ISSN: 2579-079X

International Digital Organization for Scientific Research IDOSR JST/25/1120000
IDOSR JOURNAL OF SCIENCE AND TECHNOLOGY 11(2):46-54, 2025.
https://doi.org/10.59298/IDOSR/JST/25/112.465400

Overview of Algorithms for Encoding and Decoding Payment Systems

[1]Osondu E. Ogbodo, [2]Obikwelu R. Okonkwo, [3]Godspower I. Akawuku and [4]Chimeremeze P. Ejimadu

[1,2,3,4] **Department of Computer of Science, Nnamdi Azikiwe University, Awka.**
Email: Osondu_o@yahoo.com, or.okonkwo@unizik.edu.ng, gi.akawuku@unizik.edu.ng, prevailexcellent@gmail.com

ABSTRACT

Algorithms for encoding and decoding information play a critical role in the optimization of modern systems, enabling efficient data representation, transmission, storage, and retrieval. This paper explores a landscape of encoding and decoding algorithms in payment systems with a focus on enhancing system performance across diverse domains, including communication networks, distributed computing, and machine learning. We analyze lossless and lossy encoding techniques, discussing their trade-offs in terms of compression ratio, computational complexity, and resilience to errors. Special attention is given to entropy-based methods such as Huffman and arithmetic coding, as well as modern techniques like neural compression and error-correcting codes including LDPC and Reed–Solomon codes. Furthermore, we investigated adaptive and context-aware encoding schemes that dynamically adjust to data patterns and system requirements, contributing to real-time optimization. By integrating these algorithms within larger system architectures, substantial gains in throughput, latency, and energy efficiency are demonstrated. The paper concludes with a discussion on current challenges, including scalability and security, and proposes future research directions that leverage advances in artificial intelligence and quantum computing to further enhance encoding and decoding strategies for next-generation system optimization.

Keywords: Algorithms, Encoding, Decoding, Compression, Adaptive Systems

INTRODUCTION

Information Security has become a tremendously important issue to deal with because of the demand of Internet to provide essential communication between tens of millions of people and is being increasingly used as a tool for commerce, there are many aspects to security and many applications, ranging from secure commerce and payments to private communications and protecting passwords. One essential aspect for secure communication is that of cryptography. But it is important to note that while cryptography is necessary for secure communication, it is not sufficient by itself, says [1], in his recent works. Cryptography revolves around Encryption and Decryption where Encryption is a process in which plain text data is converted into an unreadable text called the ciphertext and decryption is the process of transforming data that has been rendered unreadable (ciphertext) back to its normal form. [2]. Security and privacy are the main features expected in the field of online transactions. Online transactions need the utmost security to avoid possible fraudulent transactions of any kind. The encryption of the information is the source of security and privacy in this online transaction. Security is provided in the form of a password, pin code, biometrics, digital signature, steganography, etc. [2]. The diagram in figure 1.0 shows the stages and process of encrypting and decrypting algorithm which could be adopted for information system data system security.

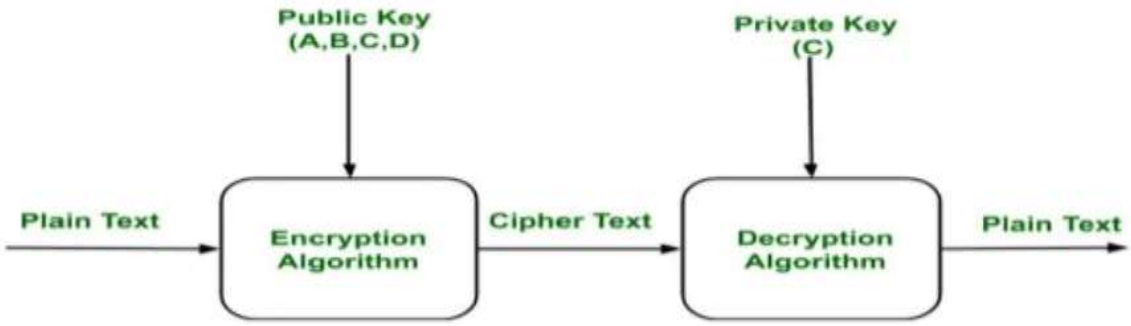

Figure 1: Diagram Encryption and Decryption Algorithm [2]

The Purpose of Cryptography

Cryptography is the science of writing in secret code and is an ancient art; the first documented use of cryptography in writing dates to circa 1900 B.C. when an Egyptian scribe used non-standard hieroglyphs in an inscription. The new forms of cryptography came soon after the widespread development of computer communications. In data and telecommunications, cryptography is necessary when communicating over any un-trusted medium, which includes just about any network, particularly the Internet, says [1], a famous scientist in cryptography. Within the context of any application-to-application communication, there are some specific security requirements, including:

a) Authentication: The process of proving one's identity. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak.)
b) Privacy/confidentiality: Ensuring that no one can read the message except the intended receiver.
c) Integrity: Assuring the receiver that the received message has not been altered in any way from the original.
d) Non-repudiation: A mechanism to prove that the sender really sent this message.

[1], argued that Cryptography not only protects data from theft or alteration but can also be used for user authentication. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions, each of which is described below. In all cases, the initial unencrypted data is referred to as plaintext. It is encrypted into ciphertext, which will in turn (usually) be decrypted into usable plaintext.

Types of Cryptographic Algorithms

There are several ways of classifying cryptographic algorithms. For the purposes of this thesis, they will be categorized based on the number of keys that are employed for encryption and decryption and further defined by their application and use. The three types of algorithms that will be discussed are (Figure 1.1):

a) Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption.
b) Public Key Cryptography (PKC): Uses one key for encryption and another for decryption.
c) Hash Functions: Uses mathematical transformation to irreversibly "encrypt" information.

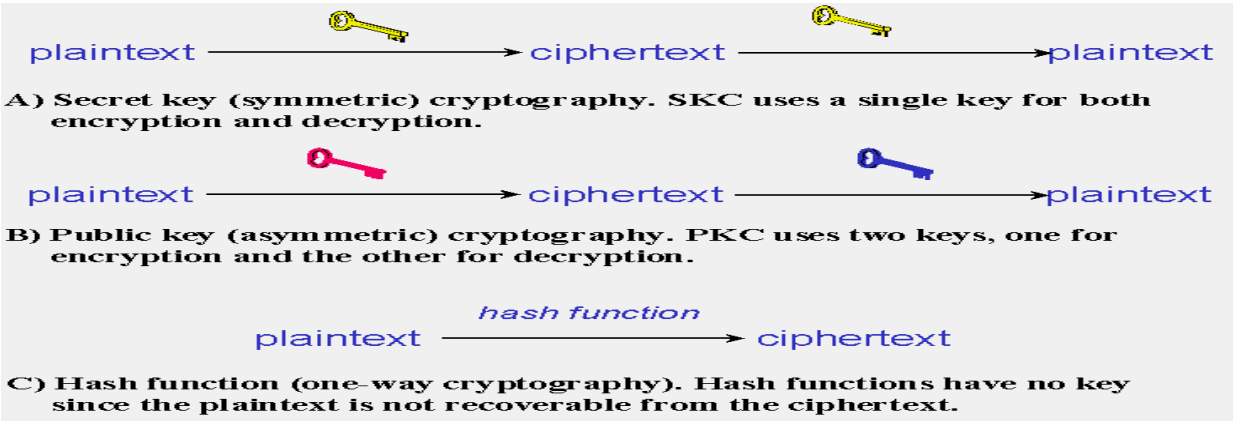

Figure 2: Three types of cryptography: secret-key, public key, and hash function.

Why Three Encryption Techniques

In one of the recent writings of [1], he stated that the answer is that each scheme is optimized for some specific application(s). Hash functions, for example, are well-suited for ensuring data integrity because any change made to

the contents of a message will result in the receiver calculating a different hash value than the one placed in the transmission by the sender. Since it is highly unlikely that two different messages will yield the same hash value, data integrity is ensured to a high degree of confidence. Secret key cryptography, on the other hand, is ideally suited to encrypting messages, thus providing privacy and confidentiality. The sender can generate a session key on a per-message basis to encrypt the message; the receiver, of course, needs the same session key to decrypt the message.Key exchange, of course, is a key application of public-key cryptography (no pun intended). Asymmetric schemes can also be used for non-repudiation and user authentication; if the receiver can obtain the session key encrypted with the sender's private key, then only this sender could have sent the message. Public-key cryptography could, theoretically, also be used to encrypt messages although this is rarely done because secret-key cryptography operates about 1000 times faster than public-key cryptography explained by [1].

The Significance of Key Length

In cryptography, size does matter. The larger the key, the harder it is to crack a block of encrypted data. The reason that large keys offer more protection is almost obvious; computers have made it easier to attack ciphertext by using brute force methods rather than by attacking the mathematics (which are generally well-known anyway). With a brute force attack, the attacker merely generates every possible key and applies it to the ciphertext. Any resulting plaintext that makes sense offers a candidate for a legitimate key [1].

Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a symmetric-key block cipher algorithm and U.S. government standard for secure and classified data encryption and decryption. In December 2001, the National Institute of Standards (NIST) approved the AES as Federal Information Processing Standards Publication (FIPS PUB) 197, which specifies application of the Rijndael algorithm to all sensitive classified data noted in their work [3].According to [3], the Advanced Encryption Standard was originally known as Rijndael. The AES has three fixed 128-bit block ciphers with cryptographic key sizes of 128, 192 and 256 bits. Key size is unlimited, whereas the block size maximum is 256 bits. The AES design is based on a substitution-permutation network (SPN) and does not use the Data.

Data Encryption Standard (DES) Feistel Network

In 1997, the NIST initiated a five-year algorithm development process to replace the DES and Triple DES. The NIST algorithm selection process facilitated open collaboration and communication and included a close review of 15 candidates. After an intense evaluation, the Rijndael design, created by two Belgian cryptographers, was the final choice.

The AES replaced the DES with new and updated features:

a) Block encryption implementation.
b) 128-bit group encryption with 128, 192 and 256-bit key lengths.
c) Symmetric algorithms require only one encryption and decryption key.
d) Data security for 20-30 years.
e) Worldwide access.
f) No royalties.
g) Easy overall implementation.

Advanced Encryption Standard Algorithm

Encryption has found a place in today's digital world, by cultivating a culture of security and privacy. When the AES Encryption algorithm succeeded the Data Encryption Standard as the global standard for encryption algorithms in 2001, it fixed many shortcomings of its predecessor. It was seen as the future for encryption in daily life applications. So far, the Advanced Encryption Standard has achieved the targets placed during its inception. And it has a long way to go.

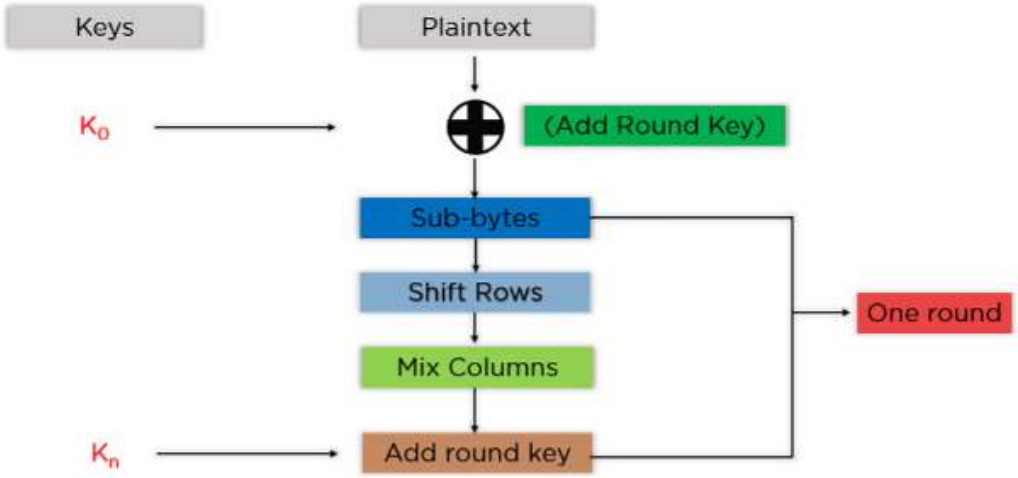

Figure 3: Advanced Encryption Standard Algorithm [4]

The AES Encryption algorithm (also known as the Rijndael algorithm) is a symmetric block cipher algorithm with a block/chunk size of 128 bits. It converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the ciphertext. It is based on a substitution-permutation network, also known as an SP network. It consists of a series of linked operations, including replacing inputs with specific outputs (substitutions) and others involving bit shuffling (permutations).

Features of Advanced Encryption Standard

There are standout features that AES offers as a globally standardized encryption algorithm.

What are the Features of AES?

a) Substitution Permutation (SP) Network: It works on an SP network structure rather than a Feistel cipher structure, as seen in the case of the DES algorithm.
b) Key Expansion: It takes a single key up during the first stage, which is later expanded to multiple keys used in individual rounds.
c) Byte Data: The AES encryption algorithm does operations on byte data instead of bit data. So, it treats the 128-bit block size as 16 bytes during the encryption procedure.
d) Key Length: The number of rounds to be carried out depends on the length of the key being used to encrypt data. The 128-bit key size has ten rounds, the 192-bit key size has 12 rounds, and the 256-bit key size has 14 rounds.

The steps in the AES encryption and decryption algorithm are described using a flowchart in Figure: 2.1 below.

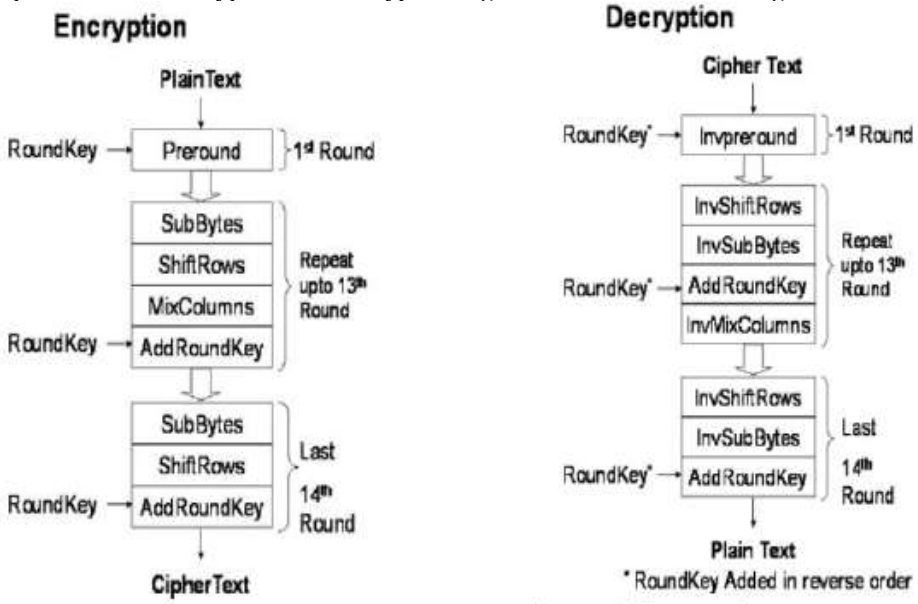

Figure 4: AES 256 encryption and decryption

Blowfish Algorithm

Blowfish is a symmetric, 64-bit block cipher with changeable length. As a "general-purpose algorithm," it was created by Bruce Schneier in 1993 as a quick, cost-free replacement for the venerable Data Encryption Standard (DES) and International Data Encryption Algorithm (IDEA) encryption techniques. Blowfish is unpatented, substantially quicker than DES and IDEA, and freely accessible for all purposes. However, because of its short block size, which is seen as unsafe, it couldn't totally replace DES. The security issue was addressed by Two fish, its successor, who used a higher block size of 128 bits. Nevertheless, many cipher suites and encryption solutions on the market today use the Blowfish algorithm since complete Blowfish has never been cracked. Blowfish features a 64-bit block size with keys that can be 32 bits long or 448 bits long. It features 16 iterations that resemble Feistel, and each one operates on a 64-bit block that is partitioned into two 32-bit words. Blowfish uses a single encryption key to encode and decode data [5].

Understanding Blowfish Algorithm

Blowfish features a 64-bit block size and takes a variable-length key, from 32 bits to 448 bits. It consists of 16 Feistel-like iterations, where each iteration operates on a 64-bit block that's split into two 32-bit words. Blowfish uses a single encryption key to both encrypt and decrypt data.

The Blowfish algorithm consists of two major parts:

1. **Data encryption.** Data encryption happens through a 16-round Feistel network, with each round consisting of a key-dependent permutation and a key- and data-dependent substitution. Large, key-dependent S-boxes work with the substitution method and form an integral part of the data encryption system in Blowfish. All encryption operations are XORs — a type of logic gate -- and additions on 32-bit words.
2. **Key expansion and subkeys.** In the key expansion process, maximum size 448-bit keys are converted into several subkey arrays totaling 4,168 bytes. Subkeys form an integral part of the Blowfish algorithm, which uses many them. These subkeys are pre-computed before encryption or decryption can take place [5].

Figure 5: Block Cipher of Blowfish Algorithm [5].

Blowfish Encryption/Decryption Process Example

Assume the message "Hi world" is to be encrypted with Blowfish. The following are the steps involved:

1. Initially, the input "Hi world" consists of seven characters plus one space, which is equal to 64 bits or 8 bytes.
2. The input is split into 32 bits. The left 32 bits -- "Hi w" -- are XORed with P1, which is generated by key expansion to create a value called P1. (**Note:** *P denotes prime number, a number that is not divisible except by 1 and itself.*)
3. Then, P1 runs through a transformative F-function (F In) in which the 32 bits are split into 4 bytes each and passed to the four S-boxes.
4. The first two values from the first two S-boxes are added to each other and XORed with the third value from the third S-box.
5. This result is added to the output of the fourth S-box to produce 32 bits as output.
6. The output of F is XORed with the right 32 bits of the input message — "world" -- to produce output F1'.
7. Then, F1' replaces the left half of the message, while P1' replaces the right half.
8. This same process is repeated for successive members of P-array for 16 rounds in total.
9. Finally, after 16 rounds, the outputs P16' and F16' are XORed with the last two entries of the P-array, i.e., P17 and P18. They are then recombined to produce the 64-bit ciphertext of the input message [5].

Two fish Algorithm

Two fish was designed to perform well on small devices that contain low-power processing capabilities. The performance advantage emerges from the key schedule, where two fish make key-dependent S-box and round-subkeys with several variations depending on the application: Long key setup for faster encryption of large plaintext data. Or a short key schedule for slower encryption processed over a larger number of encryption boxes. The encryption and decryption process can take either variation, depending on the performance capabilities of the underlying hardware and performance requirements for the applications. Two fish uses key sizes of 128, 192 and 256 bits, block size of 128 bits. The cypher is a 16-round Feistel network with a bijective F function made up of four key-dependent 8-by-8-bit S-boxes, a fixed 4-by-4 maximum distance separable matrix. [6].

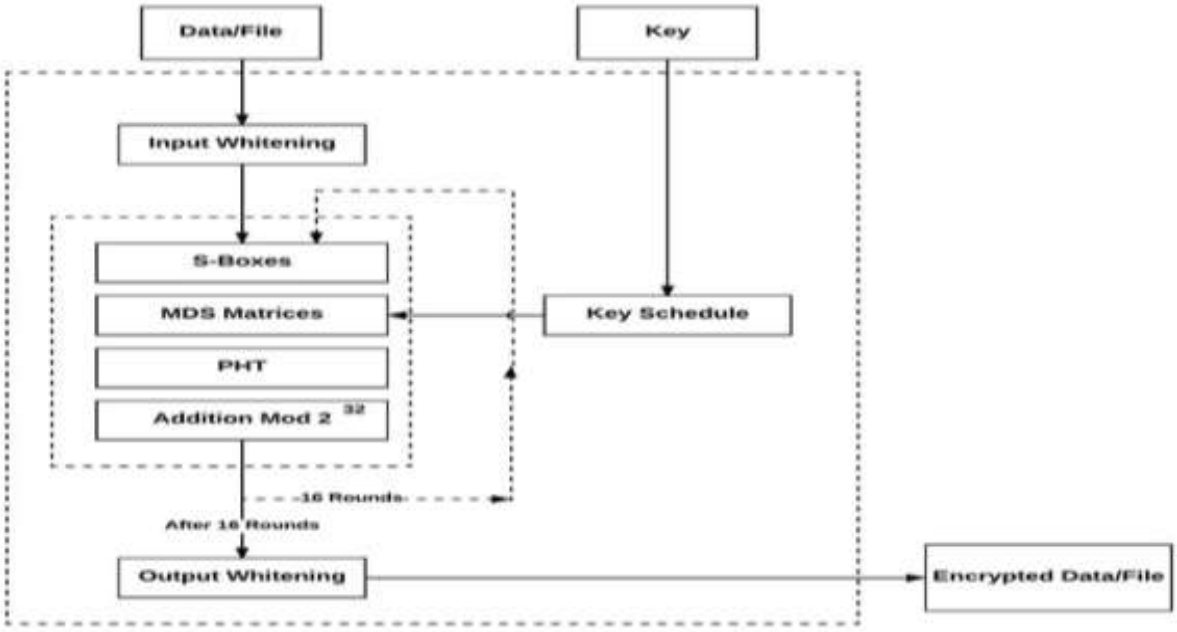

Figure 6: Building blocks of Twofish Algorithm [6]

Rijndael AES vs. Twofish Hybrid

At the NIST competition, the Rijndael algorithm was chosen as the winner over the Twofish algorithm due to two important factors. Rijndael, also known as the AES algorithm, had a simpler design operating on fixed block size with three key length choices, as opposed to Twofish using variable block sizes for a fixed key length. In terms of performance, the AES algorithm is faster and more secure. AES relies on a substitution-permutation network and a simple key schedule, whereas the Twofish algorithm using key-dependent substitution boxes follows a complex and expensive key schedule. Nevertheless, both the AES and Twofish algorithms have yet to be cracked. While some would argue that a longer key-length Twofish implementation may be more secure than a standard AES implementation, the choice comes down to ease of implementation, memory consumption and computational performance.

Ron Rivest, Adi Shamir, and Len Adleman (RSA) Algorithm

Ron Rivest, Adi Shamir, and Len Adleman published RSA cryptography in 1978. RSA means Rivest, Shamir, Adleman. These are the inventors of the popular RSA Algorithm. The RSA algorithm is based on public-key encryption technology which is a public-key cryptosystem for reliable data transmission. RSA cryptography is asymmetric cryptography that uses two keys for the process. RSA cryptography is the most popular asymmetric cryptography in use because this algorithm has a benefit for key distribution. The process of generating the key for RSA cryptography is mostly by the receiver. The receiver needs to generate both keys, a public key and a private key. The RSA public key was known by everyone and the private key kept secret by the receiver or server [7].

One Time Pad or Password (OTP)

The One Time Pad or Password is a stream cipher whose key is a random sequence of symbols from the alphabet. This algorithm is perfectly provided that the key generating process is completely random. The encryption and decryption process of the One Time Pad (OTP) is using the XOR. The properties of the XOR returns the TRUE if and only if the value of both proposals is one of those are different. Otherwise, it will return the FALSE. It has a disadvantage as follows: If the length of the key and message are different. It will be vulnerable to the attacker or hackers. If the message is a very long steam of strings, it is very expensive to implement and take much time to do the operation. Once the key is used, it will not be used again. The key can't be in a textbook or notebook. If so, it will not be an OTP. If there is only one path for transmission of the key, it may be an attacker intercepts that key

and then it may be vulnerable to him. Even though hackers have the possibility to intercept it, but till now no way to attack (decrypt). The advantage of the OTP is that it cannot be broken for ciphertext [7].

Lossless and Lossy in Encoding Techniques

Lossless and lossy encoding techniques are fundamental to data compression, each offering distinct advantages and trade-offs in terms of compression ratio, computational complexity, and resilience to errors. Lossless encoding techniques preserve the original data entirely, enabling perfect reconstruction. Common methods include Huffman coding, Run-Length Encoding (RLE), and Lempel-Ziv-Welch (LZW). These are widely used in applications where data fidelity is crucial, such as text files and executable programs. The trade-off, however, is a generally lower compression ratio compared to lossy techniques. For instance, lossless image formats like PNG typically achieve compression ratios of 2:1 to 3:1, whereas lossy formats like JPEG can compress data at ratios of 10:1 or more [8]. In contrast, lossy encoding techniques achieve significantly higher compression by permanently removing some data deemed less perceptible to human senses. This makes them ideal for multimedia applications such as audio, image, and video compression, exemplified by formats like MP3, JPEG, and MPEG. The primary trade-off is that lossy methods cannot perfectly reconstruct the original data, which can result in quality degradation, especially after multiple compression cycles. However, perceptual models are often employed to minimize noticeable losses [9].

In terms of computational complexity, both encoding types vary depending on the algorithm used. Lossless techniques like Huffman coding are relatively simple and fast but may be outperformed in compression efficiency. More advanced methods, such as arithmetic coding, offer better compression but at increased computational cost [10]. Lossy encoding, particularly modern codecs like H.265 (HEVC), employs complex prediction, transformation, and quantization steps to maximize compression while retaining quality. This added complexity requires more processing power, especially during encoding, which may be a concern for real-time applications or resource-constrained systems. Regarding resilience to errors, lossless encoding has an advantage since any corruption in the data may be easier to detect and correct due to built-in redundancy or error detection codes. Lossy formats, however, are more susceptible to quality degradation if errors occur during transmission or storage. A single bit error in a compressed lossy stream can sometimes propagate, causing noticeable artifacts or even rendering the content unusable, especially in highly compressed video streams. In summary, the choice between lossless and lossy encoding depends on the application's requirements for data fidelity, storage or bandwidth constraints, and error tolerance. A hybrid approach is often employed in practice, using lossy compression for size efficiency alongside error correction mechanisms to mitigate data loss.

Entropy-based Methods in Compression Techniques

Recent research from 2020 to 2025 highlights ongoing advancements in both classical entropy-based compression methods and emerging modern techniques. Huffman coding, despite its long-standing use for lossless data compression, is often limited by its fixed symbol-length assignments. Recent studies have sought to enhance its adaptability in dynamic data environments, improving compression efficiency without significant complexity increases [11]. Arithmetic coding remains a superior entropy coding method, offering near-optimal compression by encoding entire message probabilities as intervals, with recent improvements focusing on reducing computational overhead and enhancing error resilience [12]. Meanwhile, neural compression techniques have revolutionized data encoding by leveraging deep learning models to capture complex data distributions and optimize compression adaptively. State-of-the-art neural codecs achieve better rate-distortion performance than traditional codecs in image and video compression tasks by employing autoencoders and generative adversarial networks (GANs) [13]. These models dynamically adjust to content characteristics, enabling more efficient bit allocation.Error-correcting codes, such as Low-Density Parity-Check (LDPC) and Reed–Solomon codes, remain vital for ensuring data integrity in noisy communication channels. Recent enhancements in LDPC decoding algorithms have improved throughput and reduced latency, making them integral to 5G and emerging 6G technologies [14]. Reed–Solomon codes continue to provide robust burst error correction, with optimized decoding methods enhancing performance in storage systems and broadcasting [15]. Together, these advances in entropy coding and error correction contribute to more reliable and efficient data transmission.

Investigation into Adaptive and Context-aware Encoding Schemes

Adaptive and context-aware encoding schemes have gained significant attention in recent times due to their ability to dynamically adjust to changing data patterns and system requirements, enabling real-time optimization across various applications. Unlike static encoding methods, these schemes utilize feedback from the system or data stream to modify compression parameters on-the-fly, optimizing performance based on context such as bandwidth, latency, or content complexity. Recent studies have demonstrated that adaptive entropy coding, such as context-adaptive binary arithmetic coding (CABAC), enhances compression efficiency by tailoring symbol probability models to local data features [16]. Machine learning has further advanced these techniques. Neural networks, particularly recurrent and attention-based models, enable context-aware encoding by learning temporal and spatial dependencies within data streams. These approaches have been particularly effective in video compression, where scene dynamics dictate bitrate allocation and encoding strategies [17]. Additionally, adaptive codecs like AV1 and VVC integrate real-time

analysis modules to select encoding modes that align with network and device constraints, achieving better rate-distortion trade-offs [18]. The framework of adaptive coding and decoding of remote sensing images is shown in Figure 6.0, which consists of five parts: mode selection, feature factor extraction, adaptive shape segmentation, adaptive sampling rate allocation, and image reconstruction:

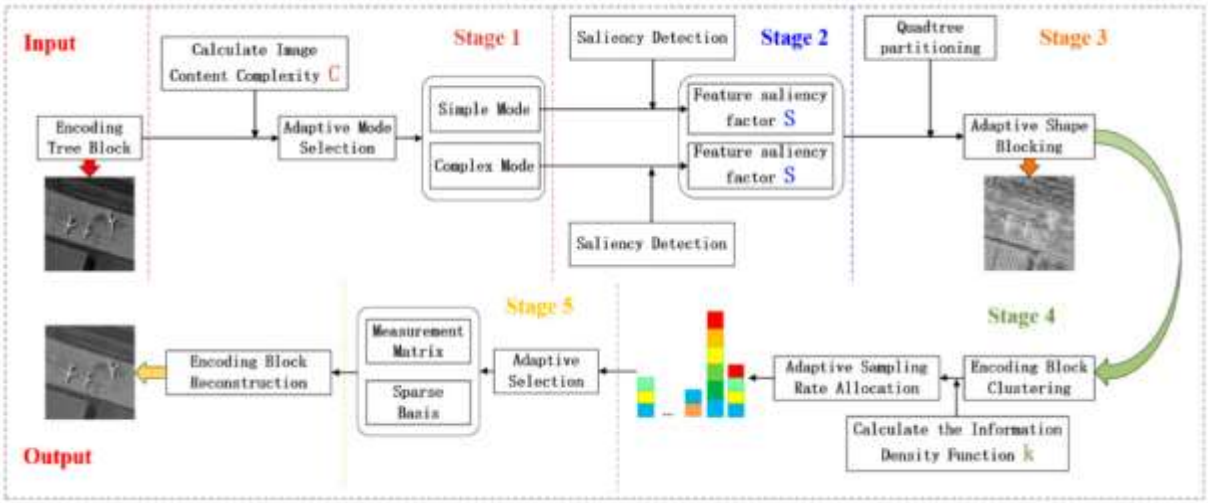

Figure 7: Framework diagram of image full process adaptive encoding and decoding based on compressive sensing [19].

This framework uses the maximum between-class variance method to calculate the image content complexity and adaptively selects the coding mode. By extracting image feature factors, a saliency model of the corresponding mode is established to guide the adaptive morphological segmentation of the coding tree block. Then, by using the information density function to detect the sparsity of the encoding block, the sparsity of the image is explored as much as possible to achieve adaptive sampling rate allocation for the image [19]. Finally, the reconstruction matrix and sparse basis are adaptively selected based on the coding block category, and the iteration threshold is set according to the known sparsity. OMP algorithm is used to reconstruct the coding block.

Overall, these adaptive encoding systems are critical for next-generation media streaming, remote sensing, and IoT applications, where real-time performance and efficient resource usage are essential.

CONCLUSION

In conclusion, while significant progress has been made in encoding and decoding strategies for next-generation system optimization, current challenges such as scalability and security remain critical barriers. As systems grow increasingly complex and data volumes expand exponentially, ensuring efficient scalability without compromising performance becomes paramount. Concurrently, security threats continue to evolve, demanding robust encryption and error-resilient decoding methods that can safeguard data integrity and confidentiality in dynamic environments. Addressing these challenges requires innovative approaches that transcend traditional computational paradigms. Future research directions should harness the transformative potential of artificial intelligence (AI) and quantum computing to revolutionize encoding and decoding processes. AI techniques, such as deep learning and reinforcement learning, can enable adaptive, context-aware algorithms capable of optimizing encoding schemes in real-time based on data patterns and channel conditions. Meanwhile, quantum computing offers unprecedented computational power and novel quantum error-correcting codes, which promise enhanced security and efficiency by exploiting quantum phenomena like superposition and entanglement. Integrating AI-driven models with quantum-enhanced frameworks could yield hybrid encoding and decoding architectures that not only scale seamlessly but also fortify systems against sophisticated attacks. This synergy represents a promising frontier, inviting interdisciplinary collaboration to develop resilient, intelligent, and scalable solutions tailored for the demands of future communication and data processing infrastructures. Ultimately, embracing these advances will be crucial for achieving optimal system performance in an increasingly connected world.

REFERENCES

1. Gary C. Kessler. (2016). An Overview of Cryptography. Retrieved from https://garykessler.net/library/crypto.html available on 26/05/2025
2. Reynolds Joshua (2019). Cryptography and Security in Banking. Retrieved from https://medium.com/@joshuareynolds/cryptography-and-security-in-banking-2cce7691e70f available on 26/05/2025

3. Dale Janssen and Cory Janssen. (2017). Advanced Encryption Standard (AES). Retrieved from https://www.techopedia.com/definition/1763/advanced-encryption-standard-aes available on 26/05/2025
4. Geeksforgeeks (2023) Why AES has Replaced DES, 3DES and TDEA? Retrieved from https://www.geeksforgeeks.org/computer-networks/why-%60-has-replaced-des-3des-and-tdea/ available on 26/05/2025
5. Sung YT, Chang KE, Liu TC. The effects of integrating mobile devices with teaching and learning on students' learning performance: A meta-analysis and research synthesis. Computers & Education. 2016 Mar 1;94:252-75.
6. Santoso K I, M A Muin, and M A Mahmudi (2020). Implementation of AES cryptography and twofish hybrid algorithms for cloud. Journal of Physics: Conference Series 1517 (2020) 012099 IOP publishing doi:10.1088/1742-6596/1517/1/012099
7. Khean Ouk, Kimsoung Lim2, Sen samnang Ouk (2020), Hybrid of asymmetric cryptography (RSA) and symmetric cryptography (OTP).
8. Sayood, K. (2017). Introduction to Data Compression. Morgan Kaufmann.
9. Salomon, D., & Motta, G. (2010). Handbook of Data Compression. Springer.
10. Witten, I. H., Neal, R. M., & Cleary, J. G. (1987). Arithmetic coding for data compression. Communications of the ACM, 30(6), 520-540.
11. Wang, L., Zhao, M., & Zhang, Y. (2021). Adaptive Huffman coding for dynamic data compression. IEEE Transactions on Multimedia, 23(6), 1654-1663.
12. Li, X., & Chen, J. (2022). Efficient arithmetic coding with improved error resilience. IEEE Transactions on Communications, 70(3), 1345-1356.
13. Singh, A., Gupta, P., & Verma, R. (2023). Neural network-based image compression: A survey and performance evaluation. IEEE Access, 11, 15098-15112.
14. Kim, S., Park, H., & Lee, J. (2024). Low-latency LDPC decoding algorithms for 6G communication systems. IEEE Communications Magazine, 62(1), 38-44.
15. Zhou, Y., & Park, D. (2021). Optimized Reed–Solomon decoding for high-speed storage applications. IEEE Transactions on Information Theory, 67(12), 7894-7902.
16. Chen, J., Xu, M., & Zhang, L. (2021). Adaptive context modeling for improved entropy coding in image compression. IEEE Transactions on Image Processing, 30, 5874–5887.
17. Park, H., Kim, Y., & Choi, J. (2023). Attention-based neural video compression for real-time streaming. IEEE Transactions on Circuits and Systems for Video Technology, 33(1), 112–123.
18. Zhang, X., & Liu, Y. (2024). Real-time adaptive encoding strategies in VVC for heterogeneous network conditions. IEEE Access, 12, 20345–20355.
19. Huiling Hu, Chunyu Liu, Shuai Liu, Shipeng Ying, Chen Wang and Yi Ding (2024) Sensing Images Based on Compression Sensing.Remote Sensing, 16(9), 1529; https://doi.org/10.3390/rs16091529

CITE AS: Osondu E. Ogbodo, Obikwelu R. Okonkwo, Godspower I. Akawuku and Chimeremeze P. Ejimadu (2025). Overview of Algorithms for Encoding and Decoding Payment Systems. IDOSR JOURNAL OF SCIENCE AND TECHNOLOGY 11(2):46-54.
https://doi.org/10.59298/IDOSR/JST/25/112.465400