests and its exceptional performance results. Moreover, it has the potential to be implemented in smart card crypto co-processors. As a next step, the ANCHOR algorithm is planned to be implemented in VHDL code, which can be used to develop Field-Programmable Gate Array (FPGA) hardware which will enable researchers to create a specialized chip using ASIC technology.

Acknowledgment

Thanks, and gratitude to everyone who provided me with assistance, advice, and support so that I could finish this research. I especially thank my professors and those in charge of supervising the research. Thank you very much for your support and for giving me an opportunity to present my research to you, which demonstrates the sincerity of my intentions to support those aspiring for knowledge. This research was not funded by a grant.

References

- [1] Gupta, Kishan Chand, Sumit Kumar Pandey, and Susanta Samanta. "FUTURE: a lightweight block cipher using an optimal diffusion matrix." In *International Conference on Cryptology in Africa*, pp. 28-52. Cham: Springer Nature Switzerland, 2022. https://doi.org/10.1007/978-3-031-17433-9_2
- [2] Nurwarsito, Heru, and Sarah Kusuma Ayu. "Comparison Analysis of Acorn Algorithm and Snow Algorithm on Smart Card using Java Card." In *2021 4th International Conference of Computer and Informatics Engineering (IC2IE)*, pp. 429-434. IEEE, 2021. <https://doi.org/10.1109/IC2IE53219.2021.9649079>
- [3] Adiwiganda, Muhammad Rizki, Endro Ariyanto, Rahmat Yasirandi, Novian Anggis Suwastika, and Yoso Adi Setyoko. "Adopting Tiny Encryption Algorithm for Patient Healthcare Record on Smart Card." In *2019 International Conference of Computer Science and Information Technology (ICoSNIKOM)*, pp. 1-5. IEEE, 2019. <https://doi.org/10.1109/ICoSNIKOM48755.2019.9111534>
- [4] Nivetha, S., N. Edna Elizabeth, T. Prasanya Padmasha, and I. Gohulalakshmi. "Secure authentication process in smart cards." In *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, pp. 1-5. IEEE, 2016. <https://doi.org/10.1109/ISCO.2016.7726879>
- [5] Kanda, Guard, and Kwangki Ryoo. "Vedic Multiplier-based International Data Encryption Algorithm Crypto-Core for Efficient Hardware Multiphase Encryption Design." *Webology* 19, no. 1 (2022): 4581-4596. <https://doi.org/10.14704/WEB/V19I1/WEB19304>
- [6] Selimis, G., N. Sklavos, and O. Koufopavlou. "Crypto processor for contactless smart cards." In *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference (IEEE Cat. No. 04CH37521)*, vol. 2, pp. 803-806. IEEE, 2004.
- [7] Mittal, Himanshu. "Diffie-Hellman based smart-card multi-server authentication scheme." In *2014 International Conference on Computational Intelligence and Communication Networks*, pp. 808-812. IEEE, 2014. <https://doi.org/10.1109/CICN.2014.173>
- [8] Eslami, Yadollah, Ali Sheikholeslami, P. Glenn Gulak, Shoichi Masui, and Kenji Mukaida. "An area-efficient universal cryptography processor for smart cards." *IEEE transactions on very large scale integration (VLSI) systems* 14, no. 1 (2006): 43-56. <https://doi.org/10.1109/TVLSI.2005.863188>
- [9] Ege, Barış, Elif Bilge Kavun, and Tolga Yalçın. "Memory encryption for smart cards." In *Smart Card Research and Advanced Applications: 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers 10*, pp. 199-216. Springer Berlin Heidelberg, 2011. https://doi.org/10.1007/978-3-642-27257-8_13
- [10] Muthulakshmi, S., and R. Chitra. "Enhanced data privacy algorithm to protect the data in smart grid." In *2021 Smart Technologies, Communication and Robotics (STCR)*, pp. 1-4. IEEE, 2021.
- [11] Schaefer, Edward F. "A simplified data encryption standard algorithm." *Cryptologia* 20, no. 1 (1996): 77-84. <https://doi.org/10.1080/0161-119691884799>
- [12] Gudodagi, Raveendra, and R. Venkata Siva Reddy. "Security Provisioning and Compression of Diverse Genomic Data based on Advanced Encryption Standard (AES) Algorithm."
- [13] Gilbert, Shirley. "RFID Security: Tiny Encryption Algorithm and Authentication Protocols." *Master Project, Ryerson University, Toronto, Canada* (2009).
- [14] Aradhyamath, S., and J. Paulose. "Multi-key Modified Tiny Encryption Algorithm for HealthCare." *International Journal of Engineering & Technology* 7, no. 2 (2018): 559-563. <https://doi.org/10.14419/ijet.v7i2.9894>
- [15] Yassein, Muneer Bani, Shadi Aljawarneh, Ethar Qawasmeh, Wail Mardini, and Yaser Khamayseh. "Comprehensive study of symmetric key and asymmetric key encryption algorithms." In *2017 international conference on engineering and technology (ICET)*, pp. 1-7. IEEE, 2017. <https://doi.org/10.1109/ICEngTechnol.2017.8308215>
- [16] Sun, Qimin, Jongho Moon, Younsung Choi, and Dongho Won. "Cryptanalysis of Smart Card Based Remote User Authentication Scheme for Multi-Server Environment." (2016). <https://doi.org/10.12792/iciae2016.031>
- [17] Zhu, Yuesheng, Bojun Wang, and Cheng Cai. "A Novel Smart-Card Based Authentication Scheme Using Proactive Secret Sharing." *International Journal of Computer and Communication Engineering* 5, no. 3 (2016): 196. <https://doi.org/10.17706/IJCCE.2016.5.3.196-205>

- [18] Albrecht, Martin R., Christian Hanser, Andrea Hoeller, Thomas Pöppelmann, Fernando Virdia, and Andreas Wallner. "Implementing RLWE-based schemes using an RSA co-processor." *Cryptology ePrint Archive* (2018). <https://doi.org/10.46586/tches.v2019.i1.169-208>
- [19] Gupta, Brij B., and Megha Quamara. *Smart Card Security: Applications, Attacks, and Countermeasures*. CRC Press, 2019. <https://doi.org/10.1201/9780429345593>
- [20] Almatarneh, Akram. "Ethics and Smart Governments: The Case of National ID Smart Card in the United Arab Emirates."
- [21] Jagadeesh, Selvaraj, Sabna Machinchery Ali, Soundara Pandian Gnana Selvan, Mohammad Aljanabi, and Manimaran Gopianand. "Hybrid AES-Modified ECC Algorithm for Improved Data Security over Cloud Storage." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 32, no. 1 (2023): 46-56. <https://doi.org/10.37934/araset.32.1.4656>
- [22] Joseph Ng, P. S., EricMok, Z. C., Phan, K. Y., Sun, J., & Wei, Z. "Mitigating Social Media Cybercrime: Revolutionising with AES Encryption and Generative AI", *Journal of Advanced Research in Applied Sciences and Engineering Technology*, (2024).