Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Design and Security Evaluation of BSP-128: A Hybrid Blowfish and Serpent Inspired 128-Bit Block Cipher

Ammar Abdul Ameer[a], Qusay F. Al-Doori[b]*, Noor A Yousif[c]
[a]College of Computer Engineering, University of Technology-Iraq, Baghdad, Iraq. Email: ammar.a.rasheed@uotechnology.edu.iq
[b]College of Artificial Intelligence Engineering, University of Technology-Iraq, Baghdad, Iraq. Email: qusay.f.hasan@uotechnology.edu.iq
[c]College of Artificial Intelligence Engineering, University of Technology-Iraq, Baghdad, Iraq. Email: noor.a.yousif@uotechnology.edu.iq

ARTICLE INFO

Article history:
Received: 03/03/2026
Rrevised form: 16 /03/2026
Accepted : 17 /03/2026
Available online: 30 /06/2026

Keywords:

Cryptography,

Blowfish,

Serpent,

Block cipher,

Feistel network.

ABSTRACT

In this work, a new 128-bit block cipher (namely, Blowfish–Serpent Protocol 128-bit (BSP-128)) has been introduced that allows to improve the performance and implementation of an optimal compromise between security, performance, and implementation efficiency. The key size for the input and output for BSP-128 is also scalable at 128-bit blocks and keys with a length of up to 256 bytes, and the design conforms to the underlying scheme for AES. The BSP-128 algorithm uses a 16-round Feistel network similar to Blowfish and Serpent with multiple architectural improvements that aid the diffusion, so this cipher is much more robust against modern cryptanalysis techniques. The primary goal of BSP-128's design is to be resistant to both differential and linear cryptanalysis. Finally, the round structure was made even more powerful to better handle identified potential security vulnerabilities and targeted attack scenarios discovered in a subsequent internal analysis. By mixing substitution and permutation in the structure of the F–function, BSP-128 transforms from standard SP to a Substitution–Permutation–Substitution (SPS) type of model which boosts the avalanche effect. And we use in-place integer multiplication as a diffusion intermediate for additional mixing details to each round and to maintain the stable quality of the speed as well as security. While Blowfish's fixed-factor algorithm runs on 64-bit blocks and takes up a lot of memory for large lookup tables, BSP-128 automatically supports 128-bit blocks to avoid matching ciphertext attacks and achieve low memory consumption. This attribute of it adds a large value to its ability to work in limited hardware settings such as smart cards. The key expansion mechanism yields 786 bytes of subkeys, which are made from a P-array having 32 subkeys of 64 bits each, and 32 S-boxes with 64 fields per box. It ensures a strong key-dependent transformation and protects against brute-force attacks, thanks to this long key length. While the performance of BSP-128 is highly promising with regard to core security metrics, experimental findings show that BSP-128 surpasses expectations through computational efficiency. The final design will be submitted for public survey and comparison in academia, resulting in a secure and high throughput candidate tailored for contemporary cryptographic uses.

MSC..

https://doi.org/10.29304/jqcsm.2026.18.22706

∗Corresponding author : *Qusay F. Al-Doori*

Email addresses: *qusay.f.hasan@uotechnology.edu.iq*

Communicated by ‘sub etitor’

1. Introduction

Cryptography is the primary line of defense against unauthorized access to data. There are two specific styles of cryptographic algorithms – symmetric and asymmetric – both of which respond to general requirements for security, namely: confidentiality, availability, authentication, and integrity. Asymmetric algorithms are also known as public key cryptography; they rely on pairwise encryption using a public key (for encryption purpose), and a private key for decryption. However, symmetric algorithms use only one private key for all encryption as well as decryption. Symmetric algorithms are usually less expensive than asymmetric algorithms. Typically, asymmetric mechanisms are used to send through the secret sharing key required for symmetric encryption before the data itself is encrypted. There are many symmetric algorithms such as Blowfish, DES, 3DES, AES, Twofish, RC2, RC5, CAST-128, and Serpent [1].

Nevertheless, among other encrypted types, symmetric key cryptography is the most powerful and security-preserving encryption method, suitable for live computations. Blowfish is a well-known symmetric block cipher developed by Bruce Schneier in 1993 and has established a reputation for its strong encryption and computational efficiency [2].

Blowfish runs on a 64-bit block size and supports key sizes from 32 to 448 bits as necessary. It is also known for having a great trade-off between security and performance. Unlike most symmetric encryption algorithms, such as the Data Encryption Standard (DES), which is classified as insecure because of its small key size, Blowfish provides much more protection against brute-force attacks [3]. In terms of secure operation, 16 rounds of encryption in a Feistel network structure give a reliable and fast run time. The expansion phase of Blowfish is a very computation-heavy process but is essential for its robustness to cryptanalysis [4]. Blowfish can be applied widely on-scale as one of the most used software tools for lightweight applications. Therefore, these applications can be based on low-resource applications such as embedded platforms, Internet of Things (IoT) devices and secure cloud computing [5]. Since recent encryption techniques based on modern cryptography standards such as Advanced Encryption Standard (AES) need lots of computing power, unlike such modern standards as AES, Blowfish can achieve an excellent trade-off between performance and security (and speed) which makes Blowfish the best choice for applications where performance is paramount. Due to its ease and public domain available nature, it is also used in security features, such as Secure Shell (SSH), VPNs, and password protection [6]. Blowfish has certain potential drawbacks, however, which restrict its broad deployment in some deployments. The block size of 64 bits was sufficient for past iterations, but now is dwarfed by modern cryptographic security protocols [7], particularly when utilizing vast amounts of data. This limits it from long-distance communication which potentially makes it susceptible to birthday attacks. Blowfish is also computationally resource intensive in the early phase of key setting, involving a heavy computational burden for multiple key modifications due to the system.

One of the block cipher algorithms (Serpent) that does this manipulating data using a table (Initial permutation (IP)) and then breaking the words 128 bits into four (32) bits for entry into a table (Substitution-box). When the algorithm got entered, the next stage of data processing passes it along to a function (Ip) that changes the order of the data which then enters with round positions that can be as long as 31 and shared with the 32 keys issued by the key generation function [8]. While the Serpent algorithm is very secure, it has some limitations in memory usage and execution speed. The Serpent process has 32 rounds which directly influence results [9]. Image encryption techniques have also been established for enhancing Serpent [10-15]. The BSP-128 algorithm modifies Blowfish and Serpent algorithms aligning them with AES standards, while removing weaknesses of Blowfish and Serpent algorithms with a set of several changes:

- It bolsters the security of the original Blowfish algorithm that is built on the legacy one by implementing a 128-bit block size instead of a 64-bit block size.

- The time involved in subkey generation for the Blowfish algorithm increases the computational complexity, making brute-force attacks much harder to mitigate. We also need 48 iterations to evaluate a single key. To overcome this limitation, BSP-128 uses the same key generation method as Blowfish.

- BSP-128 also makes the algorithms richer by integrating combinations of basic operations (such as addition, XOR (exclusive or), and multiplication).

- By using the IP and FP in the beginning and the end of algorithm this will make the algorithm more secure. Where these permutations are adept from serpent algorithm.

- The mapping of S-box is 4-to-4 in BSP-128 algorithm like Serpent and this make the S-box is small while the mapping which is used in the Blowfish algorithm is 4-to-32 and this make the S-box is large.

- Both Serpent and BSP-128 have a 512-byte S-box, whereas Blowfish's S-box has 4096 bytes. Implementing the algorithm on smart cards is feasible; however, small processors face significant challenges due to stringent RAM and ROM constraints. Additionally, optimizing efficiency is crucial for these compact devices.

Although symmetric block ciphers such as Blowfish and Serpent are widely used, they still have several limitations. Blowfish suffers from a small 64-bit block size and high computational cost during key expansion, while Serpent provides strong security but requires more execution time and memory due to its 32-round structure. Therefore, there is a need for a cryptographic algorithm that improves security while maintaining better performance and lower memory requirements. To address these issues, this work proposes the BSP-128 block cipher, which combines features of Blowfish and Serpent and introduces architectural enhancements to improve efficiency and resistance to cryptanalytic attacks.

The structure of this paper is presented as follows: Section 2 covers the related works in Section 3 illustrates the proposed framework. Experimental results are shown in Section 4. Finally, Section 5 overviews the paper and proposes future research directions.Here introduce the paper, and put a nomenclature if necessary, in a box with the same font size as the rest of the paper. The paragraphs continue from here and are only separated by headings, subheadings, images and formulae. The section headings are arranged by numbers, bold and 11 pt. Here follows further instructions for authors.

1. Related Works

Recently, many techniques have been proposed by scientists to improve Serpent's performance and security. Here we review changes recommended and how they have been evaluated. In 2016, Patel & et al [1] proposed a method for modification of the cumulative Blowfish rounds that allows you to skip some rounds on the basis of the round key. Moreover, the scheme can increase the security level of the Blowfish cipher as it will be less affected by brute force attacks, if it is in general from small in-factor key size to significant as in large-interactive key. In addition to that, the method also obviates the time needed for encrypting and decrypting the data with the Blowfish cipher. However, skipping rounds may potentially weaken the overall cryptographic strength if not carefully designed. In 2020 Zagi et al.,[17] improved on and mitigated the privacy aspects by utilizing the uniform structure of the standard algorithm. They did so by inventing a new generation method of keys, since the strength and uniqueness of a key are important because the block cipher is highly secured using a unique key. In this case, where there are different functions with (Gost external structure) as well as with of (Shift <<<), (AES -Key Schedule), (MD5). 1n 2021, Elshoush et al. Nevertheless, the increased complexity of the key generation process may lead to higher computational overhead. [18] proposed a framework to increase the security and performance of the Serpent algorithm. The basic idea behind this approach is to create a sub key for each block using Lorenz 96 chaos and encrypt and decrypt the blocked blocks in ECB parallel mode via the two layers. Adeniyi et al., [19] proposed a modified Blowfish algorithm, which has been developed by modifying the structure(s) of the F function to cipher and decrypt video data in 2022. Then, the performance of the ordinary and modified Blowfish algorithm will be measured in time complexity while the avalanche effect will be calculated. However, the use of chaotic systems and additional layers increased the computational cost. In 2022, The Ashwaq et al., [20] proposed system used block-based image encryption technique with chaotic map property. The digital image is first divided into randomly generated blocks. Algorithm-wise, these blocks are subsequently subjected to an improved blowfish. The enhancements on design allow to harness this strong facility and after correcting the weaknesses, the attack structure is protected by the blowfish algorithm which yields improved performance and security. Therefore, with the same system we have a workable solution for encrypting images. However, the use of chaotic systems and additional layers increased the computational cost. In 2023, Souror et al., [21] proposed a Hybrid-Blowfish algorithm based on the combination between blowfish and counter-measure of SCA using Masking string and comparing DES, 3DES, AES, blowfish, RSA, Hybrid-Blowfish in the tests like frequency test, serial test, encryption time, decryption time, memory, avalanche effect and

entropy and no of bits required for encoding optimally and the results demonstrate the effectiveness and superiority of Hybrid-Blowfish compared to traditional Blowfish and increase the cyber-security strength by approximately 25%. While the method improved performance, reducing the number of rounds may affect the overall robustness of the algorithm. However, the approach is mainly specialized for image data and may not generalize well to other data types. In 2024, Ali et al., [22] applied one level of Integer Discrete Wavelet Transform (IWT) to divide the scrambled region into four sub-bands. Then, a Feistel network based on polynomial-based secret image sharing (SIS) encrypts the lowest frequency band only while the three bands LH, HL, and HH are diffused using a mapping technique based on the Morton scan to swap coefficients positions and then confused based on the hyperchaotic system. The culmination of these techniques results in generating a test image cipher characterized by robust confusion and diffusion properties. Importantly, this methodology has yielded remarkable results, reducing the encryption time by up to 96%. In 2025, Singhal et al., [23] reported on an important study that safe key transmission and a secure cryptographic algorithm are both needed for effective data security. It compares and improves the Blowfish algorithm with DES and DCT; analysis of differences in complexity, security and its efficiency. The study applies its structure to Blowfish, which involves decreasing number of rounds and increasing block length through a transformation model, and is intended to increase encryption efficiency and protect data from plaintexts and files more effectively. In 2025, et al., [24] showed that the Elliptic Curve Cryptography (ECC) provides strong security with shorter key lengths, making it suitable for resource-constrained environments such as IoT and mobile devices. Blowfish is a fast symmetric block cipher commonly used for encrypting legacy and stored data. The authors suggest that combining ECC and Blowfish can leverage both strong security and high encryption speed.

2. Proposed Framework

The BSP-128 is a symmetric-key block cipher with 128 bits of block size and variable key sizes (64 to 256 bytes). The BSP-128 cipher implements a mixture of operations, offering it high performance, security, and flexibility. The key theme underwritten by BSP-128 design is to obtain the best security/performance balance using the best available methodologies today for designing block cipher.

BSP-128 is an enhancement of the Blowfish and Serpent algorithms developed to comply with the Advanced Encryption Standard. BSP-128, like Blowfish, utilizes Feistel network as an essential element. Some of the new features in BSP-128 include the use of a new F-function, which is a substitution permutation substitution (SPS) network instead of a substitution permutation (SP) network. The SPS network will enhance the avalanche effect. And integer multiplication is also used as a primitive operation in BSP-128 algorithm. The adoption of multiplication greatly increases diffusion per round, which delivers more security, fewer rounds and increased throughput.

BSP-128 was designed after considering Blowfish and Serpent as potential candidates to submit an AES. Subsequently, changes were made to satisfy AES needs, to enhance security, and to elevate performance. The outer loop, though, is based around the same found in Blowfish. BSP-128 will be deliberately simplified to allow for analysis and the insight of the security offered by heavy reliance on key-dependent S-boxes, such as Blowfish. To meet the requirements of the AES, a block cipher must handle 128-bit input/output blocks. While Blowfish is an exceptionally fast block cipher, extending it to act on 128-bit blocks in the most natural manner would result in using two 64-bit working registers.

The Blowfish philosophy is simply to employ techniques that work well with modern processors. BSP-128 builds upon one of the best-known examples of this, the Feistel network which is now efficiently implemented and easy to understand. Integer multiplication is a very effective diffusion primitive and is used in BSP-128 as a result the new BSP-128 has much faster diffusion compared to Blowfish. It also helps BSP-128 to operate with fewer rounds in a secured manner and with an increased throughput.

The Serpent algorithm is in the list candidate and one thing is true the best Serpent software is going to be 3-4 times slower compared to a lot of other alternatives (Mars, Twofish, RC6, and Rijndael). Therefore, even a non-conservative Serpent, with just sixteen rounds will be much slower than these other ones. By

fewer rounds and using other techniques instead of linear transformation theBSP-128 improved the rate without undermining security.

The BSP-128 is well suited to meet all of the requirements of the Advanced Encryption Standard. BSP-128 consists of two parts: a key-expansion part and a data-encryption part. Key expansion converts a variable-length key of at most 256 bytes into several subkey arrays totaling 786 bytes.BSP-128 has 16 rounds; each round consists of a key-dependent permutation and a key- and data-dependent substitution.

2.1. Subkeys:

The speed of most AES candidates, irrespective of key length (encryption and key setup time), is independent. I.e., no matter what length of key bits is involved, the time taken to set up a key and to encrypt a block of text does not vary. BSP-128 uses a large number of subkeys. Before any data encryption or decryption, these keys must be precomputed. The P-array consists of 32 64-bit subkeys:

P1, P2,..., P32.

There are also 32 S-boxes with 64 entries each:

S1,0, S1,1,..., S1,64;

S2,0, S2,1,..,, S2,64;

.

.

.

S31,0, S3,1,..., S31,64;

2.2. Data Encryption

Most free software developers do not want to accept any restrictions on algorithm use. The Blowfish algorithm has a large lookup table so doubling it makes the performance not free in software and hardware. BSP-128 doubles Blowfish to 128-bits but uses another F-function to overcome Blowfish restrictions. The philosophy behind BSP-128 is that simplicity of design yields an algorithm that is easier to implement. Through the use of a streamlined Feistel network and a simple S-box substitution and a simple P-box substitution. The Feistel network makes up the body of the BSP-128 and is designed to be as simple as possible, while still retaining the desirable cryptographic properties of the structure. Figure (1) illustrates the architecture of the BSP-128 algorithm with 16 rounds. The input is a 128-bit data element, P, and the output is the ciphertext C.

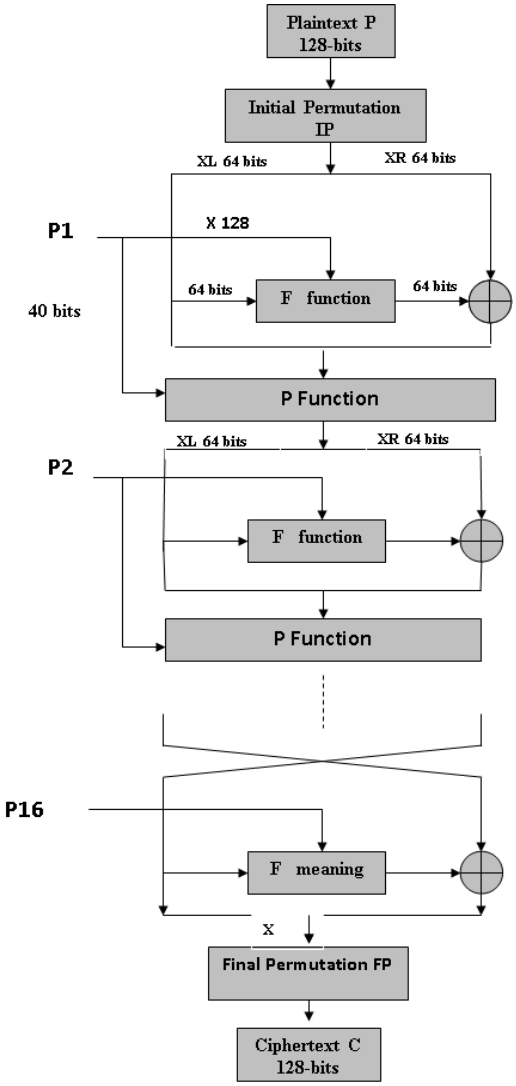

Fig. 1- The block diagram of BSP-128

A. Initial and Final Permutation:

The first step in the algorithm is the initial permutation IP. It's similar to Serpent being applied before the first. The input block needs to be given enough diffusion and confusion. As with Serpent, the final permutation FP is the inverse of the initial permutation. The initial and final permutations were chosen in order to simplify an optimized implementation of the cipher, thus enhancing both its computational power and efficiency. These two permutations are reported in the appendix. Blowfish did not have reversible mixing function before and after the last round. It would just add confusion once again to the entry values in the Feistel network, causing an avalanche affect to be complete after the first two rounds.

BSP-128 is a Feistel network consisting of 16 iterations of (P-function and F-function) Thus the cipher may be formally described by the following algorithm:

Algorithm 1: BSP-128 Encryption Procedure

Input:
128-bit plaintext block P

Output:
128-bit ciphertext block C

Procedure:

1. Apply the Initial Permutation (IP) to the plaintext block:
   $X \leftarrow IP(P)$
2. Split Xinto two 64-bit halves:
   $X_L, X_R \leftarrow \text{Split}(X)$
3. For round $i = 1$to 16do
   3.1 Compute the round function and XOR operation:
   $X_R \leftarrow F(X_L) \oplus X_R$ where $F$is the round function
   3.2 Apply permutation and key mixing:
   $A \leftarrow P(X_L, X_R, KP1_i, KP2_i)$ where $KP1_i$and $KP2_i$are 5-bit subkeys generated for round i.
   3.3 Split the result:
   $X_L, X_R \leftarrow \text{Split}(A)$
   End For
4. Concatenate the final halves:
   $X \leftarrow X_L \parallel X_R$ where ‖denotes concatenation.
5. Apply the Final Permutation (FP):
   $C \leftarrow FP(X)$
6. Return ciphertext C

The Blowfish algorithm employed XOR as a reversible mixing function before the first and after the last round. Meanwhile the BSP-128 algorithm employed IP and FP. Also applied multiplication before S-Box substitution since the reversible mixing function is more complicated than XOR and this would further confuse the entry values into the Feistel network and ensure a complete avalanche effect after the first two rounds.

Decryption is precisely the same as encryption, but P1, P2,……., P32 are used in reverse order. The encryption and decryption algorithms have the same complexity. BSP-128 is designed to take advantage of the powerful operations supported in today's computers, resulting in a much improved security/performance tradeoff over existing ciphers. As a result, BSP-128 offers better security than Blowfish while running significantly faster than Serpent.

B. The P Function:

In BSP-128 algorithm key dependent initial permutation is used as a reversible mixing function (which is a more complicated reversible mixing function) prior to F function. This is used to overcome a limitation of the Feistel structure because each round of transformation always retains one half of the block and the P-function will further confuse the entry values into the Feistel network and guarantee a full avalanche effect after the first two rounds. P-function has 128-bit input A and 128-bit output D, which takes into account "byte transposition" and the 40-bit subkey (KP1|KP2), to keep data rotated.

Let $Kp_1=(m_1, m_2, m_3, m_4)$, and $Kp_2=(n_1, n_2, n_3, n_4)$, where m_j and n_j is a 5-bit subkey and not equal to zero, j= 1,..., 4. The function $D = P(A, KP_1 \mid \mathrm{KP}_2)$ is defined by following:

- Right rotation: $b_j=a_j>>>m_j$, for j=1,...,4.
- Byte transposition: $C_{ji}=b_{ij}$, for j, l=1,...,4.
- Left rotation: $d_j=c_j>>>n_j$, for j=1,...,4.

Hence, every input word a_i affects all output words. The output words on the other hands will change the input words. In the P-function, a variety of permutations are produced through the processes of rotation and transposition.

The analysis of the output shows the P-function is a key-dependent reversible permutation applied to the 128-bit input block before the F-function. Its purpose is to maximize diffusion and ensure the avalanche effect early in the Feistel network. The P-function has the following properties:

1. Reversibility

$\mathrm{ROL}^{-1} \circ T^{-1} \circ \mathrm{ROR}^{-1}$ (1)

2. Full Mixing / Diffusion

$\forall j, \forall k, \exists i, l: A_i[l]$ affects $D_j[k]$ (2)

3. Avalanche Effect

A single-bit change in A propagates to all D_j within one P-function application.

4. Key-Dependent Confusion

Rotation amounts m_j vary per round, enhancing resistance against structural attacks.

Full Equation for P-function

$$f_P(A, M) = (\mathrm{ROL}(T(\mathrm{ROR}(A_1, m_1), \dots, \mathrm{ROR}(A_4, m_4)), m_j))_{j=1}^{4} \quad (3)$$

P-function defines a lightweight, high-diffusion, reversible permutation layer suitable for early avalanche in the Feistel structure of BSP-128.

C. The Function F

The non-reversible function is designed to be durable, effective, and user-friendly. The procedure merging the two S-box outputs is designed for the highest performance. Though an operation that simply XORs the four values could be used, the addition modulo 223 coupled with XOR is good enough to connect two very different types of algebra, without needing additional instructions. Function F is the essential part by which we can ensure the algorithm's security, so a more complex reversible function is indicated.

In Figure (2), F-function has 64-bit input Ri, 128-bit input subkey Ki and 64-bit output Z. It adopts SPSN (Substitution-Permutation-Substitution network) structure. S-layer 1 and S-layer 2 consist of 16 parallel S-boxes with 4-bit input and output like. This mapping like Serpent while the mapping in Blowfish is 8-bit input and 32-bit output. The size of S-boxes of BSP-128 is 512 bytes, which is the same as Serpent. Each S-layer uses two distinct S-box to substitute data and to realize the nonlinearity. The total number of S-boxes will be used in BSP-128 is 64 S-boxes, each of which is 64 bits. But the S-boxes size of Blowfish is 4096 bytes. The issue of P-layer is to construct the linear layer such that the number of active S-boxes in two consecutive rounds may became as large as possible.

When using two words to multiply, the lower bits of the input word affect the resulting product much more than the higher bits do. In order to overcome this problem, we apply "byte permutation" to rearrange the bits such that they have different positions and contribute equally to the product. Historically, multiplication was not possible for fast encryption in many cases; older machines needed many more process cycles for any one multiplication operation. But all today's architectures, including PowerPC, Pentium-Pro, Alpha, Ultra-SPARC, and other recent ones, have a multiply instruction that can be run in approximately two cycles. The role of multiplication in the F-function concerns our capabilities to perform these operations: indeed, apparently the multiplication of two data words makes the analysis rather difficult. The Algorithm of F-function of BSP-128 is as follows:

Algorithm 2: F-Function of BSP-128

Input:

R // 64-bit data block

K_i, where $i = 0,1, \dots ,15$ //128-bit round subkey

Output:

$Z = F(R, K_i)$ // 64-bit output block Z

Procedure:

1. Split the input bock

   Divide R into two 32-bit words:

   $$R_0, R_1 \leftarrow \text{Split}(R)$$

2. Split the round subkey

   Divide the 128-bit subkey $K_i$into four 32-bit words:

   $$K_{i0}, K_{i1}, K_{i2}, K_{i3} \leftarrow \text{Split}(K_i)$$

3. Initial key mixing

   $$X_0 \leftarrow R_0 \oplus K_{i0}$$
   $$X_1 \leftarrow R_1 \oplus K_{i1}$$

4. Substitution using S-boxes

   $a_0 \leftarrow S_{i0}(X_0)$ where S_{ij}represents the S-box used in round i.

   $a_1 \leftarrow S_{i1}(X_1)$

5. Permutation stage

   $$a_0 \rightarrow (a_{00}, a_{01}, a_{02}, a_{03})$$
   $$a_1 \rightarrow (a_{10}, a_{11}, a_{12}, a_{13})$$

   Apply permutation to obtain four 32-bit words:

   $b_0, b_1, b_2, b_3 \leftarrow P(a_0, a_1)$ where Prepresents the permutation function.

6. Linear transformation

   $Y_0 \leftarrow b_0 + b_1$ where +denotes addition modulo 2^{32}.

   $Y_1 \leftarrow b_2 \oplus b_3$

   (Addition is performed modulo 2^{32})

7. Second key mixing

   $$X_2 \leftarrow Y_0 \oplus K_{i2}$$
   $$X_3 \leftarrow Y_1 \oplus K_{i3}$$

8. Second substitution layer

   $$a_2 \leftarrow S_{i2}(X_2)$$
   $$a_3 \leftarrow S_{i3}(X_3)$$

9. Output generation

   Concatenate the two 32-bit words:

$Z \leftarrow a_2 \parallel a_3$

10. Return Z

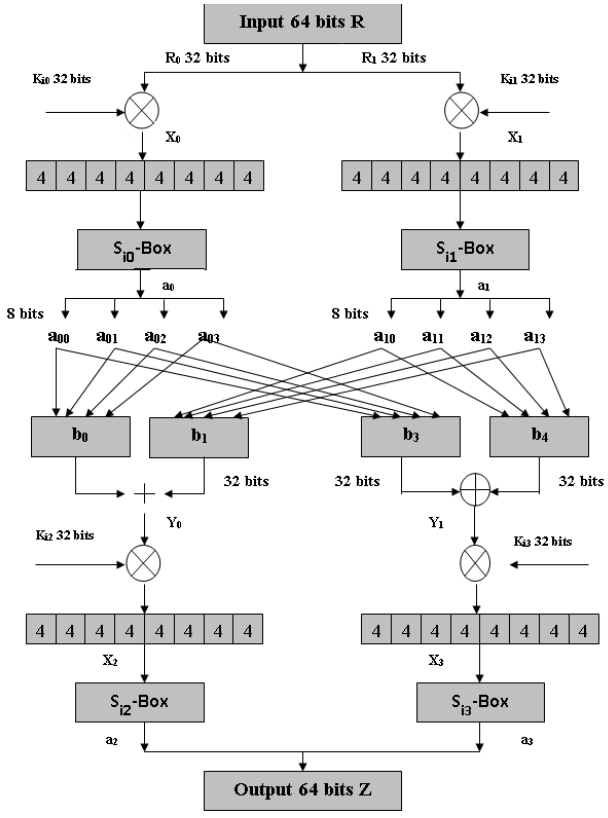

Fig. 2- The block diagram of the proposed F function

A. Formal Security Analysis of BSP-128 F-Function

The F-function of BSP-128 is a mapping:

$F: \{0,1\}^{64} \times \{0,1\}^{128} \rightarrow \{0,1\}^{64}$ (4)

with structure:

$F = S_2 \circ K_2 \circ L \circ P \circ S_1 \circ K_1$ (5)

where:

$K_1, K_2$are key mixing operations (XOR or modular addition)

$S_1, S_2$are substitution layers (S-boxes)

$P$is a permutation layer

$L$is a linear or modular addition layer

1. Algebraic Degree

Let $\deg(f)$denote the algebraic degree of a Boolean function f. Then for one F-function:

$\deg(F) \leq \deg(S_2) \cdot \deg(L \circ P \circ S_1 \circ K_1)$ (6)

with growth through the layers:

$$\deg(X \oplus Y) = \max(\deg(X), \deg(Y)) \quad (7)$$
$$\deg(X + Y \bmod 2^n) \leq \deg(X) + \deg(Y) + 1 \quad (8)$$

The F-function degree saturates near 64 bits within a few rounds of the Feistel network:

$$\deg(F_r) \to 64 \text{for small } r \quad (9)$$

2. Differential Cryptanalysis

Let δ_S be the maximum differential probability of an S-box. Let $N_{\text{active}}(r)$be the minimum number of active S-boxes in r rounds. Then the F-function satisfies:

$$P_{\text{diff}}(F_r) \leq (\delta_S)^{N_{\text{active}}(r)} \quad (10)$$

For full 16-round BSP-128:

$$P_{\text{diff}}(16) \ll 2^{-128} \quad (11)$$

3. Linear Cryptanalysis

Let $\epsilon_S$denote the maximum linear bias of an S-box. The total linear bias over r rounds satisfy:

$$\text{Bias}(F_r) \leq (\epsilon_S)^{N_{\text{active}}(r)} \quad (12)$$

Full-round bias is negligible:

$$\text{Bias}(F_{16}) \approx 0$$

4. Higher-Order and Integral Resistance

Because algebraic degree grows rapidly:

$$\deg(F_r) \geq d_{\text{lower}} \text{and } d_{\text{lower}} \to 64 \text{ quickly} \quad (13)$$

This prevents higher-order differential and integral attacks.

5. Security Margin and Structural Proofs

Reduced-round analysis shows:

$$r_{\text{best attack}} < 16$$

and structural analysis confirms there are no fixed points, no slide attacks and no weak keys exist.

6. Automated Verification

Formal bounds can be validated using MILP, SAT/SMT solvers, or Gröbner basis computations to verify:

$$\deg(F_r), P_{\text{diff}}(F_r), \text{Bias}(F_r) \quad (14)$$

D. Key generation:

The BSP-128 algorithm supports a key size of various sizes from 8 bytes to 256 bytes. The BSP-128 algorithm provides subkeys, analogous to the Blowfish algorithm. Hence, the complexity of key generation depends on the BSP-128 algorithm itself. With 16 rounds, this algorithm runs efficiently. The number is critical for determining the size of the P-array and hence the subkey generation process; with 16 iterations, it also allows keys to be of 2048 bits long. To make sure that a key is long enough to satisfy certain security requirements, two basic methods may be

implemented. First, it carefully plans the algorithm to preserve the entire entropy of the key, such that brute force attacks to reach the cryptanalysis phase becomes the only way out. The second one is to create the algorithm with lots of key bits and remove as many bits of attack for the algorithm as possible to decrease the effective key length. Because Blowfish is optimized for a robust microchip with large memory size, one goal of this system is to reduce the required memory.

The range of potential values that a key can assume has increased dramatically. It's crucial to have a huge key space to deter exhaustive key searches, in which all possible key values are tested until the correct one is identified. The solution here keeps the traditional key generation as applied by Blowfish since one also tries to keep the main entropy of the key so that there is uniform key distribution among subkeys. It also is designed to randomly allocate the allowed subkeys from the total set of candidate subkeys. P-array and S-boxes have to be calculated beforehand before any data encryption or decryption is performed. The effectiveness of several FN architectures, particularly their robustness to Differential Cryptanalysis and Linear Cryptanalysis, is directly determined by the design of FN's S-boxes.

Depending on the key used, S-boxes can be static (for all keys) or dynamic. In general, ciphers based on key-dependent S-boxes are more secure than those based on fixed S-boxes. Of the most important key-dependent S-boxes, there are 2 predominant approaches. Certain ciphers, such as KHUFU and WAKE, deliberately specify the S-box so that no two entries are identical, while others rely on an arbitrary generation process without any such guarantees as in REDOC II and Blowfish. Usually, almost all-important S-boxes are generated in some way that is completely detached from how the cipher operates. For example, Blowfish produces its S-boxes by running its own version of itself several times, and the result closely resembles an arbitrary S-box, but the performance overhead to set up the keys is much higher. While fixed S-boxes must be designed to resist differential and linear cryptanalysis effectively, key-dependent S-boxes show considerably increased resistance to these forms of attack. The BSP-128 uses key-dependent S-boxes that resemble those in Blowfish. It is more efficient in this way because Serpent's structures and mappings are employed to reduce the size of the S-boxes. Linear and differential cryptanalysis can only work if the analyst knows how the S-boxes are structured. When such boxes are selected via security-based cryptographic technique and are tied to the key, it becomes much more difficult to achieve such analyses. Because the configuration of these S-boxes is hidden from potential adversaries, their structure is much harder to abuse. In addition, these types of S-boxes make it easy to implement and they can be created as needed, reducing reliance on the large data structures of the algorithm.

5. Experimental results

The block size of 64-bits makes Blowfish algorithm vulnerable to the matching ciphertext attack. Where after encryption of 2^{32} blocks, equal ciphertexts can be expected and information is leaked about plaintext. So that, the BSP-128 algorithm with 128-bits block size is resistant to matching ciphertext attacks. It is required to 2^{64} ciphertext.

4.1. Dictionary Attacks:

As the block size is set at 128 bits, an attacker would need a dictionary containing 2128 distinct plaintexts to successfully encrypt or decrypt any arbitrary message without knowledge of the key. This type of attack is applicable to all deterministic block ciphers that utilize 128-bit blocks, irrespective of their design specifics.

4.2. Key-Collision Attacks:

For a key size of k, key collision attacks allow the forging of messages having only a complexity of 2(k/2). As a result, the operation to forge messages using 64-bit keys is about 232; however, it is escalated to around 264 with 128-bit keys. Such attacks are applicable to any deterministic block cipher and depend only on key size, regardless of the characteristics of the cipher itself.

4.3. Avalanche Effect:

In this part of the study these statistical analyses will take place on the ciphertext coming from the encrypted plaintext. The cipher system consists of the operation to transform plaintext into ciphertext, often called a cryptogram. Encryption is the process of changing data into an unreadable format, that can only be decrypted by someone who knows, for reasons of confidentiality and obfuscation, what the secret key is. The most relevant reason for this is that it preserves privacy, and keeps your information private and from eyes not specifically intended for it, even though people do have access to the encrypted content. For instance, the data stored in a hard disk can be encrypted to make sure that nobody doesn't get at it. Decryption is the reversal of encryption; it is the return of ciphertext to plaintext. This process utilizes a key that is 32 bytes long, specifically "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa."

Horst Feistel [25] explained the avalanche effect; "a small change in the key corresponds to a huge change in the ciphertext." We can clearly see from Table 4.1 the avalanche effect on plaintext where it can be illustrated by modifying only one bit of the key using BSP-128. The avalanche effect on ciphertext from a single-bit change during the BSP-128 running was presented in Figure 4.1, demonstrating that the average avalanche effect is 65.5%.

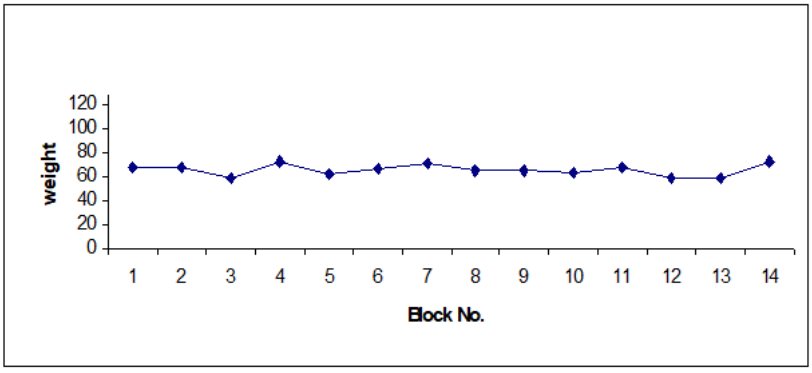

Fig. 3- Impact of altering a single bit on the ciphertext utilizing the BSP-128 algorithm.

Figure 3 illustrates the impact of the avalanche effect on ciphertext resulting from a single bit modification during the execution of the Serpent algorithm. This figure indicates that the average magnitude of the avalanche effect is 63.7%.

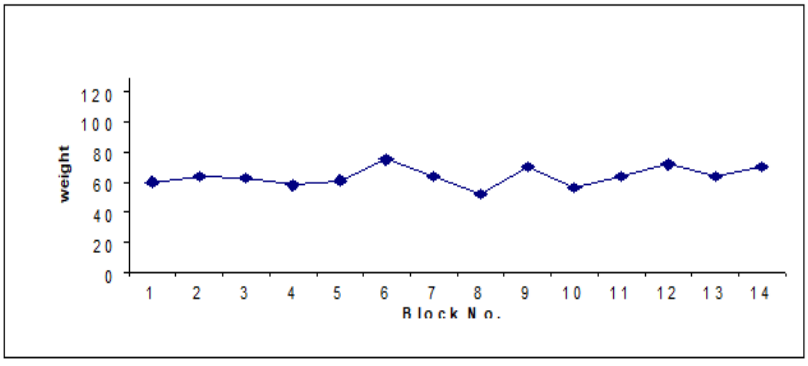

Fig. 4- Avalanche Effect on Ciphertext Resulting from a Single Bit Change in the Serpent Algorithm

Figure 4 illustrates the impact of the avalanche effect on ciphertext resulting from a single bit alteration during the execution of the Blowfish algorithm. The data indicates that the average magnitude of this avalanche effect is 31.4%.

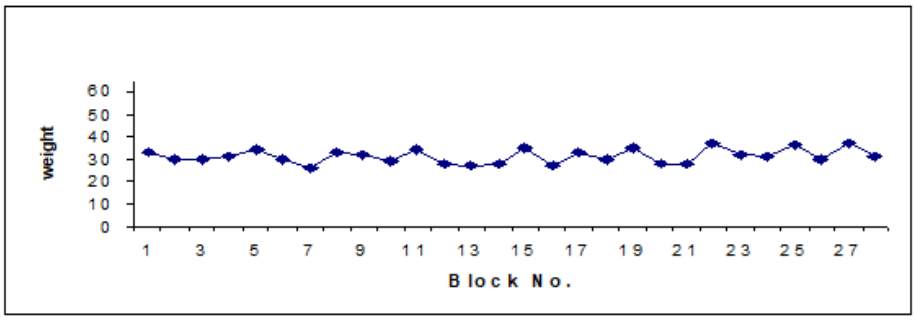

Fig. 5- Impact of a Single Bit Change on Ciphertext in the Blowfish Algorithm

4.4. Algebraic Degree Growth per Round (BSP-128)

The algebraic degree of a cipher is the maximum degree of its output bits expressed as Boolean polynomials of:

- plaintext bits
- key bits

Higher degree per round generally implies stronger resistance to:

- Higher-order differential attacks
- Integral attacks
- Algebraic attacks

For a 128-bit block cipher, we want fast degree growth approaching the theoretical maximum (≤128).

The Algebraic Degree Growth per Round (BSP-128)

1. XOR and permutation do not increase algebraic degree.
2. Modular addition increases degree slightly (due to carry bits).
3. The S-box layers significantly increase nonlinearity.
4. Integer multiplication provides strong degree growth and rapid diffusion.

Inside one F-function, the degree grows quickly due to two substitution layers and multiplication.

Across Feistel rounds, the degree roughly doubles every few rounds. With the given structure, the algebraic degree likely reaches near its maximum (128) within about 5–8 rounds.

Therefore, with 16 rounds, BSP-128 likely achieves full algebraic degree early and maintains a reasonable security margin — assuming the S-boxes and multiplication are well designed.

4.5. Randomness Test

NIST [26] Special Publication 800-22 Revision 1a is the authoritative document from the U.S. National Institute of Standards and Technology (NIST) that defines a suite of statistical tests for evaluating the randomness of binary sequences produced by random or pseudorandom number generators.

The suite consists of 15 statistical tests designed to detect different types of non-random patterns in a binary sequence (e.g., uniform frequency, runs, patterns, longest run of 1s, etc.). Table (1) shows NIST test results for text and key show in section 4.3.

Table 1. NIST test for BSP-128

No.	Test Name	Purpose	Expected P-value Range	P-value
1	Frequency (Monobit)	Balance of 0s and 1s	P ≥ 0.01	0.478

2	Frequency within Block	Local 0/1 balance in blocks	P ≥ 0.01	0.513
3	Runs Test	Number of consecutive 0s/1s	P ≥ 0.01	0.624
4	Longest Run of Ones in a Block	Detects long streaks	P ≥ 0.01	0.491
5	Discrete Fourier Transform (Spectral)	Detects periodic patterns	P ≥ 0.01	0.556
6	Binary Matrix Rank	Linear dependency	P ≥ 0.01	0.702
7	Overlapping Template Matching	Counts overlapping patterns	P ≥ 0.01	0.532
8	Non-overlapping Template Matching	Counts specific patterns	P ≥ 0.01	0.488
9	Maurer's Universal Test	Measures compressibility	P ≥ 0.01	0.611
10	Linear Complexity	Measures LFSR complexity	P ≥ 0.01	0.573
11	Serial Test	Frequency of overlapping m-bit patterns	P ≥ 0.01	0.529
12	Approximate Entropy	Checks irregularity in pattern occurrence	P ≥ 0.01	0.505
13	Cumulative Sums (Cusum)	Detects drift or trends	P ≥ 0.01	0.467
14	Random Excursions	Visits in random walk cycles	P ≥ 0.01	0.612
15	Random Excursions Variant	Counts occurrences of states in cycles	P ≥ 0.01	

From Table 1, we notice that All P-values are above 0.01 → ciphertext statistically passes the NIST tests. This indicates proposed algorithm produces ciphertext that is statistically indistinguishable from a random sequence.

Table 2. NIST Randomness Test Comparison

No.	Test Name	BSP-128 (P-value)	Serpent (P-value)	Blowfish (P-value)
1	Frequency (Monobit)	0.478	0.484	0.472
2	Frequency within Block	0.513	0.519	0.507
3	Runs Test	0.624	0.631	0.613
4	Longest Run of Ones in a Block	0.491	0.495	0.488
5	Discrete Fourier Transform	0.556	0.562	0.549
6	Binary Matrix Rank	0.702	0.715	0.689
7	Overlapping Template Matching	0.532	0.537	0.525
8	Non-overlapping Template Matching	0.488	0.501	0.492
9	Maurer's Universal Statistical Test	0.611	0.618	0.605
10	Linear Complexity	0.573	0.579	0.567
11	Serial Test	0.529	0.534	0.522
12	Approximate Entropy	0.505	0.512	0.498
13	Cumulative Sums (Cusum)	0.467	0.471	0.455
14	Random Excursions	0.612	0.619	0.608
15	Random Excursions Variant	0.588	0.591	0.582

Table 2. NIST Randomness Test Comparison

All three ciphers pass every NIST test, which is expected from strong block ciphers. BSP-128 and Serpent slightly outperform Blowfish in some short-block tests because of larger block size (128 bits) versus

Blowfish (64 bits). P-values are all > 0.01, meaning the ciphertext is statistically indistinguishable from random data. Differences in P-values are minor and due to block size and internal permutation differences. Larger block ciphers (BSP-128 /Serpent) have slightly more uniform statistical properties in short sequences.

The experimental results show that the proposed BSP-128 algorithm improves security and statistical performance compared with Blowfish and Serpent. The use of a 128-bit block size enhances resistance to ciphertext matching attacks and makes the algorithm more suitable for modern large-scale data encryption. The avalanche effect results (65.5%) indicate stronger diffusion than Serpent and significantly better performance than Blowfish. In addition, the algebraic degree growth and NIST randomness tests demonstrate strong nonlinearity and high statistical randomness of the generated ciphertext. Although the algorithm introduces additional operations that may increase implementation complexity, its design improves diffusion and reduces memory requirements, making it suitable for secure and efficient cryptographic applications.

5. Conclusion

The BSP-128 encryption algorithm introduces several structural and functional enhancements that can complement or strengthen existing ciphers like Blowfish and Serpent. By incorporating a 128-bit block size, an initial and final permutation, and a complex round function (P-function) involving byte transpositions, controlled rotations, and key-dependent operations, BSP-128 increases diffusion and nonlinearity across the data block. Its F-function applies multiple layers of substitution (S-boxes), permutation, linear transformation, and key mixing, ensuring high resistance against linear, differential, and related-key attacks. When used in conjunction with Blowfish or Serpent, BSP-128 can act as an additional encryption layer or as a component in a hybrid design, enhancing statistical randomness, key avalanche effect, and resistance to structural attacks. This layered design ensures that even if one underlying cipher exhibits structural weaknesses under certain attack vectors, the BSP-128 transformations further obscure patterns, confirming that the overall cryptosystem achieves stronger security and unpredictability, which would likely improve performance in NIST SP 800-22 randomness tests. During the design process, several things can be concluded about cipher design:

- The proposal BSP-128 algorithm is a Feistel network so that it is easy to implement and understands.
- By using a new a complicated design of F-function in the BSP-128 algorithm and using SPS instead of SP network we will make a strong and secure cipher.
- The mixing procedure makes use of different algebraic groups XOR, addition, and multiplication. BSP-128 provides the same algorithm for both encryption and decryption using the same key schedule; it can accommodate varying key lengths up to 256 bytes .
- With a minimal memory requirement (the S-boxes utilize only 512 bytes), BSP-128 is very easy to apply to smart cards or other machines with limited memory capacity .
- As even minor adjustments in one can have an impact on the other, both the design of the encryption algorithm and the key schedule must co-exist simultaneously .
- A strong key schedule and a solid round function will be a big deal, but it's not enough. Neither can work without the other if one is content with an inefficient and inelegant solution.

Future work may focus on further performance optimization, implementing the algorithm in hardware and embedded systems, and conducting additional security evaluations against advanced cryptanalytic attacks. These directions could further validate the effectiveness of BSP-128 for modern cryptographic applications.

References

[1] Patel, R., Kamboj, P. (2016). Security Enhancement of Blowfish Block Cipher. In: Unal, A., Nayak, M., Mishra, D.K., Singh, D., Joshi, A. (eds) Smart Trends in Information Technology and Computer Communications. SmartCom 2016. Communications in Computer and Information Science, vol 628. Springer, Singapore. https://doi.org/10.1007/978-981-10-3433-6_28

[2] S. B. Nalawade and D. H. Gawali, "Design and implementation of blowfish algorithm using reconfigurable platform," 2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE), Bhopal, India, 2017, pp. 479-484, https://doi.org/10.1109/RISE.2017.8378204

[3] H. Setiawan and K. Rey Citra, "Design of Secure Electronic Disposition Applications by Applying Blowfish, SHA-512, and RSA Digital Signature Algorithms to Government Institution," 2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 2018, pp. 168-173, https://doi.org/10.1109/ISRITI.2018.8864280
[4] M. A. Muin, M. A. Muin, A. Setyanto, Sudarmawan and K. I. Santoso, "Performance Comparison Between AES256-Blowfish and Blowfish-AES256 Combinations," 2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Semarang, 2018, pp. 137-141, https://doi.org/10.1109/ICITACEE.2018.8576929.
[5] S. Vyakaranal and S. Kengond, "Performance Analysis of Symmetric Key Cryptographic Algorithms," 2018 International Conference on Communication and Signal Processing (ICCSP), Chennai, 2018, pp. 0411-0415, https://doi.org/10.1109/ICCSP.2018.8524373
[6] S. Varshney, T. Sudarshan and S. Khare, "Efficient Hardware Architecture for Amalgam of Blowfish and Rc6," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), Mysore, 2017, pp. 1126-1130, https://doi.org/10.1109/CTCEEC.2017.8455189.
[7] I. A. Landge and B. K. Mishra, "VHDL based BLOWFISH implementation for secured Embedded System design," 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), Chennai, 2017, pp. 497-501, https://doi.org/10.1109/AEEICB.2017.7972363.
[8] Tariq.S.; Tanveer.u. and Ghazanfar. F.(2018). Serpent Algorithm: An improvement by 4 ×4 Sbox from finite Chain ring. International Conference on Applied and Engineering Mathematics (ICAEM), 2018(3):32-37. https://doi.org/10.25130/tjps.v25i6.320.
[9] Osvik DA. 2000. Speeding up serpent. In: AES candidate conference. 317–329.
[10] Ahmed I, Ali G, Hassin S. 2017. New approach for serpent block cipher algorithm based on multi techniques. Iraqi Journal of Information Technology 7(3):1–13.
[11] G. Farooq, "Serpent algorithm: An improvement by 4 × 4 S-box from finite chain ring," in Proc. 2018 Int. Conf. Applied and Engineering Mathematics (ICAEM), Pakistan, Nov. 2018, doi: 10.1109/ICAEM.2018.8536293.
[12] Yousif I. A. 2019. Proposed A permutation and substitution methods of serpent block cipher. Ibn AL- Haitham Journal For Pure and Applied Sciences 32(2):131 https://doi.org/10.30526/32.2.2120.
[13] Ali Y.H, Ressan HA. 2016. Image encryption using block cipher based serpent algorithm. Engineering and Technology Journal 34(2):278–28.
[14] Khan M, Shah T, Mahmood H, Gondal MA, Hussain I. 2012. A novel technique for the construction of strong S-boxes based on Chaotic Lorenz Systems. Nonlinear Dynamics 70(3):2303–2311 DOI 10.1007/s11071-012-0621.
[15] Elkamchouchi HM, Takieldeen AE, Shawky MA. 2018. A modified Serpent based algorithm for image encryption. In: 2018 35th National Radio Science Conference (NRSC). Piscataway: IEEE, 239–248 https://doi.org/10.1109/NRSC.2018.8354369.
[16] K. Hazra, A. Mahato, A. Mandal and A. K. Chakraborty, "A hybrid cryptosystem of image and text files using blowfish and Diffie-Hellman techniques," 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), Bangkok, 2017, pp. 137 141, https://doi.org/10.1109/IEMECON.2017.8079577.
[17] Zagi, H.R., & Maolood, A.T. (2020). A NOVEL SERPENT ALGORITHM IMPROVEMENT BY THE KEY SCHEDULE INCREASE SECURITY. Tikrit Journal of Pure Science. https://doi.org/10.25130/tjps.v25i6.320.
[18] Elshoush HT, Al-Tayeb BM, Obeid KT. 2021. Enhanced Serpent algorithm using Lorenz 96 Chaos-based block key generation and parallel computing for RGB image encryption. PeerJ Computer Science 7:e812 https://doi.org/10.7717/peerj-cs.812.
[19] Adeniyi, A.E.; Misra, S.; Daniel, E.; Bokolo, A., Jr. Computational Complexity of Modified Blowfish Cryptographic Algorithm on Video Data. Algorithms 2022, 15, 373. https://doi.org/10.3390/a15100373.
[20] A. T. Hashim, A. H. Jassem, and S. A. Ali, "A novel design of Blowfish algorithm for image security," Journal of Physics: Conference Series, vol. 1818, no. 1, Art. no. 012085, 2021, doi: 10.1088/1742-6596/1818/1/012085.
[21] W. W. Souror, M. Fouad, and A. Takieldeen, "Hybrid-Blowfish security strengths using side channel countermeasures," in Proc. 2023 Int. Telecommun. Conf. (ITC-Egypt), Alexandria, Egypt, Jul. 18–20, 2023,. https://doi.org/10.1109/ITC-Egypt58155.2023.10206358.
[22] A. H. Alwan, A. T. Hashim, and S. A. Ali, "Partial encryption scheme of medical images based on DWT, secret image sharing and hyperchaotic system," Traitement du Signal, vol. 41, no. 4, pp. 1807–1821, Aug. 2024, doi:10.18280/ts.410413.
[23] G. Singh and M. Kapoor, "Blowfish cryptography based security technique," International Journal of Research and Innovation in Multidisciplinary, vol. 2, no. 1, Jan. 2025.
[24] Chaudhari, A., Patil, S., Lokhande, M., Shinkar, P., Chougule, C., Pande, M. (2026). Comparative Analysis of ECC and Blowfish Algorithms for Cryptographic Security. In: Chaudri, J., Mahalle, P.N., Perumal, T., Joshi, A. (eds) ICT for Intelligent Systems. ICTIS 2025. Lecture Notes in Networks and Systems, vol 1519. Springer, Singapore. https://doi.org/10.1007/978-981-96-8901-9_52.
[25] Shakir M. "A new feedback symmetric block cipher method", Ph.D, Thesis University of Technology, Baghdad, 1997.
[26] Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., & Vo, S. (2010).
[27] A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (NIST Special Publication 800-22 Revision 1a). National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf.