Graduate College Journal – NU
Vol.19 -2024, No. (11)
ISSN: 1858-6228, http://www.neelain .edu.sd

مجلة الدراسات العليا - جامعة النيلين
المجلد (19) –2024، العدد (11)
الرقم الدولي الموحد للدوريات: 1858-6228

Analyzing the Efficiency and Complexity of Cryptography Algorithm

A Focus on Elliptic Curve Diffie-Hellman (ECDH) Hybrid with Artificial Neural Networks (ANN)

Wafa Aref Ahmed Ibrahim[1*], Fakhreddin Abbas[2], and Fakhereldeen E.E Musa[1]

[1]Department of Computer Science, Faculty of Computer Science and Information Technology, Al-Neelain University, Khartoum, Sudan
[2]Department of Statistics, Faculty of Mathematical Sciences and Statistics, Al-Neelain University, Khartoum, Sudan.
[*]Corresponding author email: wafa.2011@live.com

Abstract

Ensuring data security is crucial in today's digital environment, where information exchange happens rapidly and continuously. Encryption plays a vital role in safeguarding sensitive data from unauthorized access and breaches. This study focuses on analyzing the encryption algorithm using hybrid elliptic curve Diffie-Hellman (ECDH) with Artificial Neural Networks (ANN), known as ECDH_ANN. Multiple input scenarios were evaluated, measuring memory complexities, operational requirements, and efficiency metrics to determine algorithm effectiveness. The primary challenge lies in improving encryption algorithms, particularly elliptic curves, and studying their complexities and performance. The ultimate goal is to measure efficiency and calculate complexities through evaluating various input scenarios, estimating execution time, memory usage, and optimizing encryption and decryption processes. This study was conducted across 50 different-sized files. The results show that as the file size grows, the encryption and decryption times also rise, while memory usage stays relatively constant, indicating efficient resource management. The algorithm maintains consistent file sizes during encryption and decryption processes, distinguishing it from algorithms that may inflate file sizes. The study also demonstrates that encryption and decryption operations exhibit linear growth rates. Overall, the ECDH_ANN algorithm stands out for its ability to maintain data integrity and use computational resources efficiently, making it perfectly suited for environments prioritizing data security and computational efficiency. The study recommends using this algorithm due to its quality and suggests comparing it with other algorithms for further analysis.

Key words: Encryption, Data Security, Cryptographic Algorithms, ECDH Hybrid, Artificial Neural Networks, Computational Efficiency, Data Integrity.

Introduction

In the contemporary digital landscape, the escalating volume of sensitive data exchanged across networks underscores the paramount importance of encryption mechanisms in ensuring information security. Encryption stands as a fundamental pillar in safeguarding data integrity and confidentiality, mitigating the risks posed by unauthorized access and malicious intrusions. As cyber, threats continue to evolve in sophistication and frequency, the adoption of robust encryption methodologies becomes imperative to fortify the resilience of digital infrastructures. Artificial neural networks (ANNs) have emerged as a transformative tool in augmenting the capabilities of cryptographic algorithms, amplifying their efficacy and resilience against adversarial attacks. ANNs, inspired by the complex interconnected structure of the human brain, offer unparalleled computational power and adaptability, enabling innovative approaches to encryption and decryption processes. Through iterative learning and pattern recognition, ANNs empower cryptographic systems to enhance their ability to detect anomalies, resist cryptanalysis, and optimize resource utilization.

The primary objective of this research is to analyze the complexities and efficiency metrics inherent in cryptographic algorithms, with a particular focus on the integration of artificial neural networks within the Elliptic Curve Diffie-Hellman (ECDH) hybrid algorithm. The study seeks to ascertain the efficacy of the ECDH_ANN algorithm in preserving data integrity and computational efficiency across diverse operational scenarios. Additionally, the research endeavors to validate hypotheses regarding the scalability, accuracy, and resilience of the ECDH_ANN algorithm compared to conventional encryption methodologies.

Structured into distinct sections, this paper unfolds as follows: First, an overview of the increasing significance of encryption in the digital age presented, contextualizing the necessity for advanced cryptographic techniques. Subsequently, the role of artificial neural networks in enhancing cryptographic algorithms elucidated, highlighting their transformative impact on encryption methodologies. Following this, the research objectives and hypotheses delineated, providing a clear roadmap for the ensuing analysis and evaluation. Finally, the structure of the paper outlined, delineating the thematic organization of subsequent sections and the flow of information therein. Through this structured approach, this paper endeavors to contribute nuanced insights into the realm of cryptographic algorithms, underscoring the pivotal role of artificial neural networks in fortifying data security in the digital age.

Literature Review:

This literature review examines the integration of Elliptic Curve Diffie-Hellman (ECDH) with artificial neural networks (ANNs) in cryptographic systems. The Diffie-Hellman key exchange (DHKE) introduced asymmetric encryption but lacks authentication mechanisms, leaving it vulnerable to man-in-the-middle attacks. The ECDH protocol addresses these issues by employing elliptic curve cryptography, enhancing both security and efficiency.

Artificial neural networks (ANNs) inspired by the human brain's structure and functioning, bring unique capabilities in pattern recognition, optimization, and learning. They have been explored in encryption and decryption processes to improve security and computational efficiency. Various methodologies have been employed in relevant studies to evaluate cryptographic algorithms. For instance, Othman Alesawy and Ravie Chandren Muniyandi (2016) conducted a study on secure data transmission over private clouds using ECDH combined with ANN and genetic algorithms, showing enhancements in time efficiency, performance, and accuracy (Alesawy & Muniyandi, 2016, p. 80). Additionally, Aws Naser et al. (2016) proposed a conceptual model integrating ECDH with ANNs for cloud computing, emphasizing practical applications of neural networks in cryptographic systems (Naser & Zolkipli, 2016). DUAN et al. (2020) introduced a novel image steganography method that integrates Image Elliptic Curve Cryptography (ECC) with Deep Neural Networks (DNN). This method prioritizes secure embedding, payload capacity, and human-perceivable image quality. It involves Discrete Cosine Transform (DCT) for the secret image, ECC encryption, and SegNet DNN to increase steganography capacity. Experimental results demonstrated efficient pixel allocation, high steganography capacity, and improved image quality metrics (PSNR and SSIM), effectively concealing data and enhancing image quality (Duan et al., 2020). However, existing studies often face challenges related to scalability, computational complexity, and real-world applicability. This review aims to address these gaps by evaluating the complexities and efficiency of the ECDH hybrid algorithm using artificial neural networks. It seeks to identify optimal encryption solutions that balance security and computational efficiency through comprehensive analyses of various input scenarios and memory complexities.

Moreover, the study aims to provide insights into the scalability, accuracy, and efficiency of the ECDH hybrid algorithm under diverse conditions, aiming to contribute to a deeper understanding of its performance in cryptographic applications.

Methodology

The research design and methodology employed in this study aim to comprehensive analyze the complexities and efficiency of cryptographic algorithms, specifically focusing on the Elliptic Curve Diffie-Hellman (ECDH) hybrid algorithm using artificial neural networks (ANNs). The methodology encompasses data collection methods, analysis techniques, and criteria for measuring algorithm quality and performance see Fig(1).

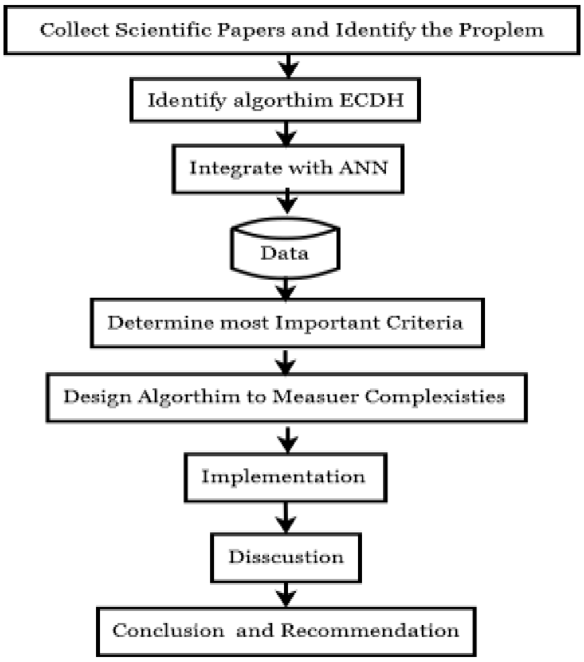
Collect Scientific Papers and Identify the Proplem
Identify algorthim ECDH
Integrate with ANN
Data
Determine most Important Criteria
Design Algorthim to Measuer Complexisties
Implementation
Disscustion
Conclusion and Recommendation

Figure (1): Methodology

Data Collection:

Data collection for this study involves gathering information on various input scenarios, memory complexities, operational requirements, and efficiency metrics related to the ECDH hybrid algorithm. Primary data sources include cryptographic literature, research papers, and technical documentation detailing algorithm specifications and performance metrics. Additionally, empirical data collected through simulations, experiments, or real-world implementations to validate algorithmic performance under diverse conditions. We put the texts we want to encrypt in text files of different sizes. Each file will be duplicated several times, where we will get 10 files of different sizes. Each file will be repeated five times with different texts, which means we have 50 files in total. We will name the files in ascending order according to their size, where we have very small files, small files, medium files, large files and very large files. For example, if we assume that we have a very small file known as "F1", we will create five other files of the same size but with different contents, and their labels become "F11", "F12", "F13", "F14", "F15". We will repeat this process with the rest of the files to obtain 50 files, where we have groups of five files of the same size but different contents. File sizes were graded and include very small files, small files, medium files, large files, and very large files, based on previous studies and their recommendations. In this way, we will be able to study the changes that may occur within files of the same size but with different contents, in order to understand the performance of the algorithms and evaluate them based on the specified criteria. The data is laid out and replicated in the following table:

Table (1): Data Structure:

#	File	Scale	Frequency				
1	F1	Name	F11	F12	F13	F14	F15
		Size	10	10	10	10	10
2	F2	Name	F21	F22	F23	F24	F25
		Size	20	20	20	20	20
3	F3	Name	F31	F32	F33	F34	F35
		Size	30	30	30	30	30
4	F4	Name	F41	F42	F43	F44	F45
		Size	40	40	40	40	40
5	F5	Name	F51	F52	F53	F54	F55
		Size	50	50	50	50	50
6	F6	Name	F61	F62	F63	F64	F65
		Size	60	60	60	60	60
7	F7	Name	F71	F72	F73	F74	F75
		Size	70	70	70	70	70
8	F8	Name	F81	F82	F83	F84	F85
		Size	80	80	80	80	80
9	F9	Name	F91	F92	F93	F94	F95
		Size	90	90	90	90	90
10	F10	Name	F101	F102	F103	F104	F105
		Size	100	100	100	100	100

Analysis Techniques:

The analysis of cryptographic algorithm complexities and efficiency entails several steps aimed at evaluating algorithmic performance across different criteria. These steps include:

1. Identification of Key Parameters: Define the key parameters relevant to the analysis, such as encryption and decryption times, memory sizes, file sizes, computational resources, and optimization results.

2. Evaluation of Algorithm Performance: Assess the performance of the ECDH hybrid algorithm using ANNs across various input scenarios. This involves conducting simulations or experiments to measure encryption and decryption times, memory usage, and computational efficiency under different conditions.

3. Comparative Analysis: Compare the performance of the ECDH hybrid algorithm 4. Criteria for Measuring Algorithm Quality: Define criteria for measuring algorithm quality and performance, including efficiency, accuracy, complexity, and scalability, security, and execution time. These criteria serve as benchmarks for evaluating the effectiveness of the ECDH hybrid algorithm in real-world scenarios.

Tools and Software:

Data analysis and visualization are essential components of this study, requiring specialized tools and software for processing and interpreting results. Commonly used tools include MATLAB, Python with libraries such as NumPy and Pandas, R, and statistical software packages. These tools facilitate data analysis, statistical modeling, visualization of results, and generation of insights from empirical data.

Furthermore, visualization techniques such as charts, graphs, and plots are employed to present findings effectively and facilitate understanding. Visualization tools like Matplotlib, Seaborn, and Tableau utilized to create visual representations of algorithmic performance metrics, aiding in the interpretation and communication of results to stakeholders and researchers.

Algorithm Description:

The ECDH (Elliptic Curve Diffie-Hellman) hybrid algorithm represents a sophisticated cryptographic technique that combines the principles of elliptic curve cryptography with the computational power of artificial neural networks (ANNs). This section provides a detailed explanation of the ECDH hybrid algorithm, highlighting its integration with ANNs, neural network architecture, parameters, and the cryptographic principles underlying its operation.

The ECDH hybrid algorithm based on the Diffie-Hellman key exchange protocol, which enables two parties to securely establish a shared secret key over an insecure channel. However, unlike the traditional Diffie-Hellman protocol, which operates in a finite field, the ECDH hybrid algorithm leverages elliptic curve cryptography for enhanced security and efficiency.

Integration with Artificial Neural Networks:

In the ECDH hybrid algorithm, artificial neural networks employed to optimize key generation, encryption, and decryption processes. The neural network architecture consists of multiple layers, including an input layer, hidden layers, and an output layer. The input layer configured to accept input data, such as plaintext or cipher text, while the output layer produces the final encrypted or decrypted output see Fig(2). The hidden layers contain neurons responsible for processing input data and extracting meaningful features relevant to the encryption process.

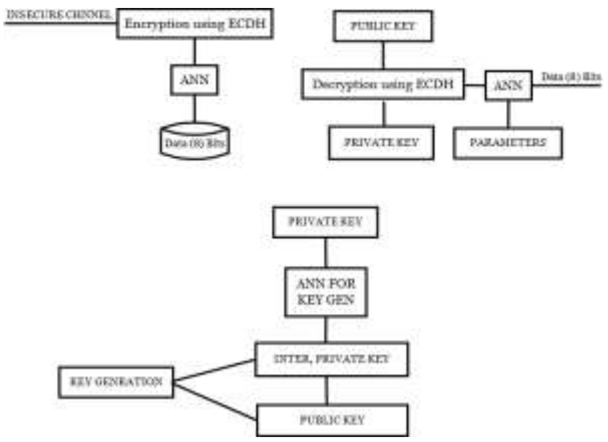

Figure (2):Hybrid Encryption,Decryption and Key Generation using ECDH and ANN

Source: Prepared by Author using Drow.io Programme, (2024).

The Diffie-Hellman Parameters:

The Diffie-Hellman key exchange (DHKE), proposed by Whitfield Diffie and Martin Hellman in 1976, is the first asymmetric cryptographic protocol published openly. It enables two parties, usually named Alice and Bob, to establish a shared secret key over an insecure channel. DHKE operates in a prime field (Zp), utilizing exponentiation, which is computationally easy to compute but difficult to reverse. Both parties agree on domain parameters, including a prime number (p) and a generator (α), which are publicly known. Each party selects a private key (a and b) and computes their respective public keys (A and B) by exponentiating the generator α to their private keys modulo p. These public keys are exchanged over the insecure channel.

Using the received public key, each party computes the shared secret key by raising it to their private key modulo p (Paar & Pelzl, 1998). This shared key enables secure communication using symmetric encryption algorithms. DHKE's security relies on the computational difficulty of the discrete logarithm problem. Attackers, with access only to the public parameters and keys, would find it challenging to derive the private keys and the shared secret key. However, the basic DHKE version is vulnerable to man-in-the-middle attacks and lacks authentication. Additional measures like digital signatures are necessary for ensuring integrity and authenticity in practical implementations. The basic idea behind the DHKE is that exponentiation in Z_p, p prime, is a one-way function and that exponentiation is commutative, i.e., (Paar & Pelzl, 1998; Singh & Aartinandal, 2020)

$$k = (a^x)^y \equiv (a^y)^x \bmod p \qquad (1)$$

The value $k \equiv (ax)^y \equiv (ay)^x \bmod p$ is the joint secret, which can be used as the session key between the two parties. Let us now consider how the Diffie–Hellman key exchanges protocol over Z_p works. In this protocol, we have two parties, Alice and Bob, who would like to establish a shared secret key. There is possibly a trusted third party that properly chooses the public parameters, which need for the key exchange. However, it is also possible that Alice or Bob generate the public parameters. Strictly speaking, the DHKE consists of two protocols, the set-up protocol and the main protocol, which performs the actual key exchange. The set-up protocol consists of the following steps: Diffie–Hellman Set-up : (Paar & Pelzl, 1998; Singh & Aartinandal, 2020)

1. Choose a large prime p.
2. Choose an integer a $\in \{2, 3, \ldots, p-2\}$.
3. Publish p and a.

These two values are sometimes referred to as *domain parameters*. If Alice and Bob both know the public parameters p and a computed in the set-up phase, they can generate a joint secret key *k* with the following key-exchange protoco(Paar & Pelzl, 1998)l:

Alice		**Bob**
choose $a = k_{pr,A} \in \{2, \ldots, p-2\}$		choose $b = k_{pr,B} \in \{2, \ldots, p-2\}$
compute $A = k_{pub,A} \equiv a^a \bmod p$		compute $B = k_{pub,B} \equiv a^b \bmod p$
	$\xrightarrow{k_{pub,A} = \mathrm{A}}$	
	$\xleftarrow{k_{pub,B} = \mathrm{B}}$	
$k_{AB} =_{k_{pub,B}}^{k_{pr,A}} \equiv B^a mod p$		$k_{AB} =_{k_{pub,A}}^{k_{pr,B}} \equiv A^b mod p$

Neural Network Parameters:

The neural network parameters, including the number of layers, the number of neurons in each layer, and activation functions, carefully tuned to optimize algorithm performance see Fig (3). Common activation functions used in the ECDH hybrid algorithm include sigmoid, tanh, and ReLU, depending on the specific requirements of the encryption task. Additionally, training algorithms such as backpropagation and gradient descent employed to iteratively adjust the neural network weights and biases to minimize prediction errors.

Neural networks have been used in cryptography operations because of their many advantages. Several algorithms have been proposed that rely on different types of neural networks to carry out the process of encryption and decryption. Among these algorithms, the backward backpropagation network is cited as an example. In the process of encryption using neural network, the steps can be summarized as follows (Singh and Singh (2012), Singh and Aartinandal (2020), Chakraborty (2010), and Al-nima, Muhanad, and Hassan (2009):

1. Enter the text to be encrypted.
2. Convert text to ASCII CODE representation.
3. Create a neural network consisting of an input layer, an output layer, and a hidden layer. The number of neurons in the input layer is equal to the size of the text to be encoded, the number of neurons in the hidden layer is specified (k), and the number of neurons in the output layer is equal to the size of the ciphertext.
4. Generate weights between the input layer and the hidden layer, and between the hidden layer and the output layer.
5. Generate thresholds for the hidden layer.
6. Train the neural network by calculating the error rate, updating weights, and the spread of error across the network.
7. Calculate the output layer using trained weights.
8. Convert ASCII CODE values to the appropriate characters.
9. Encode text using calculated results.

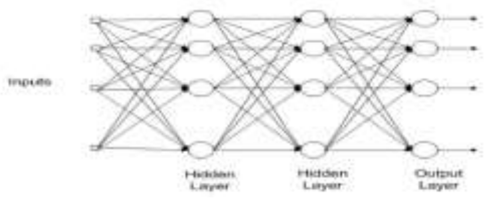

Figure (3): Artificial Neural Networks (ANN) ([
Singh, P., & Singh, H. (2012))

Criteria of Measuring the Complexities for Cryptographic Algorithms:

The efficiency measurement criteria for different encryption algorithms include algorithm execution time, file size, space complexities, number of operations, time complexity, better, medium, and worst cases, execution cost, effort expended, optimization, and algorithm efficiency. These metrics help determine the time required to execute the algorithm, the size of the file after encryption, the computational complexity of the algorithm, the best, worst, and average state of the algorithm's performance, execution cost, the amount of effort expended, and the efficiency of the algorithm (Laskari et al., 2007). The execution time is determined by multiplying the execution time taken by the number of processors used, while the amount of effort expended is determined by determining the main fundamental operation of the problem and calculating the number of basic operations performed by the algorithm . The efficiency of an algorithm calculated by multiplying the number of processors used to accomplish a particular operation by the time required to perform that operation. (Cormen et al., 1990)

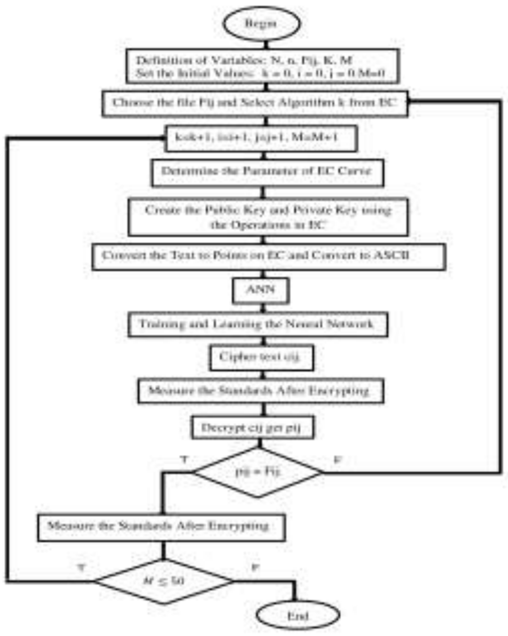
Begin
Definition of Variables: N, n, Pij, K, M
Set the Initial Values: k = 0, i = 0, j = 0 M=0
Choose the file Pij and Select Algorithm k from EC
k=k+1, i=i+1, j=j+1, M=M+1
Determine the Parameter of EC Curve
Create the Public Key and Private Key using the Operations in EC
Convert the Text to Points on EC and Convert to ASCII
ANN
Training and Learning the Neural Network
Cipher text cij
Measure the Standards After Encrypting
Decrypt cij get pij
T
pij = Pij
F
Measure the Standards After Encrypting
T
M ≤ 50
F
End

Figure (4): An Algorithm to Measure Efficiency of Encryption ECDH using ANN

The Results of Execution:

Implementing and evaluating encryption and decryption algorithms is vital in information security. This process assesses the performance of the ECDH_ANN algorithm by applying it to test files and measuring execution time, memory consumption, and complexity. Execution time depends on data volume, algorithm complexity, and system capacity, measured in seconds. Algorithm growth rate, represented by various functions, estimates efficiency and performance. Complexity analysis categorizes cases into best, worst, and average, studying memory usage and its impact on performance. File size affects memory usage and performance, aiding in optimization. Effort measures the work needed to solve problems with the algorithm, determining efficiency for files of different sizes. Optimization evaluates best, worst, and average cases to improve performance. The number of operations, including basic and iterative calculations, impacts software efficiency, aiming to reduce steps for better performance and resource use .

Table (2): Performance Evaluation of ECDH Hybrid Algorithm with ANN in Cryptographic Systems: Encryption and Decryption for Repeated Files (F1 - F10)

File	Operation	Time excute(s)	Worst cases(s)	Average cases(s)	Best cases(s)	Memory size (byte)	File size (byte)	Submitted Effort(s)
F1	Encryption	0.214640	0.267039	0.21464	0.200457	303883.4	1024	0.11588
	Decryption	0.223362	0.272167	0.223362	0.209113	246678	1024	0.11416
F2	Encryption	0.324170	0.336126	0.324170	0.308056	550261.2	2048	0.19082
	Decryption	0.350241	0.361116	0.350241	0.344511	435836.4	2048	0.18958
F3	Encryption	0.441622	0.449097	0.441622	0.431756	795007.8	3072	0.27660
	Decryption	0.481054	0.493680	0.481054	0.468236	623562.8	3072	0.26626
F4	Encryption	0.619900	0.674048	0.619980	0.586551	1038807	4096	0.32416
	Decryption	0.796081	0.813936	0.796081	0.775069	812018.8	4096	0.57796
F5	Encryption	1.036422	1.243964	1.036422	0.937621	1284336	5120	0.39270
	Decryption	1.099447	1.128025	1.099447	1.055004	1000971	5120	0.87300
F6	Encryption	1.167818	1.248574	1.167818	1.135472	1528213	6144	0.52178
	Decryption	1.303467	1.318073	1.303467	1.276176	1189008	6144	1.13798
F7	Encryption	1.366921	1.398961	1.366921	1.338915	1773615	7168	0.59318
	Decryption	1.512991	1.543327	1.512991	1.472157	1377745	7168	1.25072
F8	Encryption	1.598759	1.649303	1.598759	1.559754	2017213	8192	0.78158
	Decryption	1.708430	1.765692	1.70843	1.687031	1565704	8192	1.19362
F9	Encryption	1.813389	1.826192	1.813389	1.795187	2263234	9216	0.92804
	Decryption	1.936924	2.000418	1.936924	1.900602	1753950	9216	1.15120
F10	Encryption	2.100701	2.169962	2.100701	2.039848	2517554	10240	1.08728
	Decryption	2.276966	2.480676	2.276966	2.184023	1951457	10240	1.29348

Source: Prepared by Authors MATLAB output, (2024).

The Discussion of the Results:

The provided tables present a comprehensive analysis of various metrics, including encryption and decryption times, memory sizes, file sizes, effort, and optimization results for files F1 to F10 using the ECDH_ANN encryption algorithm. These metrics are crucial for evaluating the performance and efficiency of the algorithm, particularly in scenarios where data security and computational resources are of utmost importance. The table (2) clearly demonstrates a consistent trend of increasing encryption and decryption times as the file sizes grow larger. This trend expected, as larger files necessitate more computational resources for processing. Additionally, variations in encryption and decryption times among files of the same size indicate potential disparities in data complexity or structure, which can influence the algorithm's performance. It observed that both encryption and decryption times increase with larger file sizes.

Furthermore, by analyzing growth rate, coefficient of determination (R²), and mean square error (RMSE) values, insights into the algorithm's scalability and accuracy can gained. The high R² values and low RMSE values for both encryption and decryption indicate that linear and polynomial functions are suitable for accurately estimating processing times, which can

be represented symbolically as O(n^k) see fig(5).

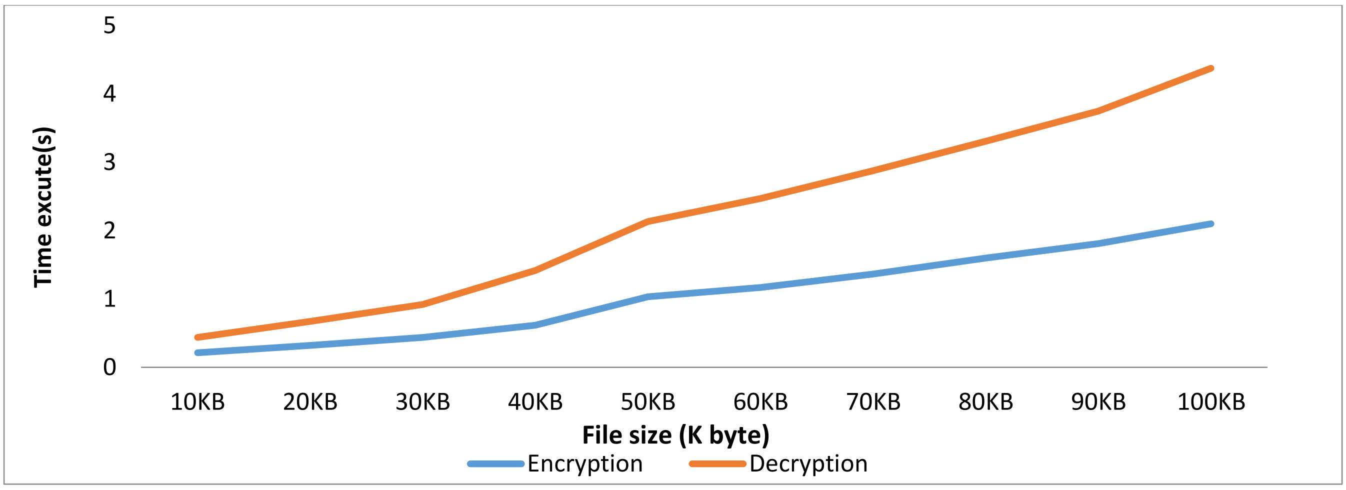

Figure (5): Growth Rate. Source: Prepared by Authors MATLAB output, (2024).

Regarding encryption, the best execution time recorded at 0.200457 seconds, the worst at 2.169962 seconds, and an average execution time of 1.06844208 seconds. Similarly, for decryption, the best execution time is 0.209113 seconds, the worst is 2.480676 seconds, and the average execution time is 1.16259372 seconds see Fig (6).

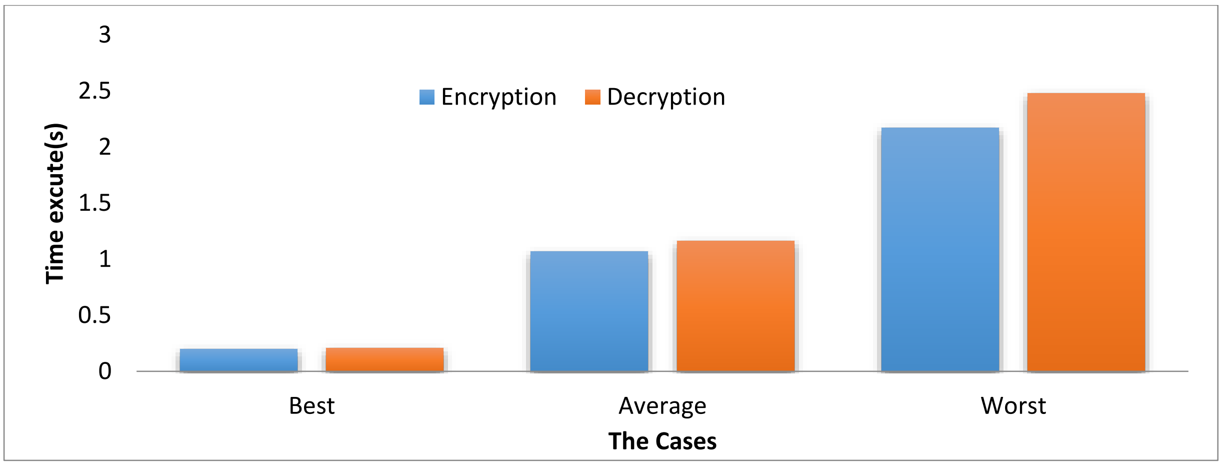

Figure (6): The Best, Average and Worst Cases of Time Execution. Source: Prepared by Authors MATLAB output, (2024).

Additionally, the analysis of memory sizes reveals the algorithm's efficiency in utilizing memory resources. Despite variations in file sizes, the memory sizes required for encryption and decryption processes remain relatively stable, indicating consistent resource allocation regardless of the file size see Fig (7).

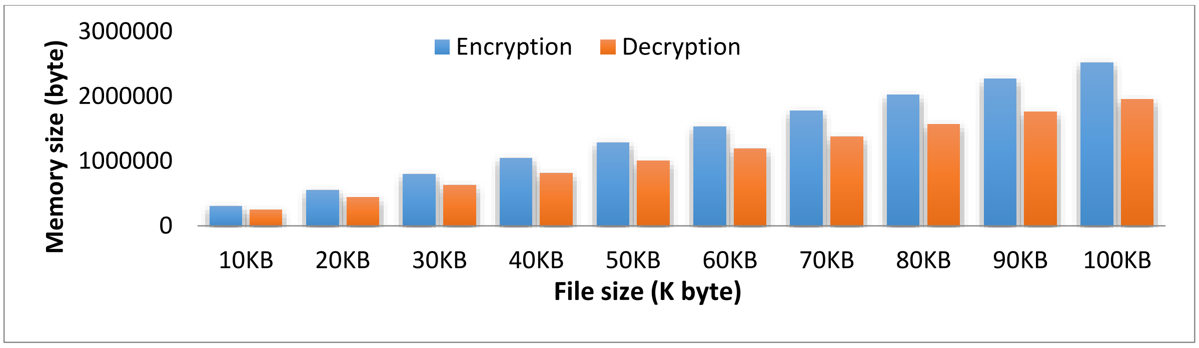

Figure (7): The Memory Sizes Required for Encryption and Decryption Source: Prepared by Authors MATLAB output, (2024).

The discussion on file size preservation throughout the encryption and decryption processes highlights the algorithm's ability to maintain data integrity without altering the file sizes. This feature is essential for applications that require data preservation and integrity assurance.

Furthermore, the discussion emphasizes that the ECDH_ANN encryption algorithm maintains a constant file size, which sets it apart from other algorithms that often result in file size increases. This file size constancy following encryption serves as an indicator of algorithm quality. The choice of encryption method depends on specific requirements, as some methods may be more space-efficient and provide better protection, while others can lead to larger files after encryption. The results indicate that the ECDH_ANN method requires the least effort in the best and average cases but the most effort in the worst case. In terms of decryption efficiency, the effort required for encryption increases as the file size grows. In addition, the optimization results shed light on the algorithm's computational complexity and potential for performance enhancement. While the algorithm achieves acceptable processing times for smaller files, there is room for improvement in handling larger files more efficiently.

Lastly, it should be note that the encryption and decryption processes of the ECDH_ANN algorithm exhibit the same convergence. However, encryption typically involves more operations compared to decryption. Addition is the most frequently used operation, while division is the least frequent.

To summarize, the tables provide a comprehensive analysis of various metrics, revealing insights into the performance, efficiency, and characteristics of the ECDH_ANN encryption algorithm. The findings highlight areas for improvement and optimization, especially for handling larger files more efficiently. The algorithm demonstrates its ability to maintain data integrity and effectively utilize computational resources.

The execution time of the decryption operation is typically about half or less than that of the encryption operation. In many encryption algorithms like ECDH_ANN (Elliptic Curve Diffie-Hellman with Artificial Neural Networks), encryption involves more complex computations, such as key generation and performing mathematical operations on the encrypted data. In contrast, the decryption process generally requires simpler operations, such as using the private key to reverse these computations, making it less computationally intensive. This difference in complexity explains why decryption time is often significantly shorter than encryption time.

The memory size required during the decryption operation is smaller than during encryption. The encryption process usually demands additional memory to store data such as public keys and intermediate calculations related to complex mathematical transformations (e.g., in neural network-based cryptography). In decryption, only the private key is typically required, with fewer intermediate steps or additional data to be stored. As a result, decryption consumes less memory compared to encryption.

The file size after the decryption operation is the same as the original size before encryption. The decryption process is designed to restore the original data from its encrypted form. In most encryption algorithms, including ECDH_ANN, the file size after decryption matches the size before encryption. This means that the encrypted data is simply a transformed version of the original, and when decrypted with the correct key, the original data is fully recovered without any loss of information.

Conclusions:

In conclusion, the complexity analysis and efficiency evaluation of the ECDH_ANN encryption algorithm demonstrate consistent trends in encryption and decryption times, memory usage, and file sizes as file sizes increase. Encryption and decryption times increase noticeably with larger file sizes, while memory usage remains stable, indicating effective resource management.

An important feature of the algorithm is its ability to maintain consistent file sizes during encryption and decryption, setting it apart from other algorithms that often inflate file sizes. This consistent file size preservation is a positive indicator of algorithm quality. The effort required for encryption varies across scenarios, with minimal effort observed in optimal and average cases, and maximal effort in worst-case scenarios. Decryption efficiency improves as file sizes increase. Optimization findings highlight potential areas for enhancing the algorithm's computational complexity and performance, particularly in managing larger files more effectively. Although processing times for smaller files are acceptable, there is room for improvement in handling larger files. It's noteworthy that the encryption and decryption processes of the ECDH_ANN algorithm converge similarly, with encryption involving more operations than decryption. Addition is the most frequently used operation, whereas division is the least frequent.These findings provide a comprehensive analysis across various metrics, offering insights into the performance, efficiency, and characteristics of the ECDH_ANN encryption algorithm. This understanding can guide further refinements and optimizations aimed at enhancing the algorithm's capability to manage larger files and optimize resource utilization. The algorithm demonstrates its ability to maintain data integrity and efficiently utilize computational resources, making it suitable for scenarios prioritizing both data security and computational efficiency.

Recommendations and Further Studies

Based on the analysis and findings presented in the text, here are some recommendations and suggestions for further studies:

1. Optimization: Focus on improving encryption and decryption efficiency for larger files, possibly through parallel computing or distributed systems.
2. Algorithm Comparison: Compare ECDH_ANN with other encryption algorithms to understand performance, efficiency, and security trade-offs.
3. Security Analysis: Conduct a thorough security assessment to identify and mitigate potential vulnerabilities in ECDH_ANN.
4. Real-world Implementation: Validate ECDH_ANN's performance in diverse environments to assess practical viability.
5. Scalability Analysis: Evaluate how ECDH_ANN handles increasing file sizes to determine scalability and resource requirements.
6. Neural Network Impact: Investigate the influence of different neural network architectures and parameters on ECDH_ANN's efficiency.
7. Energy Efficiency: Analyze ECDH_ANN's energy consumption compared to other algorithms to optimize its energy efficiency.
8. Application-specific Studies: Conduct studies in IoT, cloud computing, and mobile applications to assess ECDH_ANN's performance, efficiency, and security suitability. Comparing it with other algorithms can help identify the best fit for specific use cases.

Reference:

Alesawy, O., & Muniyandi, R. (2016). Elliptic curve Diffie-Hellman random keys using artificial neural network and genetic algorithm for secure data over private cloud. *Information Technology Journal, 15, 77-83. https://doi.org/10.3923/itj.2016.77.83

Al-nima, R. R., Muhanad, L., & Hassan, S. Q. (2009). Data encryption using backpropagation neural network.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (1990). Introduction to algorithms (2nd ed.). The MIT Press New York San Francisco St. Louis Montréal Toronto .

Chakraborty, R. C. (2010). Back propagation network soft computing back propagation network. Retrieved from http://www.myreaders.info/2010

Duan, X., Guo, D., Liu, N., Li, B., Gou, M., & Qin, C. (2020). A new high capacity image steganography method combined with image elliptic curve cryptography and deep neural network. IEEE Access, 8, 24120-24128. https://doi.org/10.1109/ACCESS.2020.297152

Laskari, E. C., Meletiou, G. C., Stamatiou, A., & Vrahatis, M. N. (2007). Assessing the effectiveness of artificial neural networks on problems related to elliptic curve cryptography. Mathematical and Computer Modelling, 46(7-8), 174-179. https://doi.org/10.1016/j.mcm.2006.12.013

Naser, A., & Zolkipli, M. F. (2016). A conceptual model using elliptic curve Diffie-Hellman with an artificial neural network over cloud computing. In National Conference for Postgraduate Research.

Paar, C., & Pelzl, J. (1998). *Understanding cryptography. Springer. https://doi.org/10.1007/978-3-642-04101-3

Singh, P., & Singh, H. (2012). Cryptography in structure adaptable digital neural networks. National Monthly Refereed Journal of Research in Science and Technology, 1 (12), 35-44.

Singh, A., & Aartinandal. (2020). Neural cryptography for secret key exchange and encryption with AES. International Journal of Advanced Research in Computer Science and Software Engineering, 3 (5), 376-381.