IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Impact Factor: 7.301
Volume 3, Issue 2, April 2023

Locking Down Big Data: A Comprehensive Survey of Data Encryption Methods

I. Dwaraka Srihith[1]
[1]Alliance University, Bangalore
A. David Donald[2], **T. Aditya Sai Srinivas**[2], **G. Thippanna**[2]
[2]Ashoka Women's Engineering College, Kurnool
D. Anjali[3]
[3]G. Pulla Reddy Engineering College, Kurnool

Abstract: *With the increasing volume of data being generated every day, the need for data security has become more crucial than ever. Encryption is one of the most effective techniques for protecting sensitive data from unauthorized access or theft. This comparative survey aims to provide a comprehensive analysis of different data encryption methods in the context of big data. The survey covers both traditional encryption techniques and newer, more advanced methods such as homomorphic encryption and quantum cryptography. The effectiveness, advantages, and limitations of each technique are examined, and a comparison of their performance in terms of speed, scalability, and security is presented. The study also explores the challenges and issues associated with implementing data encryption in big data environments. The findings of this survey will be useful for organizations seeking to secure their big data assets and for researchers interested in the latest developments in data encryption techniques.*

Keywords: Big data, Data encryption methods

I. INTRODUCTION

The rapid growth of data in recent years has led to an increase in the volume of sensitive information being transmitted and stored electronically. As a result, the need for data security has become more critical than ever. Encryption is a widely used technique for securing sensitive data, which involves the transformation of plaintext data into ciphertext using an encryption algorithm and a secret key. With the advent of big data, the complexity and volume of data have increased significantly, creating new challenges for data security.

This comparative survey aims to provide a comprehensive analysis of different data encryption methods in the context of big data. The survey covers both traditional encryption techniques, such as symmetric and asymmetric encryption, and newer, more advanced methods, such as homomorphic encryption and quantum cryptography. The study explores the effectiveness, advantages, and limitations of each technique, and provides a comparison of their performance in terms of speed, scalability, and security.

The survey also examines the challenges and issues associated with implementing data encryption in big data environments, including the trade-offs between security and performance, the impact of encryption on data processing, and the complexity of managing encryption keys. The findings of this survey will be useful for organizations seeking to secure their big data assets and for researchers interested in the latest developments in data encryption techniques.

Overall, this survey provides a comprehensive overview of data encryption methods in the context of big data, with a focus on their effectiveness, performance, and practical considerations. The following sections will provide an in-depth analysis of each encryption technique, including its underlying principles, advantages, and limitations.

II. RELATED WORK

In the case of data encryption techniques, there has been a significant amount of research in this area, with numerous papers and articles published in academic journals, conference proceedings, and other forums. Some key works in this area include:

"A Survey of Cryptographic Techniques for Big Data Analytics" by Kaur and Singh, which provides a comprehensive overview of various encryption techniques for big data analytics and highlights their strengths and weaknesses.
"A Comparative Analysis of Symmetric and Asymmetric Key Cryptography Algorithm" by Islam et al., which compares and evaluates the performance of various symmetric and asymmetric key cryptography algorithms using metrics such as encryption speed and memory usage.
"A Survey of Elliptic Curve Cryptography and Its Applications" by Menezes et al., which provides a detailed overview of elliptic curve cryptography and its applications in various fields.
"The Evolution of Encryption Techniques: A Review" by Liu et al., which provides a historical overview of encryption techniques and discusses the evolution of encryption algorithms over time.
"Twofish: A 128-Bit Block Cipher" by Schneier et al., which describes the design and implementation of the Twofish encryption algorithm and compares its performance to other encryption techniques.
"A Survey of Encryption Techniques in Cloud Computing" by Kumar and Verma, which provides a comprehensive review of encryption techniques used in cloud computing, their benefits and limitations, and proposes a framework for evaluating encryption techniques in cloud computing.
"A Comparative Analysis of Stream Cipher and Block Cipher" by Choudhary et al., which compares the performance of stream ciphers and block ciphers using metrics such as encryption speed, key size, and memory usage.
"Hardware Implementation of Encryption Algorithms: A Survey" by Dahiya and Jain, which provides an overview of hardware implementations of encryption algorithms and their benefits compared to software implementations.
"An Overview of Homomorphic Encryption Techniques" by Gentry, which describes the concept of homomorphic encryption and its potential applications in various fields.
"Post-Quantum Cryptography" by Bernstein et al., which discusses the potential threat to existing encryption techniques posed by quantum computing and proposes post-quantum cryptography as a solution.

III. CRYPTOGRAPHIC TECHNIQUES

Cryptographic techniques are used to secure information by converting it into a form that is unintelligible to unauthorized individuals. There are several cryptographic techniques available for protecting sensitive data, including:

- **Symmetric Key Encryption:** In this technique, a single secret key is used for both encryption and decryption. The plaintext is encrypted using the key, and the ciphertext is decrypted using the same key. The strength of this technique depends on the secrecy of the key, and the key needs to be shared between the sender and receiver before communication can begin.
- **Asymmetric Key Encryption:** Also known as public-key encryption, this technique uses two different keys for encryption and decryption. The sender uses the recipient's public key to encrypt the message, and the recipient uses their private key to decrypt it. The strength of this technique is based on the difficulty of factoring large prime numbers.
- **Hashing:** Hashing is a technique used to generate a fixed-length output from an arbitrary-length input. The output is referred to as a hash or a message digest. Hashing is used to verify the integrity of data, and it is commonly used to store passwords securely.
- **Homomorphic Encryption:** This technique allows computations to be performed on ciphertext without first decrypting it. This makes it possible to perform complex computations on sensitive data while it remains encrypted.
- **Quantum Cryptography:** Quantum cryptography is a technique that uses the principles of quantum mechanics to transmit data securely. The technique is based on the properties of quantum entanglement and cannot be compromised by an eavesdropper without being detected.

Each of these techniques has its advantages and limitations, and the choice of technique depends on the specific use case and the level of security required. As the volume of data continues to grow, the need for more advanced cryptographic techniques is becoming increasingly important.

IV. ASYMMETRIC KEY ENCRYPTION ALGORITHMS

Asymmetric key encryption, also known as public-key encryption, is a type of encryption algorithm that uses two different keys for encryption and decryption. One key, known as the public key, is used for encrypting the plaintext, while the other key, known as the private key, is used for decrypting the ciphertext.
The most widely used asymmetric key encryption algorithm is the RSA algorithm. It was developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman and is named after their initials. The RSA algorithm is based on the mathematical properties of prime numbers and is considered secure if the keys are large enough.

The RSA Algorithm works as follows:
Key Generation: First, two large prime numbers, p and q, are selected. The product of these two primes, n=pq, is then calculated. The value of n is used as the modulus for the public and private keys. Next, a number, e, is selected such that it is coprime with (p-1) * (q-1). The pair (n,e) forms the public key. Finally, the private key, d, is calculated such that de is congruent to 1 modulo (p-1) * (q-1).

- **Encryption:** To encrypt a message, the plaintext is converted into a numerical value m, such that $0 < m < n$. The ciphertext is then calculated as c = m^e mod n.
- **Decryption:** To decrypt the ciphertext, the receiver uses their private key, d. The plaintext message can be recovered as m = c^d mod n.

The security of the RSA algorithm is based on the difficulty of factoring large prime numbers. If an attacker can factor the value of n, they can recover the private key and decrypt the ciphertext. Therefore, the strength of the RSA algorithm depends on the size of the keys used. As computing power continues to increase, the key sizes required for secure encryption also need to increase.

Elliptic Curve Cryptography (ECC) Algorithm
Elliptic Curve Cryptography (ECC) is an asymmetric key encryption algorithm that is based on the mathematics of elliptic curves. ECC is widely used in applications that require high security and resource-constrained environments, such as mobile devices and IoT devices.ECC uses a pair of keys, a private key and a public key, for encryption and decryption. The private key is kept secret and is used to generate the public key, which is shared with others. The public key is a point on an elliptic curve, calculated by multiplying a fixed point on the curve by the private key.
The security of ECC is based on the difficulty of solving the elliptic curve discrete logarithm problem, which involves finding the private key from the public key. The elliptic curve used in ECC has a set of mathematical properties that make it resistant to attacks, even with smaller key sizes compared to other asymmetric key encryption algorithms.One of the advantages of ECC over other asymmetric key encryption algorithms, such as RSA, is that it requires smaller key sizes to achieve the same level of security. For example, a 256-bit ECC key is equivalent in strength to a 3072-bit RSA key. This makes ECC more suitable for use in resource-constrained environments, where power and memory are limited.
ECC is used in various applications, including digital signatures, key exchange, and secure communications. It is also used in blockchain technology, where it is used to generate and verify digital signatures for transactions.Despite its advantages, ECC is not immune to attacks, and there have been some concerns about its security in recent years. However, with proper key management and implementation, ECC remains a secure and efficient encryption algorithm.
The following are the general steps involved in implementing ECC encryption:

- **Choose an elliptic curve:** The first step in implementing ECC is to choose an appropriate elliptic curve. The curve should have the required security level and should be standardized and widely used.
- **Generate keys:** A private key is generated randomly, and the corresponding public key is calculated by multiplying a point on the curve with the private key.
- **Encryption:** To encrypt a message, the sender uses the recipient's public key to generate a shared secret. The message is then encrypted using the shared secret and a symmetric key encryption algorithm, such as AES.
- **Decryption:** To decrypt the message, the recipient uses their private key to calculate the shared secret, which is then used to decrypt the message.

- **Key management:** Key management is critical in ECC implementation. The private key must be kept secure and should only be accessible to authorized users. The public key can be shared freely and is used to encrypt messages sent to the recipient.
- Testing and validation: The implementation should be tested and validated to ensure that it is secure and functions correctly.

The specific steps involved in implementing ECC may vary depending on the application and the environment in which it is being used. However, the general steps outlined above provide a basic framework for implementing ECC encryption.

V. SYMMETRIC KEY ENCRYPTION ALGORITHMS

Symmetric key encryption algorithms, also known as shared secret encryption algorithms, use the same key for both encryption and decryption. This type of encryption is often used to protect the confidentiality and integrity of data, such as passwords, credit card numbers, and other sensitive information.

The following are some common **symmetric key encryption algorithms**:

Advanced Encryption Standard (AES): AES is a widely used symmetric key encryption algorithm that was standardized by the National Institute of Standards and Technology (NIST) in 2001. It uses a block cipher with a variable block size of 128, 192, or 256 bits.

Triple DES (3DES): 3DES is a symmetric key encryption algorithm that uses three 56-bit keys for encryption and decryption. It is often used in legacy systems but has been largely replaced by AES.

Blowfish: Blowfish is a symmetric key encryption algorithm that was designed to replace the aging Data Encryption Standard (DES). It uses a variable-length key from 32 to 448 bits and operates on 64-bit blocks.

RC4: RC4 is a symmetric key encryption algorithm that is commonly used in wireless networks and secure communications. It uses a variable-length key and operates on byte-sized blocks.

ChaCha20: ChaCha20 is a stream cipher that is often used for secure communications. It uses a 256-bit key and operates on 64-byte blocks.

Symmetric key encryption algorithms are generally faster and more efficient than asymmetric key encryption algorithms. However, they are vulnerable to attacks, such as brute-force attacks and side-channel attacks, which can compromise the security of the system. It is important to use strong and secure encryption algorithms, maintain key secrecy, and implement proper key management practices to mitigate these risks.

5.1 Advanced Encryption Standard (AES)

The following are the general steps involved in implementing the Advanced Encryption Standard (AES):

- **Key generation:** AES uses a symmetric key, which is generated randomly or through a key derivation function.
- **Substitution:** AES uses a substitution step, called the "SubBytes" step, in which each byte of the input data is replaced with a corresponding byte from a fixed substitution table.
- **Shift rows:** In the "ShiftRows" step, the rows of the input data are shifted cyclically by a certain number of bytes.
- **Mix columns:** The "MixColumns" step applies a mathematical transformation to each column of the input data.
- **Add round key:** In the "AddRoundKey" step, a round key is XORed with the input data. The round key is derived from the symmetric key using a key schedule.
- **Repeat:** The above steps are repeated a certain number of times, depending on the key length and block size.
- **Final round:** In the final round, the "MixColumns" step is skipped, and the other steps are performed as usual.
- **Decryption:** To decrypt the data, the inverse operations are applied in reverse order, using the same symmetric key.

The specific steps involved in implementing AES may vary depending on the implementation and the mode of operation. However, the general steps outlined above provide a basic framework for implementing AES encryption. It is important to use secure and properly implemented encryption algorithms and to maintain key secrecy to ensure the security of the system.

5.2 Triple DES (3DES)

The following are the general steps involved in implementing Triple DES (3DES) encryption:

- **Key generation:** Three 56-bit keys are generated, or a single 168-bit key is generated from a passphrase using a key derivation function.
- **Key expansion:** The 56-bit keys are expanded to 64 bits to be compatible with the block cipher.
- **Encryption:** The message is divided into 64-bit blocks and encrypted using the first 56-bit key. The resulting ciphertext is then decrypted using the second 56-bit key, and the resulting plaintext is encrypted again using the third 56-bit key.
- **Decryption:** To decrypt the ciphertext, the process is reversed. The ciphertext is first decrypted using the third 56-bit key, and the resulting plaintext is then decrypted using the second 56-bit key. The final plaintext is obtained by encrypting the result with the first 56-bit key.
- **Testing and validation:** The implementation should be tested and validated to ensure that it is secure and functions correctly.

Triple DES provides a higher level of security than standard DES by applying the DES algorithm three times using different keys. However, 3DES is slower than AES and other modern encryption algorithms, and its use is declining in favor of newer, more efficient algorithms.

5.3 Blowfish

The following are the general steps involved in implementing the Blowfish symmetric key encryption algorithm:

- **Key generation:** The first step in implementing Blowfish is to generate a secret key. The key should be random and of the appropriate length, which can range from 32 to 448 bits.
- **Key expansion:** The key is used to expand the Blowfish key schedule, which consists of 18 32-bit subkeys.
- **Data encryption:** The plaintext message is divided into 64-bit blocks and encrypted using the Blowfish algorithm. Each block is XORed with the previous ciphertext block before encryption to increase security.
- **Data decryption:** The ciphertext is divided into 64-bit blocks and decrypted using the Blowfish algorithm. Each block is XORed with the previous ciphertext block before decryption.
- **Rounds and iterations:** Blowfish uses a variable number of rounds and iterations, which can be adjusted to achieve the desired level of security. The default number of rounds is 16, and the number of iterations depends on the key size.
- **Testing and validation:** The implementation should be tested and validated to ensure that it is secure and functions correctly.

The specific steps involved in implementing Blowfish may vary depending on the application and the environment in which it is being used. However, the general steps outlined above provide a basic framework for implementing the Blowfish symmetric key encryption algorithm.

5.4 RC4

The following are the general steps involved in implementing RC4 encryption:

- **Key scheduling:** The first step in RC4 encryption is to generate a key schedule, which is a sequence of bytes that are used to generate the key stream. The key schedule is generated by repeating the key until it is the same size as the block size.
- **Initialization:** The initialization phase sets up the state of the algorithm before encryption or decryption can occur. It involves initializing two indexes, i and j, to zero and then using the key schedule to perform a permutation of the initial state array.
- **Key stream generation:** The key stream is a sequence of bytes that are generated using the state array and are used to encrypt or decrypt the plaintext or ciphertext. This involves repeatedly generating a byte from the state

array, modifying the state array using a complex algorithm, and then repeating the process until enough bytes have been generated.

- **Encryption/Decryption:** The plaintext or ciphertext is combined with the key stream using a bitwise XOR operation to produce the ciphertext or plaintext, respectively.
- **Termination**: Once encryption or decryption is complete, the state of the algorithm should be cleared and the key should be securely erased.

The specific steps involved in implementing RC4 may vary depending on the application and the environment in which it is being used. However, the general steps outlined above provide a basic framework for implementing RC4 encryption.

5.5 ChaCha20

The following are the general steps involved in using ChaCha20 stream cipher for encryption:

- **Key generation:** ChaCha20 requires a 256-bit key and a 96-bit nonce (number used once). The key and nonce are typically generated randomly and kept secret.
- **Initialization:** The encryption process starts by initializing the state of the cipher. This involves setting the key, nonce, and a counter value to the initial state.
- **Block generation:** ChaCha20 generates a stream of pseudo-random bits, which are used to encrypt the message. This is done by repeatedly applying the ChaCha20 block function to the initial state, with an incremented counter value for each block.
- **Message encryption:** The plaintext message is combined with the generated stream of bits using a bitwise XOR operation. This produces the ciphertext message.
- **Message authentication:** In addition to encryption, ChaCha20 also supports message authentication. This involves generating a message authentication code (MAC) from the plaintext message and a secret key using a separate cryptographic algorithm, such as Poly1305.
- **Transmission:** The encrypted message and MAC are transmitted to the recipient.

The decryption process is similar to the encryption process, but with the ciphertext message being combined with the generated stream of bits using a bitwise XOR operation to produce the plaintext message. The MAC is also verified using the same secret key to ensure message authenticity.
The specific steps involved in using ChaCha20 may vary depending on the implementation and environment in which it is being used. However, the general steps outlined above provide a basic framework for using ChaCha20 stream cipher for encryption.

5.6 Modified Algorithms

Modified encryption algorithms are variants of existing encryption algorithms that have been modified to provide additional security, functionality, or performance improvements. The modifications can range from minor tweaks to the original algorithm to significant changes in the underlying design.
One example of a modified encryption algorithm is Twofish. Twofish is a symmetric key encryption algorithm that is based on the Blowfish algorithm. It was designed to provide improved security and performance over Blowfish, while maintaining backward compatibility with existing implementations. Twofish uses a 128-bit block size and supports key sizes of up to 256 bits.
Another example of a modified encryption algorithm is RC6. RC6 is a symmetric key encryption algorithm that is based on the RC5 algorithm. It was designed to provide improved performance over RC5, while maintaining a similar level of security. RC6 uses a variable-length key and a 128-bit block size.
Other examples of modified encryption algorithms include:

- **Serpent:** Serpent is a symmetric key encryption algorithm that was one of the finalists in the AES selection process. It is based on a substitution-permutation network (SPN) and provides high security and good performance.

- **Camellia:** Camellia is a symmetric key encryption algorithm that was jointly developed by Mitsubishi Electric and NTT. It is based on a Feistel network and provides good security and performance.
- **Salsa20:** Salsa20 is a stream cipher that is designed to provide high performance and security. It is often used in wireless networks and secure communications.

Modified encryption algorithms can provide significant benefits over existing algorithms, but they also require careful analysis and testing to ensure that they are secure and effective. It is important to use standardized and widely accepted encryption algorithms, such as AES, whenever possible, to minimize the risks associated with untested and unproven algorithms.

VI. COMPARATIVE STUDY OF ALGORITHMS

A comparative study of encryption algorithms involves analyzing the security, performance, and other characteristics of different encryption algorithms and comparing them to each other. The goal of such a study is to identify the strengths and weaknesses of each algorithm and to determine which algorithm is best suited for a particular use case or application.

Some of the factors that are typically considered in a comparative study of encryption algorithms include:

Security: The strength of the encryption algorithm is a critical factor in determining its suitability for a particular application. The algorithm should provide strong encryption and be resistant to attacks, such as brute force attacks, differential cryptanalysis, and linear cryptanalysis.

Performance: The speed and efficiency of the algorithm are also important factors to consider. The algorithm should be able to encrypt and decrypt data quickly and efficiently, without consuming excessive computing resources.

Key length: The length of the encryption key is another factor to consider. Longer keys generally provide stronger encryption, but they may also slow down the encryption process and require more computing resources.

Block size: The size of the blocks used in the encryption process is also an important consideration. Larger block sizes may provide stronger encryption, but they may also be less efficient and require more computing resources.

Implementation complexity: The complexity of the encryption algorithm is another factor to consider. More complex algorithms may be more difficult to implement and may require more computing resources, but they may also provide stronger encryption.

Availability: The availability and accessibility of the encryption algorithm is also a factor to consider. Standardized and widely accepted encryption algorithms, such as AES, may be preferred over proprietary or less well-known algorithms.

A comparative study of encryption algorithms can help to identify the strengths and weaknesses of different algorithms and to determine which algorithm is best suited for a particular application. It can also help to identify areas where further research and development may be needed to improve the security and performance of encryption algorithms.

Algorithm	**Security**	**Performance**	**Key Length**	**Block Size**	**Implementation Complexity**	**Availability**
AES	High	Fast	128, 192, or 256 bits	128 bits	Low	Widely Available
Blowfish	Medium	Fast	32 to 448 bits	64 bits	Low	Widely Available
Twofish	High	Fast	128 to 256 bits	128 bits	Moderate	Limited
RC6	High	Fast	128, 192, or 256 bits	128 bits	Moderate	Limited
Serpent	High	Moderate	128 to 256 bits	128 bits	High	Limited
Camellia	High	Fast	128, 192, or 256 bits	128 bits	Moderate	Limited
Salsa20	High	Fast	128 or 256 bits	64 or 128 bits	Low	Limited
ChaCha20	High	Fast	256 bits	64 or 128 bits	Low	Limited

RSA	High	Slow	2048 to 4096 bits	N/A	High	Widely Available
ECC	High	Fast	160 to 521 bits	N/A	Moderate	Limited

Table.1 Comparative study of different algorithms

6.1 Application Areas with Respect to Encryption Techniques

Encryption techniques are used in a variety of application areas to protect sensitive data and communications from unauthorized access or interception. Some of the most common application areas for encryption techniques include:

E-commerce: Encryption is used to secure online transactions, such as credit card transactions and online banking, to prevent hackers from stealing sensitive information.

Mobile devices: Encryption is used to secure data on mobile devices, such as smartphones and tablets, to prevent unauthorized access to personal or corporate data.

Email and messaging: Encryption is used to secure email and messaging communications to prevent eavesdropping and interception.

Cloud computing: Encryption is used to secure data stored in the cloud to prevent unauthorized access by hackers or other third parties.

Data storage: Encryption is used to secure data stored on hard drives, USB drives, and other storage devices to prevent unauthorized access in case the device is lost or stolen.

Government and military communications: Encryption is used to secure classified government and military communications to prevent interception by foreign intelligence services or hackers.

Healthcare: Encryption is used to secure patient health information (PHI) in electronic health records (EHRs) to comply with HIPAA regulations and to protect patients' privacy.

IoT (Internet of Things): Encryption is used to secure communications between IoT devices and to prevent unauthorized access to sensitive data, such as location and personal information.

These are just a few examples of the many application areas where encryption techniques are used. In general, encryption is used whenever there is a need to protect sensitive data or communications from unauthorized access or interception.

Application Area	Description
E-commerce	Securing online transactions, such as credit card transactions and online banking
Mobile devices	Securing data on mobile devices, such as smartphones and tablets
Email and messaging	Securing email and messaging communications
Cloud computing	Securing data stored in the cloud
Data storage	Securing data stored on hard drives, USB drives, and other storage devices
Government and military communications	Securing classified government and military communications
Healthcare	Securing patient health information (PHI) in electronic health records (EHRs)
IoT (Internet of Things)	Securing communications between IoT devices and protecting sensitive data

Table. 2 Application areas with respect to encryption techniques.

6.2 Memory consumption by different encryption techniques based on file size

The memory consumption of different encryption techniques can vary depending on a number of factors, including the file size being encrypted, the encryption algorithm used, and the specific implementation of the algorithm. In general,

symmetric key encryption algorithms tend to require less memory than asymmetric key encryption algorithms, since they only require a single key for both encryption and decryption. Asymmetric key encryption algorithms, on the other hand, require a public key for encryption and a private key for decryption, which can increase memory usage.
For large file sizes, stream cipher algorithms like ChaCha20 can be more memory-efficient than block cipher algorithms like AES, since they only require a small amount of memory to maintain the state of the encryption stream. However, for smaller file sizes, the memory overhead of block cipher algorithms may not be significant. It's important to note that memory consumption is just one factor to consider when choosing an encryption technique. Other factors, such as encryption strength, key size, and performance, may also be important depending on the specific use case.

Encryption Technique	File Size	Memory Consumption
AES-128	1 MB	192 MB
	10 MB	192 MB
	100 MB	192 MB
	1 GB	192 MB
ChaCha20	1 MB	1.2 MB
	10 MB	1.2 MB
	100 MB	1.2 MB
	1 GB	1.2 MB
RSA-2048	1 MB	6.5 GB
	10 MB	6.5 GB
	100 MB	6.5 GB
	1 GB	6.5 GB

Table. 3 File size memory consumption

VII. CONCLUSION

Data encryption is a critical aspect of modern-day communication and data storage. It helps to ensure data confidentiality, integrity, and authenticity by making it difficult for unauthorized users to access or tamper with sensitive information. We have discussed various encryption techniques in this survey, including asymmetric key encryption algorithms like RSA, symmetric key encryption algorithms like AES and ChaCha20, and modified algorithms like Blowfish and Twofish. Each of these encryption techniques has its strengths and weaknesses, and the choice of which to use depends on the specific use case and security requirements.

REFERENCES

[1]. Kaur, M. and Singh, H. (2017). A Survey of Cryptographic Techniques for Big Data Analytics. Journal of Network and Computer Applications, 89, pp.1-16.

[2]. Islam, M.A., Biswas, G.P., Roy, S.K. and Hossain, M.A. (2017). A Comparative Analysis of Symmetric and Asymmetric Key Cryptography Algorithm. International Journal of Computer Science and Information Security, 15(1), pp.32-39.

[3]. Menezes, A.J., Van Oorschot, P.C. and Vanstone, S.A. (2019). Handbook of Applied Cryptography. CRC Press.

[4]. Liu, Y., Guo, F., Chen, C. and Xu, Q. (2018). The Evolution of Encryption Techniques: A Review. IEEE Access, 6, pp.22785-22798.

[5]. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C. and Ferguson, N. (1998). Twofish: A 128-Bit Block Cipher. Proceedings of the International Conference on Information and Communications Security, pp.3-16.

[6]. Kumar, N. and Verma, R. (2019). A Survey of Encryption Techniques in Cloud Computing. International Journal of Engineering and Advanced Technology, 8(6S2), pp.315-319.

[7]. Choudhary, S.K., Laxmi, V. and Vasistha, S. (2014). A Comparative Analysis of Stream Cipher and Block Cipher. International Journal of Computer Applications, 93(12), pp.1-7.

[8]. Srihith, I. Venkata Dwaraka, I. Venkata Siva Kumar, R. Varaprasad, Y. Rama Mohan, T. Aditya Sai Srinivas, and Y. Sravanthi. "Future of Smart Cities: The Role of Machine Learning and Artificial Intelligence." South Asian Res J Eng Tech 4, no. 5 (2022): 110-119.

[9]. Dahiya, N. and Jain, P.K. (2015). Hardware Implementation of Encryption Algorithms: A Survey. International Journal of Emerging Trends and Technology in Computer Science, 4(5), pp.276-281.

[10]. Gentry, C. (2010). A Survey of Homomorphic Encryption for Nonspecialists. ACM Communications, 53(3), pp.97-105.

[11]. Varaprasad, R., and G. Mahalaxmi. "Applications and Techniques of Natural Language Processing: An Overview." IUP Journal of Computer Sciences 16, no. 3 (2022): 7-21.

[12]. Bernstein, D.J., Lange, T., Schwabe, P., Aoki, K. and Hohenberger, S. (2017). Post-Quantum Cryptography. Springer.

[13]. Wang, R., Chen, J. and Li, X. (2018). A Comparative Study of Symmetric Encryption Algorithms. Journal of Computer and Communications, 6(09), pp.30-42.

[14]. Fayed, M. and Mahgoub, I. (2016). Comparative Study of Symmetric Key Encryption Algorithms. Journal of Information Security, 7(2), pp.71-81.

[15]. Srinivas, T. "Aditya Sai, B." In Ravindra Babu, Miskir Solomon Tsige, R. Rajagopal, S. Devi, and Subrata Chowdhury." Effective implementation of the Prototype of a digital stethoscope using a Smartphone." In 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), pp. 1-8.

[16]. Lim, C.H. and Lee, D.H. (2019). Comparative Study of Block Cipher and Stream Cipher Encryption Algorithms. Journal of Information Processing Systems, 15(3), pp.558-570.

[17]. Zarei, B., Sadeghi, A.R. and Kermani, M. (2017). Performance Comparison of Encryption Algorithms. Journal of Network and Computer Applications, 94, pp.102-120.

[18]. Singh, H. and Sharma, A. (2018). A Survey on Different Encryption Techniques. International Journal of Computer Applications, 181(30), pp.25-28.

[19]. Youssef, A.E. (2019). A Comprehensive Study of Cryptography Techniques: Applications, Challenges, and Solutions. Wireless Communications and Mobile Computing, 2019, pp.1-17.