DOI: 10.2478/ias-2024-0018 • JIAS • 19(6) • 2024 • 266-283

Fermat Encoding in S-Box for Secured Encryption

Md. Hasanujjaman[1], Partha Sarathi Goswami[2]

1 Department of Computer Software Technology, Sheikhpara ARM Polytechnic, West Bengal, India, Pin: 742409: *biddyut_1983@rediffmail.com*

2 Department of Cyber Forensics and Information Security, Behala Government Polytechnic, Kolkata, India, Pin: 700060: *goswamipsg@rediffmail.com*

Abstract: Today, cybersecurity is the foremost concern for a communication network. In this article, an encryption structure is projected using the AES method to convert a symmetric key to an asymmetric key by encrypting an S-Box. To enhance the level of encryption, Fermat encoding is used. Thus, cryptographic security hinges on the robustness of these S-Boxes, which contribute to the strength of the encryption algorithms. The study systematically examines the characteristics, security attributes, and applications of S-Boxes, thereby providing insights into their roles and effectiveness in safeguarding sensitive information. Findings herein contribute to informed decision-making in selecting the appropriate type of S-Boxes based on specific security requirements, paving the way for enhanced cryptographic design and implementation.

Keywords: Cryptography, Information Security, S-Boxes, Cyber Security, Network, Network Security.

1 Introduction

An S-box, short for "Substitution Box," is a key factor in modern symmetric-key cryptography. It is a mathematical function that operates on a fixed number of bits, typically 8, 16, or 32 and is used to perform a one-to-one substitution of input values into output values based on a predefined lookup table. In block ciphers, the S-box is extensively used to guarantee confusion property defined by C.E. Shannon for hiding the link between the key and the encoded data [1,2]. S-boxes are vital in cryptographic algorithms like the Advanced Encryption Standard (AES) [3] and the Data Encryption Standard (DES) to familiarize confusion and non-linearity, enhancing the security of encrypted data.

Fermat numbers are a sequence of integers that are defined by the mathematician Pierre de Fermat. It takes the form $F_n = 2^{2^n} + 1$, where "n" is a non-negative integer.

In other words, each Fermat number is obtained by raising 2 to the power of 2 raised to another power and then adding 1 to the result. The first few Fermat numbers are:

$$F_0 = 2^{2^0} + 1 = 3,$$

$$F_1 = 2^{2^1} + 1 = 5,$$

$$F_2 = 2^{2^2} + 1 = 17,$$

$$F_3 = 2^{2^3} + 1 = 257,$$

Fermat numbers have been of interest to mathematicians for centuries, partly due to Fermat's interest in them and his correspondence with other mathematicians about their properties. However, they are not all prime numbers; in fact, it is known that F_4 and beyond are composite (not prime).

2 Literature Review

In 2018, Dragan Lambić [4] proposed a method to acquire random bijective S-boxes based on better one-dimensional discrete chaotic maps. To get over the difficulty of the possibly short length of the orbits of the former discrete-space chaotic map, a special case based on the composition of permutations and sine functions was used. The map was defined over a finite set, which makes it

suitable for use on digital computers. In 2020, Eslam W. Afify et al. [5] presented a study of the three types of S-Box by performance analysis and comparative performance study. It showed how an algebraic attack is the most secure lapse of the AES S-box and proposed a modified S-box by enhancing the algebraic properties and Resistance Algebraic Attack (RAA). In 2021, Sarah Mohammed Abdullah and Iman Qays Abduljaleel [6] recommended a reliable scheme for sending speeches. The method had three phases. The first phase is to jumble the speech signal by separating the speech signal into blocks of various sizes of 256 and 128 instances and sorting blocks into matrices in accordant to the length to obtain two matrices then each matrix is divided as par its size into two parts, left and right and rotate only the right side so that the row is a column. The column is a row, while on the other side, a chessboard is applied. The second phase is DNA encryption of the disorganized speech signal by exploitation of a second rule to obtain an enciphered speech signal. The final phase consists of encoding with a Substitution Box that has multiple chaotic maps. In 2021, R. Sherine Jenny, R. Sudhakar, and M. Karthikpriya [7] presented the compact implementation of PRESENT, GIFT and LED, KLEIN, and RECTANGLE S-box. Karnaugh mapping was used to implement the Boolean S-box. In 2021, Ahmad Y. Al-Dweik et al. [8] made a key-dependent dynamic n × n clone S-Box, which has some algebraic properties that give a stronger S-Box to an AES 8 × 8 S-Box. In 2022, Sarita Sanap and Vijayshree More [9] proposed an optimized residue of prime number S-box for AES that gave 13% enhancement with respect to filled slices. The LUTs were maximized by 12.42%. In 2022, A. H. Zahid et al. [10] projected a simple and innovative scheme for the creation of highly nonlinear S-boxes with a cubic modular transformation along with the trigonometric Sine function. A heuristic-based improvement approach, which is random in nature, is proposed to amend the non-linearity of the first S-box. The method uses input factor of integer type in alteration and improvement phases that have the possibility to make a large count of sturdy S-boxes when a small alteration is applied in the parameters' values. The cipher key is utilized to apply values to the input constant for the creation of dynamic S-boxes. In 2023, Rahul Mishra, Bhupendra Singh, and Radhakrishnan Delhibabu [11] proposed the construction of the AES S-box and explored the possibility of finding S-boxes with similar or better cryptographic properties using a genetic algorithm-based approach applied to one of its elementary. In 2023, Abeer Tariq Maolood et al. [12] created a 4-bit S-box that provides DSAC ideal values equal to zero. the S-box also applied DNA coding to extend each S-box into eight S-boxes, which increases the efficiency of the encrypted images. In 2023 Md. Hasanujjaman, Partha Sarathi Goswami, et al. [13] proposed a new substitution box scheme amalgamating with the concept of Fermat encoding to obtain a secured message-passing technique.

3 Proposed Methodology

Algorithm:

Step 1: Take an 8x8 matrix as 'A' and an 8X1 matrix as 'b'

Step 2: Calculate 256 distinct values using the formula

$$S(x) = Ax + b \tag{1}$$

Step 3: Make a S-box with the 256 distinct values obtained.

Encryption of S-Box (Sbox)

Step 1: Take all the values from the S-box {S(0) … S(255)}

Step 2: Perform

$S(0)' = S(0) \oplus S(1)$

$S(1)' = S(1) \oplus S(2)$

$S(2)' = S(2) \oplus S(3)$

… …

$S(255)' = S(255) \oplus S(0)'$

where $\oplus$ is the tensor product [14].

Step 3: Generate S'_{box} with the 256 values.

Step 4: Obtain the cipher text using Fermat encoding to generate a new encrypted S-box (S'_{box}).

Decryption of the S-Box

Step 1: Calculate the values of S-box values forward with the help of Fermat numbers and calculate the values of the S-Box from S(0)′ to S(255)′

Step 2: Make a decrypted S-box with the values.

Step 3: Calculate the values of the S(0) to S(255) with the
the help of XOR operation

$S(255) = S(255)' \oplus S(0)'$
$S(254) = S(254)' \oplus S(255)'$
$S(253) = S(253)' \oplus S(254)'$
… …
$S(0) = S(0)' \oplus S(1)'$

Step 4: After this step, make a value of S(0) to S(255) and get the decrypted S-box again.

AES is a symmetric key where a single private key is used for both the encryption and decryption processes. But here, the AES is so designed that it becomes asymmetric, and the encrypted S-box is used as a public key.

The flow diagram of Encryption and Decryption of the S-box is shown in Figure 1 and Figure 2 respectively.

Flow Diagram:

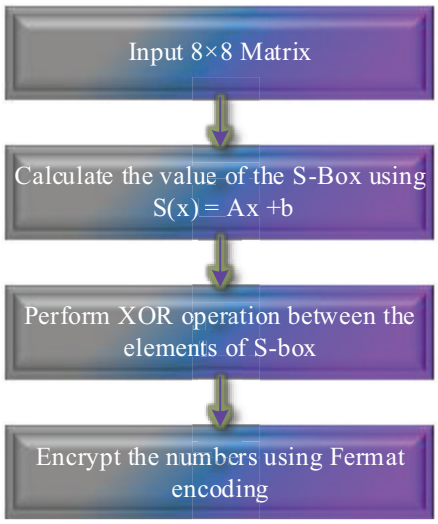

Figure 1. Encryption of the S-box

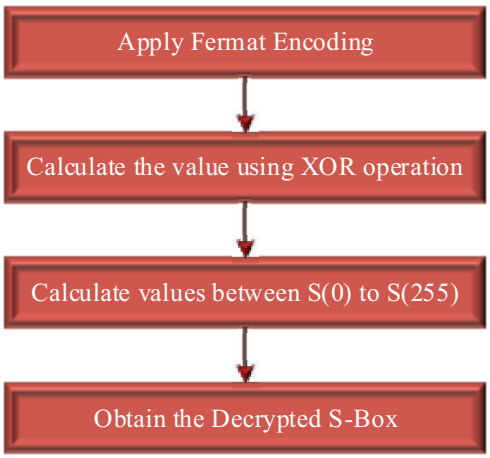

Figure 2. Decryption of the S-box

4 Illustration

The main objective is to create one non-linear S-box [15,16] and then encrypt the S-box as an asymmetric key. To do this, create an 8x8 S-box and then encrypt the S-box. Use the S-box so obtained as a public key in the AES encryption method. The encryption process is illustrated below with an example.

$z_2^8 = \{0,1\}\times\{0,1\}\times\{0,1\}\times\{0,1\}\times\{0,1\}\times\{0,1\}\times\{0,1\}\times\{0,1\}$ = *{ 0 0 0 0 0 0 0 0 , 0 0 0 0 0 0 0 1 , 00000010, 00000011, 00000100, 00000101, 00000110, 00000111, 00001000, 00001001, 00001010, 00001011, 00001100, 00001101, 00001110, 00001111, 00010000, 00010001, 00010010, 00010011, 00010100, 00010101, 00010110, 00010111, 00011000, 00011001, 00011010, 00011011, 00011100, 00011101, 00011110, 00011111, 00100000, 00100001, 00100010, 00100011, 00100100, 00100101, 00100110, 00100111, 00101000, 00101001, 00101010, 00101011, 00101100, 00101101, 00101110, 00101111, 00110000, 00110001, 00110010, 00110011, 00110100, 00110101, 00110110, 00110111, 00111000, 00111001, 00111010, 00111011, 00111100, 00111101, 00111110, 00111111, 01000000, 01000001, 01000010, 01000011, 01000100, 01000101, 01000110, 01000111, 01001000, 01001001, 01001010, 01001011, 01001100, 01001101, 01001110, 01001111, 01010000, 01010001, 01010010, 01010011, 01010100, 01010101, 01010110, 01010111, 01011000, 01011001, 01011010, 01011011, 01011100, 01011101, 01011110, 01011111, 01100000, 01100001, 01100010, 01100011, 01100100, 01100101, 01100110, 01100111, 01101000, 01101001, 01101010, 01101011, 01101100, 01101101, 01101110, 01101111, 01110000, 01110001, 01110010, 01110011, 01110100, 01110101, 01110110, 01110111, 01111000, 01111001, 01111010, 01111011, 01111100, 01111101, 01111110, 01111111, ……,11111111}*

Take any arbitrary values of A and b from the products such that A X b is unique and $|A| \neq 0$ for each session. Let,

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Calculate values of S(0) to S(255) and make the general S-box

$$(0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$S(1) = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$S(2)=\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix}\begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix}\oplus\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}=\begin{bmatrix}0\\0\\1\\0\\0\\1\\0\\1\end{bmatrix}$$

……

……

$$S(254)=\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\0\end{bmatrix}\oplus\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}=\begin{bmatrix}1\\1\\0\\0\\1\\0\\0\\0\end{bmatrix}$$

$$S(255)=\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix}\oplus\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}=\begin{bmatrix}0\\0\\1\\1\\1\\0\\0\\1\end{bmatrix}$$

$$S(255)=\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix}\oplus\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}=\begin{bmatrix}0\\1\\1\\1\\0\\0\\1\\1\end{bmatrix}$$

The binary encrypted S-box so generated is shown in table 1.

Table 1. S-box Generation

11110001,	00010010,	11110001,	11010101,	11110001,
00010010,	11110001,	01011010,	11110001,	00010010,
11110001,	11010101,	11110001,	00010010,	11110001,
01000101,	11110001,	00010010,	11110001,	11010101,
11110001,	00010010,	11110001,	01011010,	11110001,
00010010,	11110001,	11010101,	11110001,	00010010,
11110001,	01111011,	11110001,	00010010,	11110001,
11010101,	11110001,	00010010,	11110001,	01011010,
11110001,	00010010,	11110001,	11010101,	11110001,
00010010,	11110001,	01000101,	11110001,	00010010,
11110001,	11010101,	11110001,	00010010,	11110001,
01011010,	11110001,	00010010,	11110001,	11010101,
11110001,	00010010,	11110001,	00000111,	11110001,
00010010,	11110001,	11010101,	11110001,	00010010,
11110001,	01011010,	11110001,	00010010,	11110001,
11010101,	11110001,	00010010,	11110001,	01000101,
11110001,	00010010,	11110001,	11010101,	11110001,
00010010,	11110001,	01011010,	11110001,	00010010,
11110001,	11010101,	11110001,	00010010,	11110001,
01111011,	11110001,	00010010,	11110001,	11010101,
11110001,	00010010,	11110001,	01011010,	11110001,
00010010,	11110001,	11010101,	11110001,	00010010,
11110001,	01000101,	11110001,	00010010,	11110001,
11010101,	11110001,	00010010,	11110001,	01011010,
11110001,	00010010,	11110001,	11010101,	11110001,
00010010,	11110001,	11111111,	11110001,	00010010,
11110001,	11010101,	11110001,	00010010,	11110001,
01011010,	11110001,	00010010,	11110001,	11010101,
11110001,	00010010,	11110001,	01000101,	11110001,
00010010,	11110001,	11010101,	11110001,	00010010,
11110001,	01011010,	11110001,	00010010,	11110001,
11010101,	11110001,	00010010,	11110001,	01111011,
11110001,	00010010,	11110001,	11010101,	11110001,
00010010,	11110001,	01011010,	11110001,	00010010,
11110001,	11010101,	11110001,	00010010,	11110001,
01000101,	11110001,	00010010,	11110001,	11010101,
11110001,	00010010,	11110001,	01011010,	11110001,
00010010,	11110001,	11010101,	11110001,	00010010,
11110001,	00000111,	11110001,	00010010,	11110001,
11010101,	11110001,	00010010,	11110001,	01011010,
11110001,	00010010,	11110001,	11010101,	11110001,
00010010,	11110001,	01000101,	11110001,	00010010,
11110001,	11010101,	11110001,	00010010,	11110001,
01011010,	11110001,	00010010,	11110001,	11010101,

11110001,	00010010,	11110001,	01111011,	11110001,
00010010,	11110001,	11010101,	11110001,	00010010,
11110001,	01011010,	11110001,	00010010,	11110001,
11010101,	11110001,	00010010,	11110001,	01000101,
11110001,	00010010,	11110001,	11010101,	11110001,
00010010,	11110001,	01011010,	11110001,	00010010,
11110001,	11010101,	11110001,	00010010,	11110001,
11001000				

Now encrypt the S-box a 2nd time using the Fermat encoding process [17].

$$F_0 = 2^{2^0} + 1 = 3,$$

$$F_1 = 2^{2^1} + 1 = 5,$$

$$F_2 = 2^{2^2} + 1 = 17,$$

$$F_3 = 2^{2^3} + 1 = 257 \bmod 256 = 1,$$

$$F_4 = 2^{2^4} + 1 = 65537 \bmod 256 = 1,$$

$$F_5 = 2^{2^5} + 1 = 4294967297 \bmod 256 = 1,$$

$$F_6 = 2^{2^6} + 1 = 18446744073709551617 \bmod 256 = 1,$$

Encoding the numbers using Fermat encoding (backward method) is shown in table 2 below.

Table 2. Fermat encoding using backward method

Index = 0	**Index = 1**	**Index = 2**
241 - 3 = 238	18 - 3 = 15	241 - 3 = 238
238 - 5 = 233	15 - 5 = 10	238 - 5 = 233
233 - 17 = 216	10 - 17 = 249	233 - 17 = 216
216 - 1 = 215	249 - 1 = 248	216 - 1 = 215
215 - 1 = 214	248 - 1 = 247	215 - 1 = 214
214 - 1 = 213	247 - 1 = 246	214 - 1 = 213
213 - 1 = 212	246 - 1 = 245	213 - 1 = 212
212 - 1 = 211	245 - 1 = 244	212 - 1 = 211
Index = 3	**Index = 4**	**Index = 5**
213 - 3 = 210	241 - 3 = 238	18 - 3 = 15
210 - 5 = 205	238 - 5 = 233	15 - 5 = 10
205 - 17 = 188	233 - 17 = 216	10 - 17 = 249
188 - 1 = 187	216 - 1 = 215	249 - 1 = 248
187 - 1 = 186	215 - 1 = 214	248 - 1 = 247
186 - 1 = 185	214 - 1 = 213	247 - 1 = 246
185 - 1 = 184	213 - 1 = 212	246 - 1 = 245
184 - 1 = 183	212 - 1 = 211	245 - 1 = 244
Index = 6	**Index =7**	**Index = 8**
241 - 3 = 238	90 - 3 = 87	241 - 3 = 238
238 - 5 = 233	87 - 5 = 82	238 - 5 = 233
233 - 17 = 216	82 - 17 = 65	233 - 17 = 216
216 - 1 = 215	65 - 1 = 64	216 - 1 = 215
215 - 1 = 214	64 - 1 = 63	215 - 1 = 214
214 - 1 = 213	63 - 1 = 62	214 - 1 = 213
213 - 1 = 212	62 - 1 = 61	213 - 1 = 212
212 - 1 = 211	61 - 1 = 60	212 - 1 = 211
……..	……	……..
……..	……	……..

Index = 247	**Index =248**	**Index = 249**
90 - 3 = 87	241 - 3 = 238	18 - 3 = 15
87 - 5 = 82	238 - 5 = 233	15 - 5 = 10
82 - 17 = 65	233 - 17 = 216	10 - 17 = 249
65 - 1 = 64	216 - 1 = 215	249 - 1 = 248
64 - 1 = 63	215 - 1 = 214	248 - 1 = 247
63 - 1 = 62	214 - 1 = 213	247 - 1 = 246
62 - 1 = 61	213 - 1 = 212	246 - 1 = 245
61 - 1 = 60	212 - 1 = 211	245 - 1 = 244
Index = 250	**Index = 251**	**Index = 252**
241 - 3 = 238	213 - 3 = 210	241 - 3 = 238
238 - 5 = 233	210 - 5 = 205	238 - 5 = 233
233 - 17 = 216	205 - 17 = 188	233 - 17 = 216
216 - 1 = 215	188 - 1 = 187	216 - 1 = 215
215 - 1 = 214	187 - 1 = 186	215 - 1 = 214
214 - 1 = 213	186 - 1 = 185	214 - 1 = 213
213 - 1 = 212	185 - 1 = 184	213 - 1 = 212
212 - 1 = 211	184 - 1 = 183	212 - 1 = 211
Index =253	**Index = 254**	**Index = 255**
18 - 3 = 15	241 - 3 = 238	200 - 3 = 197
15 - 5 = 10	238 - 5 = 233	197 - 5 = 192
10 - 17 = 249	233 - 17 = 216	192 - 17 = 175
249 - 1 = 248	216 - 1 = 215	175 - 1 = 174
248 - 1 = 247	215 - 1 = 214	174 - 1 = 173
247 - 1 = 246	214 - 1 = 213	173 - 1 = 172
246 - 1 = 245	213 - 1 = 212	172 - 1 = 171
245 - 1 = 244	212 - 1 = 211	171 - 1 = 170

Thus, the encrypted S-box so obtained is shown in table 3.

Table 3. Encrypted S-Box

211	244	211	183	211	244	211	60	211
244	211	183	211	244	211	39	211	244
211	183	211	244	211	60	211	244	211
183	211	244	211	93	211	244	211	183
211	244	211	60	211	244	211	183	211
244	211	39	211	244	211	183	211	244
211	60	211	244	211	183	211	244	211
233	211	244	211	183	211	244	211	60
211	244	211	183	211	244	211	39	211
244	211	183	211	244	211	60	211	244
211	183	211	244	211	93	211	244	211
183	211	244	211	60	211	244	211	183
211	244	211	39	211	244	211	183	211
244	211	60	211	244	211	183	211	244
211	225	211	244	211	183	211	244	211
60	211	244	211	183	211	244	211	39
211	244	211	183	211	244	211	60	211
244	211	183	211	244	211	93	211	244
211	183	211	244	211	60	211	244	211
183	211	244	211	39	211	244	211	183
211	244	211	60	211	244	211	183	211
244	211	233	211	244	211	183	211	244
211	60	211	244	211	183	211	244	211
39	211	244	211	183	211	244	211	60

211	244	211	183	211	244	211	93	211
244	211	183	211	244	211	60	211	244
211	183	211	244	211	39	211	244	211
183	211	244	211	60	211	244	211	183
211	244	211	170					

Decoding the numbers using the Fermat encoding scheme (forward method) is shown in Table 4 below.

Table 4. Decryption using Fermat encoding in forward method

Index = 0	**Index = 1**	**Index = 2**
211 + 3 = 214	244 + 3 = 247	211 + 3 = 214
214 + 5 = 219	247 + 5 = 252	214 + 5 = 219
219 + 17 = 236	252 + 17 = 13	219 + 17 = 236
236 + 1 = 237	13 + 1 = 14	236 + 1 = 237
237 + 1 = 238	14 + 1 = 15	237 + 1 = 238
238 + 1 = 239	15 + 1 = 16	238 + 1 = 239
239 + 1 = 240	16 + 1 = 17	239 + 1 = 240
240 + 1 = 241	17 + 1 = 18	240 + 1 = 241
Index = 3	**Index = 4**	**Index = 5**
183 + 3 = 186	211 + 3 = 214	244 + 3 = 247
186 + 5 = 191	214 + 5 = 219	247 + 5 = 252
191 + 17 = 208	219 + 17 = 236	252 + 17 = 13
208 + 1 = 209	236 + 1 = 237	13 + 1 = 14
209 + 1 = 210	237 + 1 = 238	14 + 1 = 15
210 + 1 = 211	238 + 1 = 239	15 + 1 = 16
211 + 1 = 212	239 + 1 = 240	16 + 1 = 17
212 + 1 = 213	240 + 1 = 241	17 + 1 = 18
Index = 6	**Index =7**	**Index = 8**
211 + 3 = 214	60 + 3 = 63	211 + 3 = 214
214 + 5 = 219	63 + 5 = 68	214 + 5 = 219
219 + 17 = 236	68 + 17 = 85	219 + 17 = 236
236 + 1 = 237	85 + 1 = 86	236 + 1 = 237
237 + 1 = 238	86 + 1 = 87	237 + 1 = 238
238 + 1 = 239	87 + 1 = 88	238 + 1 = 239
239 + 1 = 240	88 + 1 = 89	239 + 1 = 240
240 + 1 = 241	89 + 1 = 90	240 + 1 = 241
……..	……	……..
……..	……	……..
Index = 247	**Index =248**	**Index = 249**
60 + 3 = 63	211 + 3 = 214	244 + 3 = 247
63 + 5 = 68	214 + 5 = 219	247 + 5 = 252
68 + 17 = 85	219 + 17 = 236	252 + 17 = 13
85 + 1 = 86	236 + 1 = 237	13 + 1 = 14
86 + 1 = 87	237 + 1 = 238	14 + 1 = 15
87 + 1 = 88	238 + 1 = 239	15 + 1 = 16
88 + 1 = 89	239 + 1 = 240	16 + 1 = 17
89 + 1 = 90	240 + 1 = 241	17 + 1 = 18
Index = 250	**Index = 251**	**Index = 252**
211 + 3 = 214	183 + 3 = 186	211 + 3 = 214
214 + 5 = 219	186 + 5 = 191	214 + 5 = 219
219 + 17 = 236	191 + 17 = 208	219 + 17 = 236
236 + 1 = 237	208 + 1 = 209	236 + 1 = 237
237 + 1 = 238	209 + 1 = 210	237 + 1 = 238
238 + 1 = 239	210 + 1 = 211	238 + 1 = 239
239 + 1 = 240	211 + 1 = 212	239 + 1 = 240
240 + 1 = 241	212 + 1 = 213	240 + 1 = 241

Index =253	Index = 254	Index = 255
244 + 3 = 247	211 + 3 = 214	170 + 3 = 173
247 + 5 = 252	214 + 5 = 219	173 + 5 = 178
252 + 17 = 13	219 + 17 = 236	178 + 17 = 195
13 + 1 = 14	236 + 1 = 237	195 + 1 = 196
14 + 1 = 15	237 + 1 = 238	196 + 1 = 197
15 + 1 = 16	238 + 1 = 239	197 + 1 = 198
16 + 1 = 17	239 + 1 = 240	198 + 1 = 199
17 + 1 = 18	240 + 1 = 241	199 + 1 = 200

Thus, the decrypted S-box so obtained is shown in Table 5.

Table 5. Decrypted S-Box

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

Decrypt again with the XOR method to find the final S-box that is given in Table 6.

Table 6. Final Decrypted S-Box

11000110,	00110111,	00100101,	11010100,	00000001,
11110000,	11100010,	00010011,	01001001,	10111000,
10101010,	01011011,	10001110,	01111111,	01101101,
10011100,	11011001,	00101000,	00111010,	11001011,
00011110,	11101111,	11111101,	00001100,	01010110,
10100111,	10110101,	01000100,	10010001,	01100000,
01110010,	10000011,	11111000,	00001001,	00011011,
11101010,	00111111,	11001110,	11011100,	00101101,
01110111,	10000110,	10010100,	01100101,	10110000,
01000001,	01010011,	10100010,	11100111,	00010110,
00000100,	11110101,	00100000,	11010001,	11000011,
00110010,	01101000,	10011001,	10001011,	01111010,
10101111,	01011110,	01001100,	10111101,	10111010,
01001011,	01011001,	10101000,	01111101,	10001100,
10011110,	01101111,	00110101,	11000100,	11010110,
00100111,	11110010,	00000011,	00010001,	11100000,
10100101,	01010100,	01000110,	10110111,	01100010,
10010011,	10000001,	01110000,	00101010,	11011011,
11001001,	00111000,	11101101,	00011100,	00001110,
11111111,	10000100,	01110101,	01100111,	10010110,
01000011,	10110010,	10100000,	01010001,	00001011,
11111010,	11101000,	00011001,	11001100,	00111101,
00101111,	11011110,	10011011,	01101010,	01111000,
10001001,	01011100,	10101101,	10111111,	01001110,
00010100,	11100101,	11110111,	00000110,	11010011,
00100010,	00110000,	11000001,	00111110,	11001111,
11011101,	00101100,	11111001,	00001000,	00011010,
11101011,	10110001,	01000000,	01010010,	10100011,
01110110,	10000111,	10010101,	01100100,	00100001,
11010000,	11000010,	00110011,	11100110,	00010111,
00000101,	11110100,	10101110,	01011111,	01001101,
10111100,	01101001,	10011000,	10001010,	01111011,
00000000,	11110001,	11100011,	00010010,	11000111,
00110110,	00100100,	11010101,	10001111,	01111110,
01101100,	10011101,	01001000,	10111001,	10101011,
01011010,	00011111,	11101110,	11111100,	00001101,
11011000,	00101001,	00111011,	11001010,	10010000,
01100001,	01110011,	10000010,	01010111,	10100110,
10110100,	01000101,	01000010,	10110011,	10100001,
01010000,	10000101,	01110100,	01100110,	10010111,

11001101,	00111100,	00101110,	11011111,	00001010,
11111011,	11101001,	00011000,	01011101,	10101100,
10111110,	01001111,	10011010,	01101011,	01111001,
10001000,	11010010,	00100011,	00110001,	11000000,
00010101,	11100100,	11110110,	00000111,	01111100,
10001101,	10011111,	01101110,	10111011,	01001010,
01011000,	10101001,	11110011,	00000010,	00010000,
11100001,	00110100,	11000101,	11010111,	00100110,
01100011,	10010010,	10000000,	01110001,	10100100,
01010101,	01000111,	10110110,	11101100,	00011101,
00001111,	11111110,	00101011,	11011010,	11001000,
00111001				

5 Pseudocode of the Algorithm

```
void print_bigger_matrix(int matrix[ROWS_16][COLS_16][BITS], const char *filename) {  FILE *file = fopen(filename, "w");
                                        if (file == NULL) {
                                        printf("Error opening file.\n");
                                        return; }
  fprintf(file, "[");
  for (int i = 0; i < ROWS_16; i++) {
    for (int j = 0; j < COLS_16; j++) {
      fprintf(file, "");
      for (int k = 0; k < 8; k++) {
        fprintf(file, "%d", matrix[i][j][k]); }
      fprintf(file, ", ");  }
    fprintf(file, "\n"); }
  fprintf(file, "]");
  fclose(file);}

void cartesian_product(int sets[][SET_SIZE], int result[][NUM_SETS]) {
  int index[NUM_SETS] = {0};
  int total_elements = 1;
  for (int i = 0; i < NUM_SETS; ++i) {
    total_elements *= SET_SIZE; }
  for (int i = 0; i < total_elements; ++i) {
    for (int j = 0; j < NUM_SETS; ++j) {
      result[i][j] = sets[j][index[j]];}
    for (int j = NUM_SETS - 1; j >= 0; --j) {
      if (index[j] == SET_SIZE - 1) {
        index[j] = 0;
      } else {
        index[j]++;
        break;  } } } }

void convert_to_3D(int matrix[256][8], int result[16][16][8]) {
  for (int i = 0; i < 256; i++) {
    int row = i / 16;
    int col = i % 16;
    for (int j = 0; j < 8; j++) {
      result[row][col][j] = matrix[i][j];  } } }
```

```
void xor(int mat1[8][1], int mat2[8][1], int res[8][1]){
  for (int i = 0; i < 8; i++){
    res[i][1] = mat1[i][1] ^ mat2[i][1]; } }
```

ENCRYPTION

```
printf("Enter the numbers separated by spaces:\n");
for (int i = 0; i < NUM_SETS; ++i) {
   for (int j = 0; j < SET_SIZE; ++j) {
      scanf("%d", &sets[i][j]);  } }
      cartesian_product(sets, binary_matrix);

       int A[8][8] = {
         {1, 0, 0, 0, 1, 1, 1, 1},
         {1, 1, 0, 0, 0, 1, 1, 1},
         {1, 1, 1, 0, 0, 0, 1, 1},
         {1, 1, 1, 1, 0, 0, 0, 1},
         {1, 1, 1, 1, 1, 0, 0, 0},
         {0, 1, 1, 1, 1, 1, 0, 0},
         {0, 0, 1, 1, 1, 1, 1, 0},
         {0, 0, 0, 1, 1, 1, 1, 1}
       };
       //transpose_row(binary_matrix, b_matrix);
       int B[8][1] = {
         {1},
         {1},
         {0},
         {0},
         {0},
         {1},
         {1},
         {0}
       };

FILE *encryption_debug = fopen("Encryption_debug.txt", "w");
 int decimal = 0;
 for (int index = 0; index <=255; index++){
   fprintf(encryption_debug, "index %d\n", decimal);
         decimal_to_binary(decimal, x_matrix);
         fprintf(encryption_debug, "X matrix\n");
   for (int i = 0; i < 8; i++){
     fprintf(encryption_debug, "%d ", x_matrix[i][1]);      }
     fprintf(encryption_debug, "\n");
     matrix_multiply(A, x_matrix, a_x);
     fprintf(encryption_debug, "Ax\n");
      for (int i = 0; i < 8; i++){
        fprintf(encryption_debug, "%d ", a_x[i][1]);         }
    fprintf(encryption_debug, "\n");
    xor(a_x, B, output_matrix);
     fprintf(encryption_debug, "Ax+B\n");
  for (int i = 0; i < 8; i++){
     fprintf(encryption_debug, "%d ", output_matrix[i][1]);
     resultant_matrix[index][i] = output_matrix[i][1];     }
       fprintf(encryption_debug, "\n");
       decimal++;        }
 fprintf(encryption_debug, "\n");
   fprintf(encryption_debug, "Ax+B\n");
  for (int i = 0; i < 256; i++){
```

```
    for (int j = 0; j < 8; j++){
   fprintf(encryption_debug, "%d ", resultant_matrix[i][j]);       }
         fprintf(encryption_debug, "\n");      }
 fprintf(encryption_debug, "\nS Box\n");
convert_to_3D(resultant_matrix, s_box);
 for (int i = 0; i < 16; i++){
    for (int j = 0; j < 16; j++){
     fprintf(encryption_debug, "");
       for (int k = 0; k < 8; k++){
fprintf(encryption_debug, "%d", s_box[i][j][k]);          }
          fprintf(encryption_debug, ",");          }
         fprintf(encryption_debug, "\n");       }
       print_bigger_matrix(s_box, "b_sbox.txt");
       fclose(encryption_debug);
       fflush(encryption_debug);

       //print_bigger_matrix(s_box, "b_sbox1.txt");
       for (int i = 0; i < ROWS_16; i++) {
         for (int j = 0; j < COLS_16; j++) {
           for (int k = 0; k < BITS; k++) {
             int s_0[8];
             int xor_val;
             if (i==0 && j==0){
               xor_val = s_box[i][j][k] ^ s_box[i][j+1][k];
               s_0[k] = xor_val;
               s_box_f[i][j][k] = xor_val;             }
             else if (i==15 && j==15){
               xor_val = s_0[k] ^ s_box[i][j][k];
               s_box_f[15][15][k] = xor_val;          }
             else{
               s_box_f[i][j][k] = s_box[i][j][k] ^ s_box[i][j+1][k];   } } } }
       print_bigger_matrix(s_box_f, "b_enc_sbox.txt");
       convert_to_decimal(s_box_f, s_box_d);
       printf("\nS box:\n");
       print_16x16_matrix(s_box_d);
FILE *fermat_encoding = fopen("Fermat_encoding.txt", "w");
       fprintf(fermat_encoding, "Fermat Encoding: \n");
       int fe_index = 0;
       for (int i = 0; i < ROWS_16; i++){
         for (int j = 0; j < COLS_16; j++){
           fprintf(fermat_encoding, "Index = %d\n", fe_index);
           int element = s_box_d[i][j];
           fprintf(fermat_encoding, "\n");
           for (int k = 0; k < BITS; k++){
            fprintf(fermat_encoding, "%d - %d = ", element, fer-mat_numbers[k]);
             encrypted_val = element - fermat_numbers[k];
             if (encrypted_val<0)
               encrypted_val = 256 + encrypted_val;
             fprintf(fermat_encoding, "%d\n", encrypted_val);
             element = encrypted_val; }
           fe_index++;
           fprintf(fermat_encoding, "\n");
           fprintf(fermat_encoding, "\n");
           s_box_e[i][j] = encrypted_val;      }    }
       fclose(fermat_encoding);
       fflush(fermat_encoding);
    printf("\nS box Encrypted:\n");
      print_16x16_matrix(s_box_e);
```

```
    FILE *file = fopen("encrypted_sbox.txt", "w");
      if (file == NULL) {
        perror("Error opening file");
        return 1;    }
      for (int i = 0; i < ROWS_16; ++i) {
        for (int j = 0; j < COLS_16; ++j) {
          fprintf(file, "%d ", s_box_e[i][j]);        }
        fprintf(file, "\n");        }
     fclose(file);

printf("\nS-Box stored in 'encrypted_sbox.txt' successfully.\n");
 printf("Binary initial sbox stored in 'b_sbox.txt'\n");
printf("Binary encrypted sbox stored in 'b_enc_sbox.txt'\n");
      break;
```

DECRYPTION

```
      printf("Enter the encrypted s-box filename(decimal): ");
      scanf("%s", filename);

      file = fopen(filename, "r");
      if (file == NULL) {
        perror("Error opening file");
        return 1;      }

      for (int i = 0; i < ROWS_16; ++i) {
        for (int j = 0; j < COLS_16; ++j) {
          if (fscanf(file, "%d", &s_box_e[i][j]) != 1) {
            fprintf(stderr, "Error reading from file\n");
            fclose(file);
            return 1;          }      }      }

      fclose(file);
      printf("\nEncrypted S-box read from file:\n");
      print_16x16_matrix(s_box_e);
    FILE *fermat_decoding = fopen("Fermat_decoding.txt", "w");
      fprintf(fermat_decoding, "Fermat Decoding: \n");
      int fd_index = 0;
      for (int i = 0; i < ROWS_16; i++){
        for (int j = 0; j < COLS_16; j++){
fprintf(fermat_decoding, "Index = %d\n", fd_index);
          int element = s_box_e[i][j];
          fprintf(fermat_decoding, "\n");
          for (int k = 0; k < BITS; k++){
fprintf(fermat_decoding, "%d + %d = ", element, fer-mat_numbers[k]);
 decrypted_val = element + fermat_numbers[k];
  if (decrypted_val>=256)
     decrypted_val = decrypted_val - 256;
fprintf(fermat_encoding, "%d\n", decrypted_val);
            element = decrypted_val;      }
          fd_index++;
          fprintf(fermat_encoding, "\n");
          fprintf(fermat_encoding, "\n");
          s_box_d[i][j] = decrypted_val;
          int_to_binary(decrypted_val, s_box_f[i][j]);       }       }
      fclose(fermat_decoding);
      fflush(fermat_decoding);
      printf("\nFermat decoded S-box:\n");
      print_16x16_matrix(s_box_d);
```

```
        print_bigger_matrix(s_box, “b_dec_sbox.txt”);
        int s_255[8];
        for (int i = 15; i >= 0; i--) {
          for (int j = 15; j >= 0; j--) {
            for (int k = 0; k < BITS; k++) {
              if (i==15 && j==15){
      s_box[i][j][k] = s_box_f[0][0][k] ^ s_box_f[i][j][k];
               s_255[k] = s_box[i][j][k];
              }else{
               s_255[k] = s_box[i][j][k];      } } } }
            printf(“\nDecrypted S-box:\n”);
          convert_to_decimal(s_box, d_s_box);
          print_16x16_matrix(d_s_box);
         FILE *decrypted_sbox = fopen(“decrypted_sbox.txt”, “w”);
          if (decrypted_sbox == NULL) {
            perror(“Error opening file”);
            return 1;      }
          for (int i = 0; i < ROWS_16; ++i) {
            for (int j = 0; j < COLS_16; ++j) {
          fprintf(decrypted_sbox, “%d “, d_s_box[i][j]);         }
    fprintf(decrypted_sbox, “\n”);        }
          fclose(decrypted_sbox);
          fflush(decrypted_sbox);
          print_bigger_matrix(s_box, “b_dec_sbox.txt”);
    printf(“Binary decrypted sbox stored in ‘b_dec_sbox.txt’\n”);
    printf(“Decrypted(decimal) sbox stored in ‘decrypted_sbox.txt’\n”);
```

6 Performance Analysis

The algorithm utilizes the Fermat encoding technique with the cryptographic properties of AES S-Box, and the improved AES S-Box examination results are given in Table 7 below. The number of terms enhances the AES S-Box and improves digital joint addition. Also, the enhanced AES in the S-Box Reverse Sports Joint [18] includes an infinite number of terms from the A-box AES key. It can uphold a dynamic way out of just 9 weaknesses in the AES S-Box digital hinge [19] as well as in the S-Box Affine-Power-Affine (APA) [20] logical component. The associated alteration period has been extended; the enhanced AES S-Box has an optimal implementation of comparative alteration compared to AES S-Box in addition to the S-Box (APA) [21-24].

Table 7. Comparison of S-boxes

Performance index	AES S-Box	Affine-Power-Affine	Proposed Scheme
Balance criteria	balance	balance	balance
Differential uniformity(F)	4	4	4
Non-zero linear structure	none	none	none
Number of terms in S-box algebraic expression	9	255	256
Iterative period	less than 88	less than 88	256

7 Conclusion

In this paper, AES encryption is transformed from symmetric key cryptography to asymmetric key cryptography by encrypting the S-box with the help of the XOR operation and Fermat encoding. Here, the S-box is used as a public key, with which in the AES, there will be two keys: one is the private key and another is the encrypted S-box as a public key.

References

[1] Hayat, U., Ullah, I., Azam, N.A., and Azhar, S. (2022). 'A novel image encryption scheme based on elliptic curves over finite ring.' *Entropy* 24 (5): 571.

[2] Nastou, P., Stamatiou, Y.C. (2002). 'Enhancing the security of block ciphers with the aid of parallel substitution box construction.' In *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*, 29–34. IEEE.

[3] Daemen, J., Rijmen, V. (2002). *The Design of Rijndael*. Vol. 2. Springer-verlag, New York.

[4] Lambić, D. (2018). 'S-box design method based on improved one-dimensional discrete chaotic map.' *Journal of Information and Telecommunication* 2 (2): 181-191.

[5] Afify, E.W., Khalil, A.T., El Sobky, W.I., and Abo Alez, R. (2020). 'Performance Analysis of Advanced Encryption Standard (AES) S-boxes.' *International Journal of Recent Technology and Engineering (IJRTE)* 9 (1): 2214-2218.

[6] Abdullah, S.M., Abduljaleel, I.Q. (2021). 'Speech Encryption Technique using S - box based on Multi Chaotic Maps.' *TEM Journal* 10 (3): 1429-1434.

[7] Jenny, R.S., Sudhakar, R., and Karthikpriya, M. (2021). 'Design of Compact S Box for Resource Constrained Applications.' *Journal of Physics: Conference Series*.

[8] Al-Dweik, A.Y., Hussain, I., Saleh, M., and Mustafa, M.T. (2022). 'A novel method to generate key-dependent s-boxes with identical algebraic properties.' *Journal of Information Security and Applications* 64.

[9] Sanap, S., More, V. (2022). 'Design of efficient S-box for Advanced Encryption Standard.' *J. Integr. Sci. Technol.* 10 (1): 39-43.

[10] Zahid, A.H., et al. (2021). 'Construction of Optimized Dynamic S-Boxes Based on a Cubic Modular Transform and the Sine Function.' *IEEE Access* 9: 131273-131285.

[11] Mishra, R., Singh, B., and Delhibabu, R. (2023). 'Searching for S-boxes with better Diffusion using Evolutionary Algorithm.' *Cryptology ePrint Archive*.

[12] Maolood, A.T., Farhan, A.K., El-Sobky, W.I., Zaky, H.N., Zayed, H.L., Ahmed, H.E., and Diab, T.O. (2023). 'Fast Novel Efficient S-Boxes with Expanded DNA Codes.' *Security and Communication Networks*.

[13] Hasanujjaman, Md., Goswami, P.S., Banerjee, S., and Uz Zaman, J.K.M.S. (2023). 'Secured Encryption Technique in S-Box using Fermat Encoding.' In *Intelligent Systems Design and Applications - 23rd International Conference on Intelligent Systems Design and Applications (ISDA 2023), Volume 2*, December 11-13.

[14] Goswami , P.S., Chakraborty, T., and Chattopadhyay, A. (2020). 'Knapsack encoding for secured quantum key distribution protocol.' *Modern Physics Letters A* 35 (36): 2050295 (16 pages).

[15] Webster, A.F., and Tavares, S.E. (1986). 'On the design of S-boxes.' In *Proc. Conf. Theory Appl. Crypto. Tech.*, Santa Barbara, CA, USA, pp. 523–534, Aug. 1986.

[16] Lambiá, D. (2020). 'A new discrete-space chaotic map based on the multiplication of integer numbers and its application in S-box design.' *Nonlinear Dyn.*, vol. 100, no. 1, pp. 699–711, Mar. 2020.

[17] Goswami, P.S., Chakraborty, T., and Chattopadhyay, A. (2021). 'A Secured Quantum Key Exchange Algorithm using Fermat Numbers and DNA Encoding.' In *2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Erode, India, pp. 1-8, 2021.

[18] Ibrahim, S., and Abbas, A.M. (2020). 'A novel optimization method for constructing cryptographically strong dynamic S-boxes.' *IEEE Access*, vol. 8, pp. 225004–225017, 2020.

[19] Zahid, A.H., Tawalbeh, L., Ahmad, M., Alkhayyat, A., Hassan, M.T., Manzoor, A., and Farhan, A.K. (2021). 'Efficient dynamic S-box generation using linear trigonometric transformation for security applications.' *IEEE Access*, vol. 9, pp. 98460–98475, 2021.

[20] Tsafack, N., Kengne, J., Abd-El-Atty, B., Iliyasu, A.M., Hirota, K., and Abd-EL-Latif, A.A. (2020). 'Design and implementation of a simple dynamical 4-D chaotic circuit with applications in image encryption.' *Information Science*, vol. 515, pp. 191–217, Apr. 2020.

[21] Arshad, B., and Siddiqui, N. (2020). 'Construction of highly nonlinear substitution boxes (S-boxes) based on connected regular graphs.' *Int. J. Comp. Sc. Info. Sec.*, vol. 18, no. 4, p. 6, 2020.

[22] Siddiqui, N., Yousaf, F., Murtaza, F., Haq, M.E., Ashraf, M.U., Alghamdi, A.M., and Alfakeeh, A.S.A.S. (2020). 'A highly nonlinear substitution-box (S-box) design using action of modular group on a projective line over a finite field.' *PLoS ONE*, vol. 15, no. 11, Art. no. e0241890, 2020.

[23] Alhadawi, H.S., Majid, M.A., Lambiá, D., and Ahmad, M. (2022). 'A novel method of S-box design based on discrete chaotic maps and cuckoo search algorithm.' *Multimedia Tools Appl.*, vol. 80, no. 5, pp. 7333–7350, Feb. 2022.

[24] Razaq, A., et al. (2020). 'A Novel Method for Generation of Strong Substitution-Boxes Based on Coset Graphs and Symmetric Groups.' *IEEE Access*, vol. 8, pp. 75473-75490, 2020.

Author Biographies

Md. Hasanujjaman has been a Lecturer in Computer Application at Sheikhpara ARM Polytechnic (Govt. of West Bengal) since 21.11.2013. He has been acting as the Principal-in-Charge of that Institute since 18.02.2021. He started his career as a Faculty (Full-time) at the Department of Computer Science, Balurghat College, Dakhin Dinajpur, from 2007 to 2008. Then, he joined as an Assistant Teacher in Computer Science at Raiganj Mohanbati High School (H.S.), Uttar Dinajpur, from 2008 to 2013. He received his MCA and then M.Phil. in Computer Science. He has qualified WBSET examinations and SWAYAM 8 Modules of NITTTT conducted by NTA. He has published around 04 research papers. His research areas are Advanced Encryption Standard, Cryptography, Fermat Coding, IoT, etc.

Dr. Partha Sarathi Goswami is presently working as a Lecturer in Department of Cyber Forensics and Information Security at Behala Government Polytechnic, Kolkata, West Bengal. Earlier he has served as the Head of the Department of Cyber Forensics and Information Security Department. At present he is also the Secretary of Academic Council at Behala Government Polytechnic. He has completed his Ph.D. in CSE, M.Tech. in CSE, M.Phil. in Computer Science and MCA. He has served different colleges in India as a Lecturer as well as HOD since 2004. Dr. Goswami is an Associate Member of the Institute of Engineers, India, and a Member of the International Association for Engineers, Hong Kong. He has authored numerous papers in Journals and Conferences of national and international repute. He has also authored books titled "Quantum Key Exchange with Secure Encoding" and "Cyber Crimes and Laws." His research interests include Cryptography, Bio-informatics, Quantum Computing, DNA Cryptography and Cyber Security.