Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

Secure and Compressed Data Transmission Using ECC Key Sharing, DNA Cryptography, and LZ77 Compression

Asamene Kelelom [1], Addisu Oumer [2], Pawan Kumar[3] Beer Singh[4]

[1,2] College of Engineering and Technology Samara University, Ethiopia.
[1,2]Samara University, Ethiopia
[3] COER University, Roorkee, India
[4]Seth Vishambhar Nath Institute of Engineering & Technology, Barabanki, INDIA

Emails: asamenek@su.edu.et, addisuoumer@su.edu.et, pawan0871@gmail.com, beersinghtu@gmail.com
*Corresponding author E-mail: asamenek@su.edu.et

ABSTRACT

The need for secure, lightweight, and efficient data protection mechanisms has become paramount in the changing environment of digital communication, especially for privacy-sensitive and bandwidth-limited applications. This article presents a new hybrid cryptographic system that combines three one-of-a-kind techniques in a synergistic way. Elliptic Curve Cryptography (ECC), DNA-based cryptography, and LZ77 data compression. The framework adopts ECC for the sharing of secret keys between the two communicative parties as its security is strong and the computational overhead is minimal. After the session key is agreed upon, DNA cryptographic methods are used for the encryption and decryption of the sensitive data, taking advantage of the high parallelism, randomness, and large encoding capacity of DNA sequences. To make the transmission even more efficient, the original text is first compressed via the LZ77 algorithm to remove redundancies prior to encryption. The security of this combined method to withstand cryptanalytic attacks is guaranteed, besides it allows for a significant level of data compression, thus it can be used for the secure transmission of data in limited environments such as Internet of Things (IoT) and telemedicine. The tests carried out confirm that the proposed protocol preserves privacy, integrity, and performance at the same time it is storage and transmission requirements are considerably optimized.

***Keywords**: ECC, Security, LZ77, DNA, Data Compression, Hybrid Cryptography, Encryption and Decryption.*

1.0 INTRODUCTION

Ensuring the safety of data in cloud databases is a very complicated and demanding problem due to the nature of the system being prone to hostile assaults, data breaches, and insecure access points. Over the last years, many researchers have come up with a variety of security methods such as Access Control, Intrusion Detection and Avoidance Systems, Encryption-Based Storage System, and Key Management Techniques to enhance the data security essnvironment. With the advent of a networked digital world, the protection of sensitive data in transmission has become the most important issue.

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

Cryptographic techniques are the only methods through which this can be achieved since they also maintain the confidentiality, integrity, and authenticity of communication over the internet.

The paper entitled "Secure and Compressed Data Transmission Using ECC Key Sharing, DNA Cryptography, and LZ77 Compression" identifies the issues that most communication systems face with respect to confidentiality, integrity, and efficiency. This study presents a novel system that combines Elliptic Curve Cryptography (ECC) for secure key exchange, DNA cryptography for advanced encryption, and LZ77 compression for effective data size reduction. The reason for such an integration is the growing demand for secure communication over bandwidth-constrained and vulnerable networks such as those of IoT and cloud environments.

ECC is a well-known public-key cryptography technology that efficiently ensures security while using lower key sizes, which decreases computational overhead and increases efficiency. Additionally, when it is used with key-sharing protocols like Elliptic Curve Diffie-Hellman (ECDH), the two parties wishing to communicate are moved from the public communication channel to the private one at the conclusion of the session. Because of the robust basis upon which the most common forms of attacks are minimised, it is nearly hard for an attacker to intercept the "common" key. Nonetheless, it is still possible to accomplish lightweight execution of devices with limited resources; for example, ECC may be utilised for secret data transfer and secure key agreements [26,27,28].

Utilising DNA sequences, DNA cryptography codes and encrypts data using the principles of molecular biology. DNA encryption transforms digital data into nucleotide base sequences (Adenine, Thymine, Cytosine, Guanine), in contrast to traditional binary-based cryptography. In order to guarantee maximum security and pave the way for innovative encryption techniques, this approach takes advantage of the very complicated and dense genetic code. In order to increase resilience and unpredictability, which in turn address the problems of data integrity and protocol standardisation, the most recent additions also overlay chaotic dynamics on top of the encryption process. DNA cryptography is a new kind of secure communication that may also be utilised for data storage [29–33].

The study used LZ77 compression, a traditional lossless compression method that finds repeated data sequences and leverages that redundancy to represent the data with less storage without losing any information, to increase transmission efficiency. Therefore, the system can conserve bandwidth and reduce the delay time if the LZ77 compression is performed to the data prior to encryption or the transmission process. In order to make it a suitable carrier of security for compressed data transfer, a number of recent publications have examined the problems of computation sensitivity and the possibility of optimisation to the extent of plugging solutions for cryptographic frameworks [34–37].

This work thoroughly investigates the synergy between LZ77 compression, DNA-based encryption, and ECC key exchange. Such a combined system operates in the background to send data in a safe and dependable way. Because it uses less bandwidth and is thus economical, it may be applied in a wide range of scenarios. In comparison to traditional cryptosystems, the work's theoretical approach, algorithm design, and experimental verification yield more security and improved transmission efficiency.

The invention of ECC to accomplish effective and secure key distribution is the first item on the list of the accomplishments. Second, as a unique method of data encryption, the study introduces the idea of combining chaotic dynamics with DNA cryptography. Ultimately, the authors have chosen to transmit

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

data with lower overheads by using LZ77 compression. The primary focus of this multidisciplinary effort is modern communication, which demands both strict security and practical efficiency.

2.0 RELATED WORK

Qazi et al. [1] proposed a method that used the Elliptic Curve Digital Signature (ECDSA) cryptographic technique to control node memory space and secure node-to-node communication. Additionally, the Algorithm for Wireless Secure Communication (ASCW) suggested the key management with the most appropriate time. ASCW contributed to the improvement of node-level communication security by reducing network and security threats. A real-world testbed was put up to verify data packet size, greeting messages, and key generation time. Kumar and Sharma [2] described a genetic algorithm-based ECC-key generation technique that includes uniform crossover, mutation, selection, fitness evaluation, and chromosomal beginning. Both private and public keys were generated by the methodology itself; the private key was selected at arbitrary from the curve's range. With an emphasis on how it could improve cryptographic security and efficiency in environments with limited resources, this study assesses the ECC-GA method's performance and resource utilisation. In addition, search space and key size were taken into account.

Novel key generation and encryption techniques were developed by Suthanthiramani et al. [3] to guarantee the security of the private information kept in cloud databases. It combined the first non-key attribute with the primary key value's least common multiple and greatest common divisor. Elliptic Curve Cryptography with Base100 Table, a cutting-edge method for encryption and decryption, was presented in the study. When compared to the current cloud storage methods, the suggested model increases data security by at least 5%. Moosavi and Izadifar [4] are thinking on a comprehensive IoT security solution. First, a secure mutual authentication system using Elliptic Curve Cryptography (ECC) with Quark lightweight hash design; second, secure end-to-end communication using ECC and DNA. Messages are encrypted and decrypted using biologically generated DNA sequences in DNA-based cryptography.

Ma and Du [5] launched an attribute-based strong designated verifier signature scheme that is computationally efficient. To this end, they use elliptic curve encryption to minimize the computational cost of the operations. The proposed method improves computational efficiency and, due to the use of a more efficient access framework, it is preferred not only for better access control, but also for resource-limited cloud end-users situations. Hagras [6] have introduced novel authenticated public key elliptic curve based on deep convolutional neural network (APK-EC-DCNN) targeted for image encryption in cybersecurity. The scheme features the use of elliptic curve discrete logarithmic problem to secure key exchange and adopts a chaotic system to increase the security, which leads to fast and robust encryption and, at the same time, the good performance of the scheme in comparison with the existing ones. Kumar et al. [7] presented a hybrid encryption scheme that integrates DNA Cryptography with Elliptic Curve Cryptography (ECC) and exploits the high randomness of DNA coding and the strong security of ECC. The method greatly exceeds the existing methods and achieves higher values of such metrics as entropy, correlation coefficient, and PSNR. The paper [8] proposes a DNA-based lightweight cryptography system (DNA-LWCS) that enhances the IoT communication security by using ellipse curve encryption (ECC) along with minor encryption techniques. In this system, three essential keys are extracted from publicly available DNA sequences that together generate a private key. The lightweight design of the system not only ensures the integrity and confidentiality of the IoT data exchanges but also does not put a significant burden on the system resources, thus it is appropriate for IoT applications. Mukherjee et al. [9] have proposed a Genetic Algorithm method for the improvement of weak DNA-based cryptographic keys that can help to

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

overcome security problems caused by the redundant and observable patterns of the keys. This method implies the alteration of fitness functions and the use of genetic agents designed for DNA crypto operations. The authors claim that in the original population of 25 x 25 DNA keys 14 weak keys have been found, and they suggest a way of generating up to 8 new populations. Berezin et al. [10] The bioeconomy depends on the synthetic DNA sequences for the verification. DNA watermarks can hold the identification information and improve the communication through error correction and encryption procedures. Innovations in the digital signature methods make it possible not only public authentication of unaltered sequences, but also self-documentation of synthesized DNA. Special concerns for GMOs are necessary if we want to have public verification, establish authorship, facilitate traceability, and detect illicit usage. Zheng et al. [11] A new DNA motif with both enthalpy and entropy features has been designed. It is used for biosensing and information encryption. The pH-responsive A+/C DNA motif is capable of programmable changes, which affect the entropic parts and the enthalpic control. The method is confirmed by thermodynamic studies, thus it can precisely optimize the performance of the motif. The prepared DNA motifs have been turned on to glucose biosensing and crypto-steganography systems, thus they can be developed in both fields.

Patel and Veeramalai [12] have proposed a safe image transmission method by merging chaotic maps, Halton sequences, a 5D hyper-chaotic system, and DNA encoding in a picture encryption technique. To create the HaLT map, which is a chaotic map, chaotic maps were combined with the Halton sequence, which was used for the cryptography of an initially scrambled image. The operation of encryption has multiple tiers of obfuscation and, for that reason, uses a 5D hyper-chaotic map to generate the random sequences that are responsible for ensuring a higher security level.

Chen et al. [13] have devised a two-stage system for DNA sequencing data compression, which includes repacking the original data into a binary stream and using an LZMA encoder. The method was planned to be implemented in an extremely compressed LZMA stream for which LZMA was accelerated by FPGA to get the performance enhancement. The utility "repaq" was introduced as a lossless, non-reference compressor for the FASTQ format files that can achieve better compression ratios than the existing FASTQ compressors. Besides, the design is also suitable for other sequencing data compression methods.Hong and Boucher et al.[14] (Year) provide a comprehensive evaluation of the evolutions of LZ77-based compression algorithms and their variants, with a particular focus on genetic data. The authors delineate the difficulties caused by highly repetitive nature of genetic sequences and also survey the effectiveness of various methods, e.g., prefix-free parsing and AVL grammars, in improving both compression ratios and computational speed. By employing the inherent redundancy in genomic data, these methods show dramatic enhancements over conventional ones. Besides merging advanced compression methods, this article serves as a resource hub directing new research paths in the field of genomics.

Nishimoto and Tabei [15] have introduced efficient algorithms for the non-overlapping Lempel-Ziv-77 factorization calculation as well as the longest preceding non-overlapping factor table calculation that use succinct suffix tree representations. Their solutions operate within a small space model and address issues of huge sequence processing. Moreover, the paper also presents substring compression requests for Lempel-Ziv-78 factorization, which can be subject to a logarithmic multiplicative delay depending on the suffix tree implementation. Major contributions include advancements in substring compression searches, LPnF table computation, and use of suffix trees for non-overlapping Lempel-Ziv factorizations.Kempa and Langmead [16] have presented a space-optimized LV grammar construction from LZ77 parses, which achieves Re-Pair competitive sizes while allowing faster RL-

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

BWT computation by interaction with modern LZ77 techniques. This [17] work is of profound theoretical value for space-efficient LZ factorization and clearly has potential applications in both genomics and data compression. Despite the fact that the technical scope is quite impressive, making it more accessible and subjecting it to empirical testing would increase its impact significantly. It is very well suitable for publication in the Algorithms journal, subject to minor modifications facilitating readability and usability.Saxena and Kumar have [21] proposed an innovative and efficient, it proposes a hybrid security framework such that biometric hashing, mutual authentication of ElGamal encryption, and fuzzy logic. DNA + Palliér hybrid cryptography nested by Saxena and Kumar [22], It has been demonstrated to security, significantly reduce bandwidth, and improve encryption/ decryption performance compared with existing methods. Thus, it is the most suitable way of secure information exchange over an insecure network. The three-tier hybrid encryption framework was the invention of combines AES, RSA, and ECIES to greatly improve the security of data while computation time is reduced as compared with traditional ECIES, making this technology highly appropriate for secure digital communication [23]. Encryption is a very significant key exchange security upgrade and proposes exponentially stronger hybrid keys, thus AES + Paillier has become a very powerful hybrid cryptography model [24]. Dual-phase Hypervisor Controller employing EM clustering and Fuzzy Time Series analysis significantly enhances the DDoS detection accuracy in the cloud environment, thus, it can outperform existing IDSs with higher detection rates and lower false negatives [25].

3.0 BACKGROUND

Elliptic Curve Cryptography (ECC) is founded on the generation of secure key pairs for digital signatures, encryption, and key exchange. Whereas RSA utilizes large integers, ECC keys are derived from points on an elliptic curve, providing the same security with considerably shorter key lengths.

An elliptic curve over a finite field is defined by:

$$y^2=x^3+ax+b(\text{mod } p)$$

where:

p is a large prime (for prime fields) or 2^m (for binary fields).

a and b are coefficients defining the curve.

The curve must be non-singular ($4a^3+27b^2\neq 0$).

The order n of base point on the curve G (smallest integer where nG=O, the point at infinity) must be a large prime to resist attacks.

Private Key (K_x): A randomly selected integer dd where $1\leq K_x n-1$.

Public Key (K_y): Computed as $Q=d\times G$ (scalar multiplication of G by d).

Choose a cryptographically secure random number d within [1,n−1].

Compute the public key $Q=d\times G$ using elliptic curve point multiplication.

Output:

Private key: K_x (kept secret)

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

Public key: K_y (shared publicly)

3.1. DNA CRYPTOGRAPHY

DNA Cryptography is the science of hiding digital information inside DNA.The individual character of the DNA sequence makes dissimulation, encryption and indirect encryption of sequence-based information in the DNA sequence possible. Mimicking the biological DNA's information storage capacity, which can hold large volumes of information efficiently, this method uses the four nucleotides Adenine (A), Thymine (T), Cytosine (C), and Guanine (G) as a novel form of data's representation and transformation. The algorithm of DNA cryptography typically involves encoding data into DNA sequences, applying biological or mathematical operations, and then decoding the result. While implementations can vary, most DNA cryptography algorithms follow these core steps [18,19]:

General Steps in a DNA Cryptography Algorithm

Data Encoding

Convert plaintext into a binary format.

Map binary data to DNA bases (A, T, C, G) using a predefined encoding rule. For example, 00 = A, 01 = C, 10 = G, 11 = T.

Apply various DNA-inspired operations, such as:

Substitution: Replace DNA bases according to a key or rule.

Complementation: Use base-pairing rules (A↔T, C↔G) to transform the sequence.

Permutation/Scrambling: Rearrange the order of DNA bases or blocks to increase confusion.

Algebraic Operations: Perform arithmetic or logical operations on DNA sequences (e.g., XOR, addition).

Encryption Process

Combine DNA sequence operations with key-based transformations to produce the encrypted DNA sequence.

Decryption Process

Reverse the DNA operations using the correct keys and rules to recover the original DNA sequence.

3.2. Lz77 DATA COMPRESSION

The LZ77 data compression algorithm is a dictionary-based, lossless compression method that encodes repeated sequences in data as references to previous occurrences within a sliding window. Here's a step-by-step outline of the LZ77 algorithm [20]:

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

LZ77 Compression Algorithm Steps

Initialize Buffers: Maintain a search buffer (the sliding window of recently processed data) and a look-ahead buffer.

Symbol	Description
S	Input sequence (string or byte stream)
i	Current position in the input sequence
W⊂S	Search buffer (window before position i)
L⊂S	Look-ahead buffer (window starting at position i)
Ti	Encoded triple at position i, i.e., (o,ℓ,c)

Process Input: For each position in the input, search the search buffer for the **longest match** with the prefix of the look-ahead buffer.

Match Found in the Search Buffer

Let:

$m \in W$ be the longest match for a prefix of L

o be the *offset* of the match: $o = i - \text{position}(m)$

ℓ be the *length* of the matched substring: $\ell = |m|$

c be the *next symbol* following the match in the look-ahead buffer: $c = S[i + \ell]$

Then the encoded triple is: $T_i = (o, \ell, c)$

Case 2: No Match Found

If no match is found:

$o = 0$

$\ell = 0$

$c = S[i]$

Then the encoded triple is: $T_i = (0, 0, S[i])$

4.0 Methodology

Symbol	Description
$E(F_p)$	Elliptic curve defined over a finite field F_p
G	Generator point of the elliptic curve E
∞	Point at infinity (identity element for elliptic curve addition)
a, b	Coefficients of the elliptic curve equation: $y^2 = x^3 + ax + b$
n	Order of the subgroup generated by G
h	Cofactor, computed as h = # $E(F_p)$/ n
K	Key space (set of all valid ECC private keys)

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

M	Message space (set of all valid plaintext messages)
K_x	ECC private key (randomly selected from key space K)
K_y	ECC public key, computed as $K_y = K_x \cdot G$
$M \in$ M	A plaintext message from the message space
B_M	Binary representation of the message M
D_M	DNA sequence representation derived from binary message B_M
B_K	Binary representation of the public key K_y
B'_K	Padded or repeated binary key to match the length of B_M
B_D	Binary form of the DNA sequence D^M
B_E	Encrypted binary sequence after processing
D_E	Encrypted DNA sequence
C	Final compressed ciphertext (byte data)

4.1 Main Execution Flow

1. Input message
2. Generate (sk, vk) = GenerateECCKeys()
3. encrypted_dna = DNAEncrypt(message, vk)
4. compressed_data = CompressDNA(encrypted_dna)
5. decompressed_dna = DecompressDNA(compressed_data)
6. decrypted_msg = DNADecrypt(decompressed_dna, sk)
7. Output decrypted_msg

4.2 ZlIB Comression and decompression pseudo code:

1. Choose curve $E(F_p)$ (e.g., NIST P-256).
2. Generate private key $K_x \in Z_p$.
3. Compute public key: $K_y = K_x \cdot G$.
4. Return (Private key K_x, Public key K_y).
5. Convert each character $c \in M$ to 8-bit binary.
6. Concatenate all binary values to get B_M.
7. Return (Binary String B_M).
8. Define map: '00' → A, '01' → T, '10' → C, '11' → G.
9. Apply mapping to 2-bit segments of B_M to get D_M.

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

10. Return D_M.
11. Convert $M \rightarrow B_M$ using TextToBinary().
12. Convert $B_M \rightarrow D_M$ using BinaryToDNA().
13. Convert $K_y \rightarrow B_K$ (binary form).
14. Pad B_K to match length of B_M: $B'_K = Repeat(B_K)$.
15. Convert $D_M \rightarrow B_D$ (reverse map to binary).
16. Compute: $B_E = B_D \oplus B'_K$
17. Convert $B_E \rightarrow D_E$ using BinaryToDNA().
18. Return D_E.
19. Convert D to UTF-8 bytes.
20. Compress using zlib $\Rightarrow C$.
21. Return C.

The sends the encrypted and compressed message. The encrypted message travels over the networks. Due to compressed encrypted message size will be less other than without compressed encrypted message. The receiver receives the message, firstly the receiver will decompressed the message before decrypted. And follows following algorithm that are described below.

1. Decompress C using zlib.
2. Decode UTF-8 to obtain D.
3. Return D.

1. Compute $K_y = K_x \cdot G$ and convert to B_K.
2. Pad B_K to match length of $D_E \rightarrow B'_K$.
3. Convert $D_E \rightarrow B_E$.
4. Compute: $B_D = B_E \oplus B'_K$.
5. Convert $B_D \rightarrow M$ using BinaryToText().
6. Return M

The algorithm derives a binary key from Kx by multiplying it with G, then pads it to match the encrypted data length. It converts the encrypted data into binary and performs an XOR operation with the padded key to get the decrypted binary form. Finally, it transforms this binary data into readable text and returns the original message.

5.0 RESULT AND DISCUSSION

Encrypted size scales deterministically with the original here: every encrypted output is exactly 4× the original message size, while compression varies by content and becomes more effective as the message grows. Compressed size starts larger than the original for very small inputs, then drops below the original after roughly 2 KB and trends toward about 42% of the original by ~8 KB.

Table 1. comparison between message size, encrypted message size and compressed data size in bytes.

Original Message	Encrypted Message	Compressed Message
30 bytes	120 bytes	68 bytes

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

61 bytes	244 bytes	109 bytes
122 bytes	488 bytes	191 bytes
244 bytes	976 bytes	352 bytes
488 bytes	1952 bytes	664 bytes
977 bytes	3908 bytes	1265 bytes
1954 bytes	7816 bytes	1903 bytes
3909 bytes	15636 byte	2306 bytes
5865 bytes	23460 bytes	2885 bytes
7819 bytes	31276 bytes	3291 bytes

The encrypted size can be computed using a simple formula where the encrypted data length is always four times the original message size. For example, if the original message is 977 bytes, the encrypted size will be $4 \times 977 = 3908$ bytes. Estimating the compressed size is less straightforward because it depends on the content and its redundancy. There is no fixed formula for compression, so the best approach is to use empirical ratios derived from observed data and interpolate between known values. First, sort the data by original size, then identify the two original size entries that bracket the message size you want to estimate. After that, perform a linear interpolation based on the compression ratios (compressed size divided by original size) that are shown in tables. As an example, small messages have a compression ratio of more than 1, which means they can be initially expanded when compressed, but the ratio decreases significantly as the message size increases. The data-derived ratios indicate that the compression ratio is about 2.27 at the very small message (30 bytes) and it gradually goes down to less than 1 around 2 KB and then it stabilizes between approximately 0.4 and 0.6 for the messages that are larger than several KB. In other words, compression becomes more and more efficient as the message size gets larger, thus the data size is being reduced to about 40-60% of the original. To put it simply, encryption is almost always expanding the data size by a factor of four, while compression efficiency gets better with the length and the content of the message, so an estimation from empirical data is needed to be accurate.

Table 2. comparison between encryption time, decryption time, data compression time and data decompression time in microseconds.

EncryptionTime	**DecryptionTime**	**Compression Time**	**Decompression Time**
1095 µs	1147 µs	95 µs	21 µs
1779 µs	1126 µs	67 µs	18 µs
3597 µs	3258 µs	72 µs	17 µs
6586 µs	4573 µs	119 µs	27 µs
12843 µs	10854 µs	143 µs	51 µs
28753 µs	18349 µs	403 µs	40 µs
56085 µs	37343 µs	607 µs	40 µs
109375 µs	68676 µs	865 µs	50 µs
159423 µs	135745 µs	1397 µs	1818 µs
217820 µs	140449 µs	1427 µs	79 µs

The table shows a clear trend in the computational times for encryption, decryption, compression, and decompression across varying data sizes or complexities.

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

5.1 Encryption and Decryption Times:

Both encryption and decryption times increase substantially with the size and complexity of the data. Encryption times rise from about 1,095 µs to over 217,820 µs, demonstrating a near-exponential growth as data scales. Decryption times also increase from around 1,147 µs to 140,449 µs, but decryption is generally faster than encryption for larger datasets. This is typical as encryption often involves more complex computations than decryption.

5.2 Compression and Decompression Times:

Compression times are still pretty low compared to encryption and decryption, starting at 67 µs and going up to 1,427 µs. This means that compression doesn't use as much processing power, but it still works with larger amounts of data. Decompression times are always the fastest, usually less than 100 µs, with one exception at 1,818 µs. This means that decompression is very well optimised and works very quickly, usually faster than any other operation.

5.3 Performance Insights:

The most time-consuming operation in the process is encryption, while decryption is resource-intensive but usually faster than encryption. Compression and decompression are significantly faster, and decompression imposes the least computational resource demand. Interestingly, at the data point corresponding to 159,423 µs encryption time, there is a sudden increase in decompression time, which may indicate specific variations in data characteristics or processing overhead at that scale. From such performance characteristics, system design and optimization should focus mainly on enhancing the efficiency of encryption and decryption, as these have the greatest impact on processing time, whereas compression and decompression, since much faster, might have a lesser effect on overall throughput.

Table 3. comparison between memory utilization during encryption, decryption compression and decompression.

Encryption	Decryption	Compress	Decompression
14.03 KB	13.34 KB	293.81 KB	23.02 KB
27.26 KB	26.21 KB	293.81 KB	23.02 KB
53.49 KB	51.73 KB	293.81 KB	23.02 KB
104.02 KB	100.82 KB	293.81 KB	23.02 KB
207.22 KB	201.17 KB	293.81 KB	23.02 KB
414.42 KB	402.64 KB	293.81 KB	23.02 KB
829.49 KB	806.26 KB	293.81 KB	23.71 KB
1661.84 KB	1615.7 KB	293.81 KB	31.34 KB
2472.02 KB	2402.94 KB	293.81 KB	119.11 KB
3330.20 KB	3238.23 KB	293.81 KB	119.11 KB

The table 3 shows, how much memory is used during encryption, decryption, compression, and decompression. The amount of memory needed for encryption and decryption grows a lot with the size of the data. For small inputs, it starts at about 14 KB and goes up to for the biggest data sizes, more than 3,330 KB. Encryption usually uses a little more memory than

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

decryption, but the difference between the two is still small and stays the same across all data sizes. The amount of memory used for compression is always about 293.81 KB across all tested data points. This means that the compression algorithm needs a constant amount of memory, no matter how big the input is. This could be because of an internal buffer or a fixed algorithmic design. The amount of memory used for decompression is also low and stable, changing only a little from about 23 KB to 119 KB as the size of the data grows. This shows that decompression uses resources in an efficient and predictable way. The table shows that the memory needs for encrypting and decrypting data grow in direct proportion to the size of the data. On the other hand, the memory needs for compressing and decompressing data stay about the same and low. This means that, in terms of memory resources, it is important to make the processes of encrypting and decrypting as efficient as possible when working with large data sets. Compression and decompression, on the other hand, are less likely to cause memory problems.

6.0 CONCLUSION

These findings indicate that encryption and decryption are the significant computational and memory bottlenecks in any secure data processing workflow, and it is there that targeted optimizations must be done to achieve efficiency for large-scale applications. The compression and decompression steps are important in reducing data size and transmission bandwidths but have predictable resource usage and much more modest, which are less likely to constrict system performance.But additional investigation could be done on making encryption algorithms or acceleration methods that are more efficient and lower the high time and memory costs of cryptographic processing. Also, adaptive compression techniques that better fit the characteristics of the data may improve the use of storage and networks even more without slowing things down. This effort will set a basic standard for finding the right balance between security, speed, and resource use, which will help make secure data transmission systems even better.

Author Contributions

All authors contributed equally to the conception and design of the study.

Asamene Kelelom and Addisu Oumer developed the cryptographic framework and implemented the ECC–DNA–LZ77 methodology. Pawan Kumar and Beer Singh carried out performance evaluation, analysis of results, and manuscript drafting. All authors reviewed, revised, and approved the final version of the manuscript.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

References

[1] R. Qazi, K. N. Qureshi, F. Bashir, N. U. Islam, S. Iqbal, and A. Arshad, "Security protocol using elliptic curve cryptography algorithm for wireless sensor networks," J. Ambient Intell. Humanized Comput., vol. 12, pp. 547–566, 2021.

[2] S. Kumar and D. Sharma, "Key generation in cryptography using elliptic-curve cryptography and genetic algorithm," Eng. Proc., vol. 59, no. 1, p. 59, 2023.

[3] P. Suthanthiramani, S. Muthurajkumar, G. Sannasi, and K. Arputharaj, "Secured data storage and retrieval using elliptic curve cryptography in cloud," Int. Arab J. Inf. Technol., vol. 18, no. 1, pp. 56–66, 2021.

[4] S. R. Moosavi and A. Izadifar, "End-to-end security scheme for e-Health systems using DNA-based ECC," in Proc. Silicon Valley Cybersecurity Conf., Cham, Switzerland: Springer, Dec. 2021, pp. 77–89.

[5] R. Ma and L. Du, "Efficient attribute-based strong designated verifier signature scheme based on elliptic curve cryptography," PLoS One, vol. 19, no. 5, p. e0300153, 2024.

[6] E. A. Hagras, S. Aldosary, H. Khaled, and T. M. Hassan, "Authenticated public key elliptic curve based on deep convolutional neural network for cybersecurity image encryption application," Sensors, vol. 23, no. 14, p. 6589, 2023.

[7] V. N. Kumaran et al., "A secure medical image encryption technique based on DNA cryptography with elliptic curves," Sci. Rep., vol. 15, no. 1, pp. 1–18, 2025.

[8] S. Aqeel, A. S. Khan, I. A. Abbasi, F. Algarni, and D. Grzonka, "Enhancing IoT security with a DNA-based lightweight cryptography system," Sci. Rep., vol. 15, no. 1, p. 13367, 2025.

[9] P. Mukherjee et al., "Best fit DNA-based cryptographic keys: The genetic algorithm approach," Sensors, vol. 22, no. 19, p. 7332, 2022.

[10] C. T. Berezin, S. Peccoud, D. M. Kar, and J. Peccoud, "Cryptographic approaches to authenticating synthetic DNA sequences," Trends Biotechnol., 2024.

[11] L. L. Zheng et al., "Enthalpy and entropy synergistic regulation–based programmable DNA motifs for biosensing and information encryption," Sci. Adv., vol. 9, no. 20, p. eadf5868, 2023.

[12] S. Patel and T. Veeramalai, "Image encryption using a spectrally efficient Halton logistics tent map and DNA encoding," Entropy, vol. 24, no. 6, p. 803, 2022.

[13] S. Chen et al., "Efficient sequencing data compression and FPGA acceleration based on a two-step framework," Front. Genet., vol. 14, p. 1260531, 2023.

[14] A. Hong and C. Boucher, "Enhancing data compression: Recent innovations in LZ77 algorithms," J. Comput. Biol., 2025.

[15] T. Nishimoto and Y. Tabei, "LZRR: LZ77 parsing with right reference," Inf. Comput., vol. 285, p. 104859, 2022.

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

[16] D. Kempa and B. Langmead, "Fast and space-efficient construction of AVL grammars from the LZ77 parsing," in LIPIcs–Leibniz Int. Proc. Informatics, vol. 204, 2021, p. 56.

[17] D. Köppl, "Non-overlapping LZ77 factorization and LZ78 substring compression queries with suffix trees," Algorithms, vol. 14, no. 2, p. 44, 2021.

[18] N. Dhamala and K. P. Acharya, "A comparative analysis of DES, AES and Blowfish based DNA cryptography," Adhyayan J., vol. 11, no. 11, pp. 69–80, 2024.

[19] X. Xue, D. Zhou, and C. Zhou, "New insights into the existing image encryption algorithms based on DNA coding," PLoS One, vol. 15, no. 10, p. e0241184, 2020.

[20] R. K. Giri et al., "An innovation analysis of LZ77 and LZ78 compression algorithms for data compression & source coding," in Proc. 15th Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT), Kamand, India, 2024, pp. 1–5.

[21] V. Saxena and P. Kumar, "Secure transaction of digital currency through fuzzy based cryptography," Indian J. Sci. Technol., vol. 16, no. 37, pp. 3148–3158, 2023.

[22] P. Kumar and V. Saxena, "Nested levels of hybrid cryptographical technique for secure information exchange," J. Comput. Commun., vol. 12, no. 2, pp. 201–210, 2024.

[23] P. Kumar, V. Saxena, and K. V. Singh, "Analysis of hybrid cryptography for secure exchange of information," Int. J. Comput. Appl., vol. 185, no. 4, pp. 37–42, 2023.

[24] P. Kumar and V. Saxena, "Hybrid cryptography for security key exchange through AES and Paillier," Eur. Chem. Bull., vol. 12, no. 10, pp. 3913–3921, 2023.

[25] S. Kumar et al., "Securing cloud-based systems: DDoS attack mitigation using hypervisor-intrusion detection approach," Procedia Comput. Sci., vol. 259, pp. 1366–1375, 2025.

[26] M. Pundir and A. Kumar, "An efficient conference key agreement protocol suited for resource constrained devices," J. Parallel Distrib. Comput., vol. 196, p. 105011, 2025.

[27] A. Kumar and M. Hussain, "Secure ECC based key exchange mechanism for devices in IoT networks," in Proc. 14th Int. Conf. Contemporary Comput., Aug. 2022, pp. 175–179.

[28] Z. Qin et al., "An efficient key management scheme based on ECC and AVL tree for large scale wireless sensor networks," Int. J. Distrib. Sens. Netw., vol. 11, no. 9, 2015.

[29] X. Huang et al., "Towards next-generation DNA encryption via an expanded genetic system," Nat. Sci. Rev., vol. 12, no. 4, Apr. 2025.

[30] A. Kaushik and S. Satvika, "The chaotic dynamics of DNA: A survey on DNA cryptography," Int. J. Comput. Appl., vol. 187, no. 16, pp. 29–37, Jun. 2025.

[31] V. N. S. Kumaran et al., "A secure medical image encryption technique based on DNA cryptography with elliptic curves," Sci. Rep., vol. 15, p. 20003, 2025.

[32] T. Mahjabin et al., "A survey on DNA-based cryptography and steganography," IEEE Access, vol. 11, pp. 116423–116451, 2023.

Received: 2025-10-24
Accepted: 2026-01-11
Published Online: 2026-01-20
DOI: 10.70454/JRICST.2026.30101

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

[33] L. Chu et al., "A review of DNA cryptography," Intell. Comput., vol. 4, p. 0106, 2025.

[34] J. Blocki, S. Lee, and B. S. Y. Garcia, "Differentially private compression and the sensitivity of LZ77," arXiv preprint arXiv:2502.09584, 2025.

[35] Y. Huang, A. Song, C. Guo, and Y. Yang, "ASIC design of LZ77 compressor for computational storage drives," Electron. Lett., vol. 59, no. 22, p. e13000, 2023.

[36] R. K. Giri et al., "An innovation analysis of LZ77 and LZ78 compression algorithms for data compression & source coding," in Proc. 15th ICCCNT, Kamand, India, 2024, pp. 1–5.

[37] J. Blocki, S. Lee, and B. S. Y. Garcia, "Differentially private compression and the sensitivity of LZ77," arXiv preprint arXiv:2502.09584, 2025.