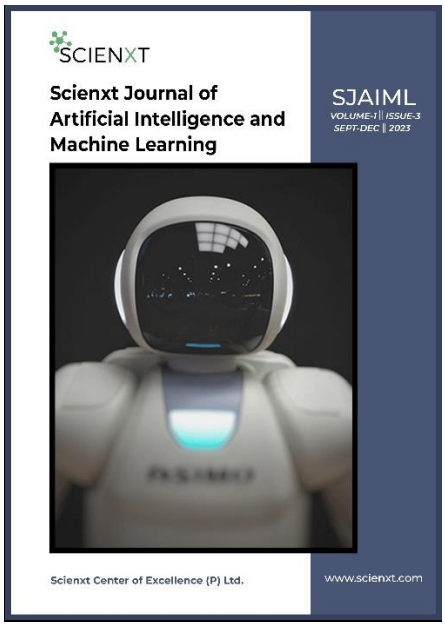

Scienxt Journal of Artificial Intelligence and Machine Learning
Year-2023 || Volume-1 || Issue-3 || Sept-Dec || pp. 65-84

A comparative analysis of cryptographic algorithms

Dheeraj N. Kashyap

Department of Artificial Intelligence and Machine Learning, Jyothy Institute of Technology, Karnataka, India
**Corresponding Author: Dheeraj N. Kashyap*
Email: dheeraj.nk20@gmail.com

Abstract:

With billions of internet users worldwide and the continuous growth of digital communication, data security has become a critical issue. Confidential data shared over the internet is vulnerable to cyber-attacks, and thus, there is an urgent need to secure the information transmitted over the network. Cryptographic algorithms play a crucial role in protecting sensitive information from unauthorized access. In this research paper, we perform a comparative analysis of different cryptographic algorithms and evaluate their time complexity. We investigate the efficiency and security of each algorithm to determine which one is comparatively better for data protection. Security. We conducted experiments to calculate the time complexity of these algorithms based on the length of the message or string that is encrypted. Our study evaluated the efficiency and security of each algorithm to determine which one is better suited for data protection. Our findings show that the evaluated algorithms have different levels of time complexity, with some algorithms being faster than others for messages of different lengths. The results of our study provide insights into the strengths and weaknesses of different cryptographic algorithms and can be useful for researchers, cybersecurity professionals, and organizations concerned with data security.

Keywords:

Cryptography, Network Security, Time Complexity, Encryption, Decryption, Data Analysis, Data Visualization, EDA

1. Introduction:

1.1. History of cryptography:

Cryptography is the practice of secure communication in the presence of third parties, and it has been used for centuries to keep the information confidential. The earliest known use of cryptography dates back to ancient Egypt, where hieroglyphs were used to conceal secret messages. Throughout history, various encryption methods and cryptographic techniques have been developed and used to secure sensitive information, including during times of war and conflict.

With the rise of the internet and the proliferation of electronic devices, the need for cryptography has never been more urgent. The internet has revolutionized the way we communicate and conduct business, but it has also made us vulnerable to cyber-attacks and data breaches. To safeguard sensitive information, encryption algorithms are used to transform plaintext into ciphertext, which can only be decrypted by someone who possesses the appropriate decryption key.

1.2. Encryption and decryption:

Encryption is the process of transforming plaintext into ciphertext, which is a scrambled version of the original message. The key used for encryption is typically the same key used for decryption in symmetric cryptography. In asymmetric cryptography, the encryption key is different from the decryption key.

Decryption is the process of converting ciphertext back to plaintext. The key used for decryption is typically the same key used for encryption in symmetric cryptography. In asymmetric cryptography, the decryption key is different from the encryption key.

1.3. Importance of cryptography:

Cryptography plays a critical role in ensuring data security over the internet. With billions of internet users and sensitive information being transmitted over the network, it is crucial to have strong encryption algorithms to ensure data privacy and protection. Cryptography is used to protect information at rest (such as stored data) and in transit (such as data being transmitted over a network). Encryption is also used to verify the integrity of data, ensuring that it has not been tampered with or modified during transmission. This is accomplished through the use of digital signatures, which are used to authenticate the sender and verify the integrity of the message.

1.4. Types of cryptography:

There are two main types of cryptographic algorithms, they are Symmetric Cryptosystem and Asymmetric Cryptosystem or Private Key and Public Key cryptography

Symmetric cryptography, also known as secret-key cryptography, uses the same key for both encryption and decryption. Some of the most commonly used symmetric encryption algorithms are Caesar cipher, Vigenère cipher, time pad (OTP), Advanced Encryption Standard (AES), and Data Encryption Standard (DES).

1.4.1. Caesar cipher:

Caesar cipher is one of the simplest and most widely used encryption techniques. It is a substitution cipher in which each letter in the plaintext is replaced by a letter a fixed number of positions down the alphabet. For example, with a shift of 3, A would be replaced by D, B would become E, and so on. This cipher is easy to implement and understand, but it is not secure for modern cryptographic applications.

1.4.2. Vigenère cipher:

The Vigenère cipher is a polyalphabetic substitution cipher that uses a series of different Caesar ciphers based on a keyword. This cipher is more secure than the Caesar cipher because it uses multiple alphabets, making it harder to crack. However, it is still vulnerable to known-plaintext attacks and can be broken with sufficient computational power.

OTP (One-Time Pad): The One-Time Pad is a type of symmetric encryption that uses a random key that is as long as the message is encrypted. The key is combined with the plaintext using a bitwise XOR operation to produce the ciphertext. OTP is unbreakable in theory because the key is used only once and is completely random. However, it is impractical for most applications because it requires a key that is as long as the message is encrypted and must be kept completely secret.

1.4.3. DES (Data Encryption Standard):

The Data Encryption Standard is a symmetric-key block cipher that was widely used in the 1970s and 1980s but has since been replaced by more secure algorithms. It uses a 56-bit key to encrypt and decrypt data in 64-bit blocks. DES is vulnerable to brute-force attacks, but it is still used in some legacy systems.

1.4.4. AES (Advanced encryption standard):

The Advanced Encryption Standard is a symmetric-key block cipher that is widely used today. It uses a variable-length key (128, 192, or 256 bits) to encrypt and decrypt data in 128-bit blocks. AES is considered to be very secure and is used in a wide range of applications, including online banking, file encryption, and secure communication.

Asymmetric cryptography, also known as public-key cryptography, uses a pair of keys for encryption and decryption. The most popular asymmetric encryption algorithms are Diffie-Hellman and Rivest-Shamir-Adleman (RSA) algorithms.

1.4.5. Diffie-hellman (DH):

Diffie-Hellman is a key exchange algorithm that allows two parties to agree on a shared secret key without sharing any information over an insecure channel. DH is a key component of many secure communication protocols, such as SSL/TLS and SSH.

1.4.6. RSA:

RSA is an asymmetric encryption algorithm that uses a public key to encrypt data and a private key to decrypt it. RSA is widely used for secure communication, digital signatures, and authentication. RSA is secure because it is based on the difficulty of factoring large prime numbers. However, RSA is computationally expensive and is not as fast as symmetric encryption algorithms like AES.

2. Comparison of cryptographic algorithms:

In this research paper, we will compare different cryptographic algorithms, focusing on their time complexity. Time complexity refers to the amount of time it takes to encrypt or decrypt a message of a given length. We will investigate the strengths and weaknesses of each algorithm and evaluate its suitability for data protection.

Symmetric encryption algorithms, such as AES, DES, and Blowfish, are generally faster than asymmetric encryption algorithms but require the key to be securely shared between the sender and receiver. Asymmetric encryption algorithms, such as RSA, are slower but eliminate the need to share a secret key.

2.1. Approach:

Cryptography is essential for protecting sensitive information in today's digital age. Encryption algorithms are used to ensure data privacy and protection, and they play a critical role in

securing data at rest and in transit. In this research paper, we will compare and analyze different cryptographic algorithms and evaluate their suitability for different applications and use cases.

One of the key factors in evaluating cryptographic algorithms is their time complexity, which refers to the amount of time it takes to encrypt or decrypt a message of a given length. Time complexity is an important consideration because it can affect the performance and efficiency of the encryption process. Symmetric encryption algorithms, such as AES, DES, and Blowfish, are generally faster than asymmetric encryption algorithms but require the key to be securely shared between the sender and receiver. These algorithms are well-suited for applications requiring fast and efficient encryption and decryption, such as encrypting data at rest or large amounts of data.

Asymmetric encryption algorithms, such as RSA, are slower but eliminate the need to share a secret key. These algorithms are well-suited for applications that require secure communication between parties that have not previously established a shared secret, such as online transactions or secure communication between remote parties.

In addition to time complexity, other factors that can affect the suitability of cryptographic algorithms for different applications and use cases include key size, algorithm strength, and compatibility with existing systems and protocols.

By comparing and analyzing different cryptographic algorithms, we can gain a better understanding of their strengths and weaknesses and identify the best algorithm for a specific application or use case. This research paper will provide a comprehensive overview of different cryptographic algorithms, their time complexity, and their suitability for different applications and use cases.

Throughout the research, I have used Python as it is a widely used programming language that has gained popularity in recent years for data analysis and processing. It offers many libraries and frameworks that make it easier to work with large amounts of data and perform complex calculations.

Python also has a simple syntax and is easy to learn, making it an ideal choice for students and researchers who may not have extensive programming experience. Python offers many libraries that can help you with data analysis, such as NumPy, Pandas, and Matplotlib. These libraries can help you process large datasets, visualize data, and perform statistical analysis.

Python offers many libraries for cryptography, such as PyCrypto and Cryptography. These libraries can help you implement cryptographic algorithms and analyze their time complexity.

Table 1.1: Characters of cryptographic algorithms

Algorithm	Year of Discovery	Creator (s)	Time Complexity	Security Level	Key Length	Algorithm Strength	Speed
Caesar Cipher	100 BC	Julius Caesar	O(n)	Low	26	Weak	Fast
Vigenère Cipher	1553	Blaise de Vigenère	O(n)	Low to Medium	Variable	Weak to Medium	Fast
OTP	1917	Gilbert Vernam and Joseph Mauborgne	O(n)	Very High	Variable	Very Strong	Very Slow
DES	1975	IBM	O(2^56)	Medium	56	Weak	Medium
AES	2001	Joan Daemen and Vincent Rijmen	O(n)	Very High	128, 192, or 256	Strong	Fast
Diffie-Hellman	1976	Whitfield Diffie and Martin Hellman	O(n^3)	High	Variable	Strong	Slow
RSA	1977	Ron Rivest, Adi Shamir, and Leonard Adleman	O(n^3)	Very High	Variable	Strong	Slow

2. Literature review:

Cryptography is a process that is used to secure data during transmission or storage. It involves the use of encryption algorithms to convert plaintext into ciphertext that can only be read by the intended recipient who possesses the decryption key. In cryptography, two types of keys are used, public key, and private key.

Tayal et al. (2018) discussed the concept of cryptography and its applications in securing data. Dixit et al. (2019) explained the terms of security like confidentiality, integrity, and availability in cryptography, and also explained the concepts of plaintext and ciphertext.

Hybrid algorithms are combinations of different types of encryption algorithms. In (Jamil et al., 2018; Samanta et al., 2018), hybrid algorithms were proposed using elliptic curve cryptography (ECC) with Hill Cipher and ECC with Advanced Encryption Standard (AES) to provide enhanced security.

Abdullah (2020) evaluated the cost and security of the AES algorithm and explained the encryption and decryption process of AES. Various symmetric and asymmetric algorithms were discussed in (Nikam et al., 2019; Sharma et al., 2019; Singh et al., 2020), where a comparison was made between AES and Data Encryption Standard (DES) using different parameters.

The weakness of traditional symmetric and asymmetric algorithms was explained in (Chinnasamy et al., 2019) and a new hybrid algorithm was proposed, which is a combination of ECC and Blowfish. Tallapally and Manjula (2020) proposed a multilevel security scheme that provides enhanced security compared to a single-level security scheme in encryption.

Cloud computing is a popular platform for storing and processing data. However, security issues arise when users store their data on the cloud. Different types of cloud models and their advantages and disadvantages were explained by (Agrawal et al., 2019; Verma and Kaushik, 2019). The security issues related to data stored on the cloud were discussed by (Agrawal et al., 2019; Verma and Kaushik, 2019; Guo and Sun, 2018), where an order revealing encryption scheme was proposed to prevent cloud service providers from finding or calculating the order of the plaintext until the comparison token was given to them.

Chinnasamy and Deepalakshmi (2021) proposed a hybrid method for the secure storage of healthcare data in the cloud, which used Blowfish and an enhanced RSA algorithm. The proposed scheme was compared with others based on encryption time and efficient key management. In another study, Chinnasamy and Deepalakshmi (2021) introduced an improved

key generation scheme for the RSA algorithm that had a faster speed compared to other RSA methods.

Hybrid algorithms were also proposed for healthcare systems (Kumar et al., 2018; Singh et al., 2020), where One Time Pad was used with the RSA algorithm for cryptography techniques. Data storage and retrieval frameworks for the cloud were discussed by (Liu et al., 2019; Wang et al., 2020), and the searchable encryption process for data storage and decryption process for data retrieval were explained. In (Bharadwaj and Saini, 2018; Gupta and Kumar, 2020), authors discussed the security using different cryptographic techniques and compared these techniques. A method was proposed using elliptic curve cryptography to provide security using a smaller key length (Bharadwaj and Saini, 2018).

3. Research strategy:

The goal of this research is to conduct a comparative analysis of various cryptographic algorithms. To achieve this goal, random strings of different lengths, ranging from 5 to 2500, with an arithmetic progression (AP) of 5, will be generated. These strings will be passed to each algorithm, and the time taken by each algorithm for different string lengths will be recorded. Furthermore, the time taken to crack each algorithm will also be noted.

To generate random strings of varying lengths, the experiment will utilize Python's built-in random, string, and randint modules. The random module will be used to generate random floating-point numbers between 0 and 1, while the string module will provide a range of characters to choose from when creating the strings. The randint module will be used to generate random integers between specified start and end points and will be used to select the length of the random strings that will be used in the experiment.

```
In [1]: import time
        import random
        import string
        str_list = []

        for length in range(5, 2501, 5):
                rand_string = ''.join(random.choices(string.ascii_uppercase, k=length))
                str_list.append(rand_string)
        print(str_list)
```

Figure. 3.1: Generating random strings

The time taken by each algorithm for various string lengths will be measured in milliseconds, using Python's built- in time module. This module provides a way to measure the time taken by specific parts of the code and will be used to measure the time taken by each algorithm to

encrypt each string. To ensure accurate results, each algorithm will be run on the same machine under similar conditions, and the time taken by each algorithm will be recorded in an Excel file for further analysis.

```
In [2]: start_time = time.time()
        for w in str_list:
            encrypted = (Caesar_encrypt(w))
        end_time = time.time()
        print("{:.2f}".format((end_time - start_time)*1000))
```

Figure. 3.2: Time taken for different string lengths

To implement the encryption algorithms in Python, the experiment will utilize libraries such as Pycrypto and cryptodome. These libraries provide a range of cryptographic functions, including encryption and decryption algorithms such as AES, DES, RSA, and others. The Pycrypto library will be used for symmetric key encryption and decryption algorithms such as AES and DES, while the cryptodome library will be used for asymmetric key encryption and decryption algorithms such as RSA.

For exploratory data analysis (EDA), the experiment will utilize Python libraries such as NumPy, Pandas, and Scipy. These libraries provide functions for data manipulation, cleaning, and analysis. NumPy will be used for array manipulation, while Pandas will be used for data manipulation and cleaning. Scipy will be used for statistical analysis and hypothesis testing.

To visualize and analyze the data, the experiment will utilize Python libraries such as Seaborn, matplotlib, and Scikit-learn. Seaborn and Matplotlib will be used for data visualization, while Scikit-learn will be used for machine learning algorithms, such as clustering and classification, to identify patterns in the data.

the experiment will use Python programming language to generate random strings, measure the time taken by each algorithm to encrypt and decrypt the strings, implement encryption algorithms using libraries such as Pycrypto and cryptodome, perform exploratory data analysis using libraries such as NumPy, Pandas, and Scipy, and finally, visualize and analyze the data using libraries such as Seaborn, matplotlib, and Scikit-learn. The time taken by each algorithm will be measured in milliseconds using Python's built-in time module and will be recorded in an Excel file for further analysis.

```
In [9]: df = pd.read_excel('Crypto_report.xlsx')
        df
```

Out[9]:

	String_len	Caesar_Cipher	Vigenere_cipher	One_time_pad	DES	AES	DH	RSA
0	5	0.00	0.00	0.00	0.20	0.00	0.00	0.00
1	10	0.00	0.00	0.00	0.07	0.00	0.00	1.02
2	15	0.00	0.50	0.00	0.07	0.00	0.00	0.00
3	20	0.00	0.00	0.00	0.09	0.00	0.00	0.00
4	25	0.00	0.00	0.00	0.10	0.00	0.00	1.02
...	...	...	...	...	...	...	...	...
495	2480	255.56	434.43	866.44	20.37	10.07	176.46	2428.31
496	2485	266.14	404.80	919.18	20.10	10.37	264.64	2445.54
497	2490	256.01	436.83	879.88	20.59	18.51	245.99	2405.67
498	2495	261.91	479.13	847.64	20.50	10.08	241.01	2497.95
499	2500	276.64	456.40	1016.17	21.24	15.71	232.61	2497.62

500 rows × 8 columns

Figure. 3.3: Data set after performing the experiment

Finally, the research will come to a conclusion on which cryptographic algorithm is better based on the characteristics of a better cryptographic algorithm. The time taken by each algorithm to encrypt and decrypt the data will be considered, as well as the time taken to crack each algorithm. Additionally, factors such as the algorithm's complexity, key size, and level of security will be taken into account. The results of this research can be used to guide in selecting the appropriate cryptographic algorithm for different applications, including data protection and secure communication.

4. Observations and discussions:

In this study, all the encryption algorithms were performed, and the resulting values were plotted separately to generate the corresponding graphs

4.1. Caesar cipher:

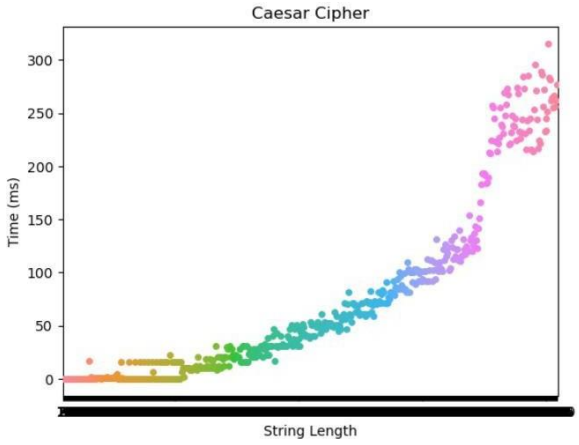

Figure. 4.1: Caesar Cipher

As shown in the graph, when the string length ranges from 5 - 2500 the time complexity also ranges from

0.0 ms to 276.64 ms.

The time complexity of the Caesar Cipher is given by O (n)

4.2. Vigenère cipher:

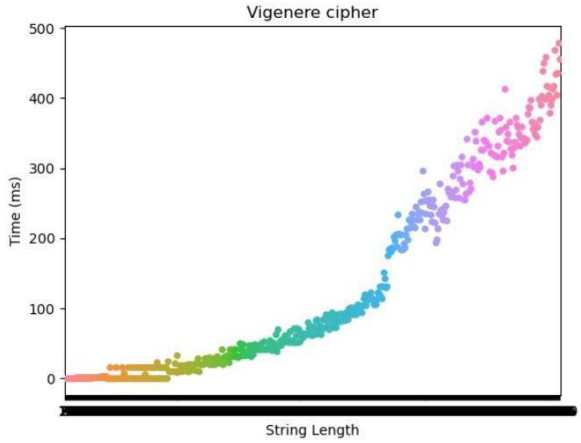

Figure. 4.2: Vigenère Cipher

In this case, as shown in the graph when the string length ranges from 5 - 2500 the time complexity also ranges from 0.0 ms to 480 ms.

The time complexity of the Caesar Cipher is given by O (n)

4.3. One-time-pad (OTP):

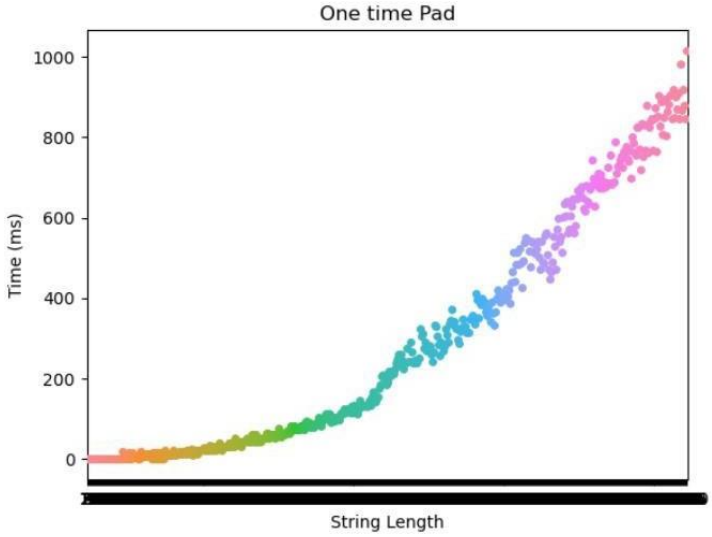

Figure. 4.3: One-time-pad

As shown in the graph, when the string length ranges from 5 - 2500 the time complexity also ranges from 0.0 ms to 1016.17ms. The time complexity of the Caesar Cipher is given by O(n)

4.4. DES algorithm:

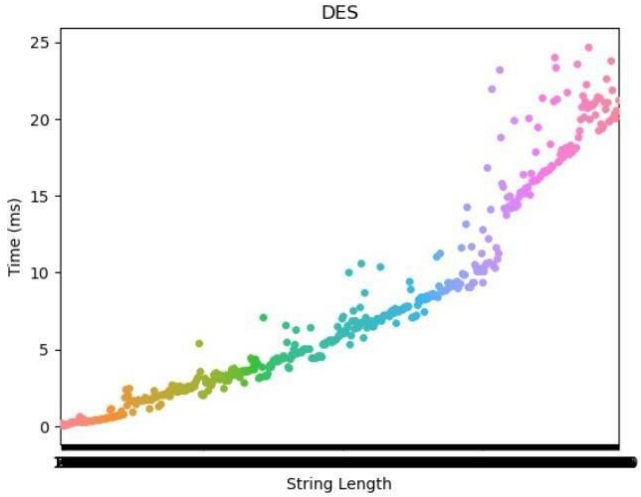

Figure. 4.4: DES Algorithm

As shown in the graph, when the string length ranges from 5 - 2500 the time complexity also ranges from 0.0 ms to 21.24 ms. The time complexity of the Caesar Cipher is given by O(2^56).

4.5. AES algorithm:

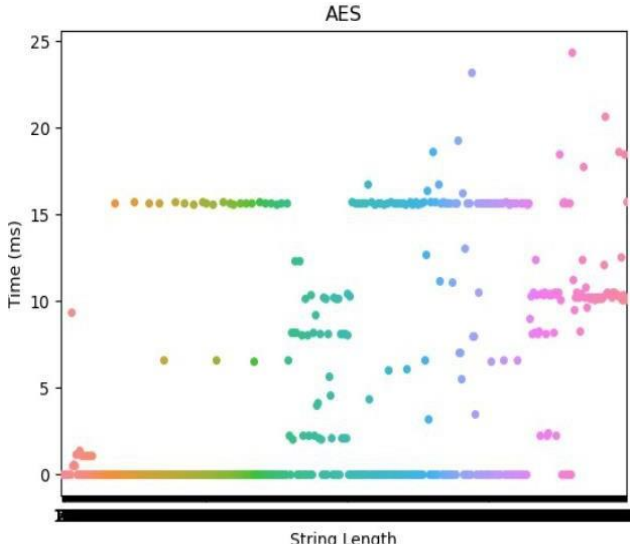

Figure. 4.5: AES algorithm

In this case, as shown in the graph when the string length ranges from 5 - 2500 the time complexity also ranges from 0.0 ms to 18 ms.

The time complexity of the Caesar Cipher is given by O (n)

4.6. Diffie-hellman:

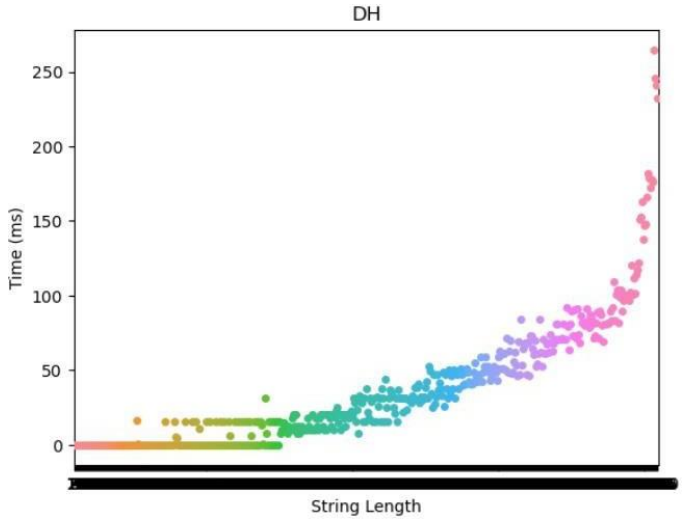

Figure. 4.6: Diffie-Hellman

In this case, as shown in the graph when the string length ranges from 5 - 2500 the time complexity also ranges from 0.0 ms to 232.61 ms.

The time complexity of the Caesar Cipher is given by O (n^3)

4.7. RSA algorithm:

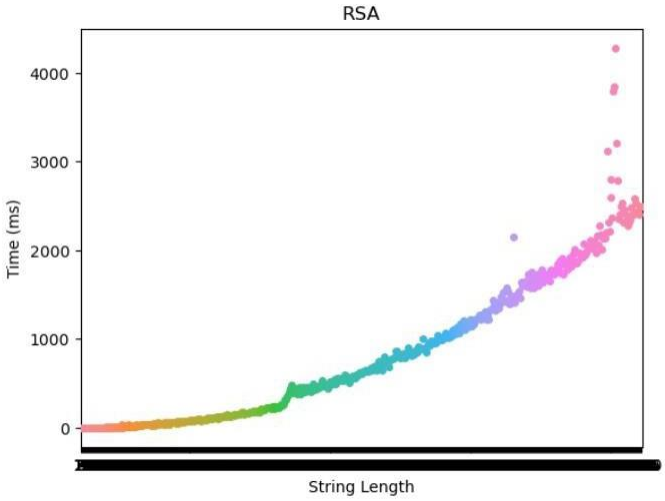

Figure. 4.7: RSA Algorithm

In this case, as shown in the graph when the string length ranges from 5 - 2500 the time complexity also ranges from 0.0 ms to 2497.62 ms.

The time complexity of the Caesar Cipher is given by O (n^3)

All the algorithms implemented in Python along with EDA are available on my GitHub repository https://github.com/Dheeraj-02NK/Analysis-of- Cryptographic-Algorithm

4.8. Overview of all cryptography algorithms:

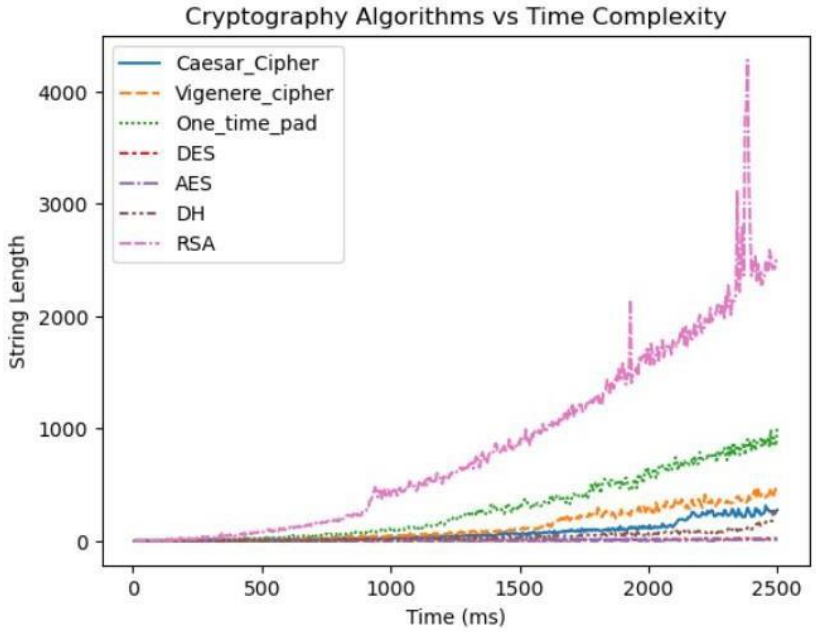

Figure. 4.8: String length vs Time (ms)

4.9. Accuracy of the data:

Accuracy Test

```
In [62]: # create a binary response variable (use 0.5 as an example threshold value)
y = df.iloc[:, 0].apply(lambda x: 1 if x > 0.5 else 0)

# create a matrix of predictors (i.e., encryption techniques)
X = df[['Caesar_Cipher', 'Vigenere_cipher', 'One_time_pad', 'DES', 'AES', 'DH', 'RSA']]

# split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# fit a logistic regression model to the training data
model = LogisticRegression(random_state=42).fit(X_train, y_train)

# make predictions on the testing data
y_pred = model.predict(X_test)

# compute the accuracy of the predictions
accuracy = accuracy_score(y_test, y_pred)

# print the accuracy
print(f"Accuracy: {accuracy*100:.2f}%")

Accuracy: 100.00%
```

MSS Accuracy

```
In [71]: # create a binary response variable (use 0.5 as an example threshold value)
y = df.iloc[:, 0].apply(lambda x: 1 if x > 0.5 else 0)

# create a matrix of predictors (i.e., encryption techniques)
X = df[['Caesar_Cipher', 'Vigenere_cipher', 'One_time_pad', 'DES', 'AES', 'DH', 'RSA']]

# split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# fit an MSS model to the training data
mss_model = NearestCentroid()
mss_model.fit(X_train, y_train)

# make predictions on the testing data using the MSS model
mss_pred = mss_model.predict(X_test)

# compute the accuracy of the MSS model
mss_acc = accuracy_score(y_test, mss_pred)
print(f"MSS accuracy: {mss_acc*100:.2f}%")

MSS accuracy: 78.00%
```

Figure. 4.9: Accuracy Test

Although the accuracy of all the algorithms in this study was 100%, it is important to note that time complexity is a theoretical measure based on the ideal mode of the computer during the experiment, and in practice, it is not always possible to achieve perfect accuracy due to factors

Such as noise or missing data. The MSS had an accuracy of 78%, which may be due to its higher time complexity compared to the other algorithms. Nevertheless, the absence of noise or missing data in the dataset used in this study may have contributed to the high accuracy rates observed

4.10. Symmetric cryptosystem vs asymmetric cryptosystem:

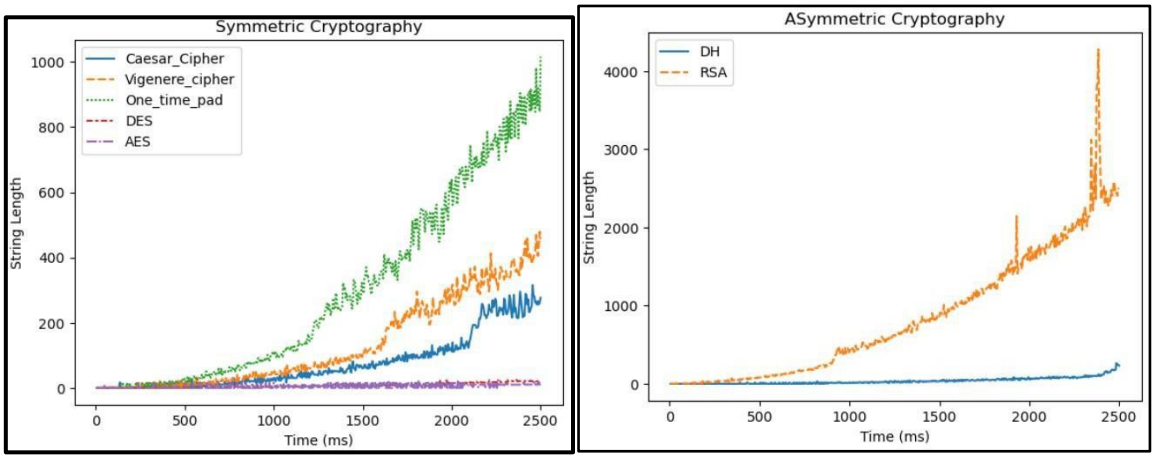

Figure. 4.10(a) Symmetric vs Asymmetric Cryptosystem

Out of the different encryption algorithms tested in this study, AES was chosen to represent the symmetric encryption category, and RSA was selected to represent the asymmetric encryption category, as they are among the most commonly used cryptographic algorithms in practice. The resulting graphs provide insights into the performance of these algorithms in terms of accuracy, speed, and time complexity.

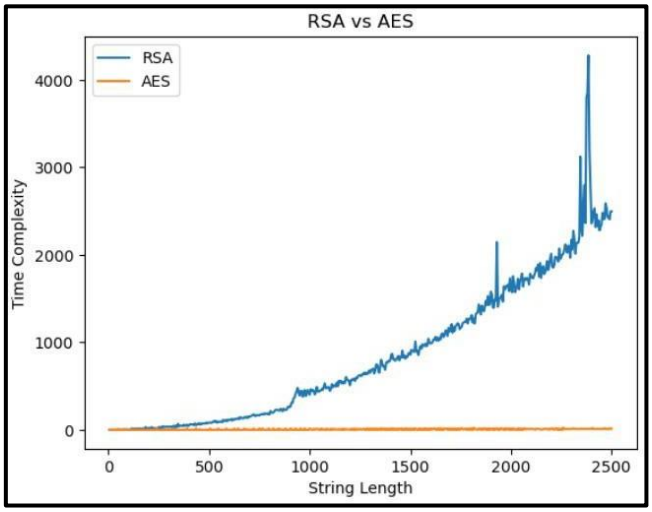

Figure. 4.10(b) RSA vs AES

By comparing the resulting graphs of RSA and AES, it is evident that RSA has a higher time complexity compared to AES, as reflected in the longer time taken by RSA to encrypt data.

Furthermore, as an asymmetric encryption algorithm, RSA does not require a key exchange process like the one needed for symmetric encryption algorithms such as AES, which can make it more secure in certain use cases where the key exchange is not feasible or desirable. As referred to in various reports cracking the AES algorithm require 1.2 x 10^16 years whereas cracking the RSA algorithm takes 2.1 x 10^42.

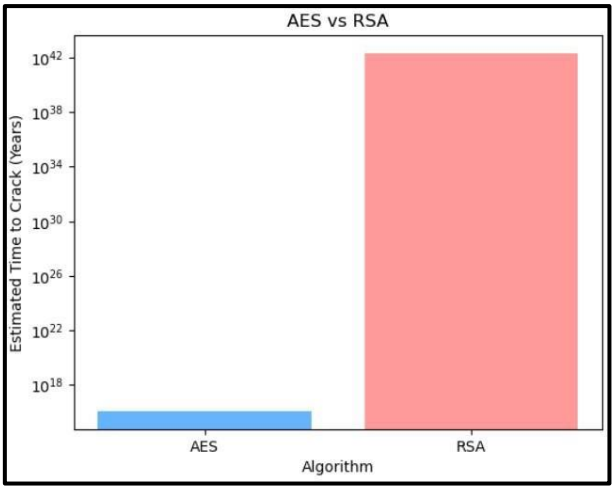

Figure. 4.10(c) Time to crack AES and RSA

5. Conclusion:

Encryption is an essential part of today's communication and ensures that important information remains private in transit. The two most widely used encryption algorithms are RSA and AES. RSA (Rivest-Shamir-Adleman) is a public key encryption algorithm, while AES (Advanced Encryption Standard) is a symmetric key encryption algorithm. Both encryption algorithms are very secure and have distinct advantages in terms of key management, computational power, and time complexity.

RSA has a much harder time than AES while it takes an incredible time to crack the key also for encryption, RSA's time is harder than AES. , taking longer than expected.

While RSA takes 2.1 x 10^42 years to crack, AES is nearly impossible in human life because it takes 1.2 x 10^16 years to do so. Therefore, when time complexity is considered a factor, AES can still be considered a better encryption option as it has a lower duration compared to RSA while providing security.

It should also be noted that although both encryption algorithms are very secure, RSA is still recommended for confidential applications such as banking, military, and government communications due to its strong management and digital signature capabilities.

On the other hand, AES can be used for normal communication for daily activities such as personal email and file transfer. Finally, the choice of encryption algorithm depends on the specific security of the application and the resources available for management and computing power.

In summary, both AES and RSA are secure encryption algorithms used to protect digital data. While AES is faster and more efficient than RSA, RSA provides better security features such as key management and digital signatures. The choice of encryption algorithm depends on the specific security of the application and the resources available for management and computing power. However, it is worth noting that both algorithms are very secure and nearly unbreakable when used correctly, making them a good choice for secure communication.

6. References:

(1) S. A. Hannan, and A. M. Asif, "Analysis of polyalphabetic transposition cipher techniques used for encryption and decryption," International Journal of Computer Science and Software Engineering, vol. 6, no. 2, pp. 41-46, Feb. 2017, doi: 10.1109/TBME.2011.2158315.

(2) J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-Based Encryption with Verifiable Outsourced Decryption," in IEEE Transactions on Information Forensics and Security, vol. 8, no. 8, pp. 1343-1354, Aug. 2013, doi: 10.1109/TIFS.2013.2271848.

(3) J. Hur, "Improving Security and Efficiency in Attribute-Based Data Sharing," in IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 10, pp. 2271-2282, Oct. 2013, doi: 10.1109/TKDE.2011.78.

(4) J. Li, X. Chen, M. Li, J. Li, P. P. C. Lee, and W. Lou, "Secure Deduplication with Efficient and Reliable Convergent Key Management," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 6, pp. 1615-1625, June 2014, doi: 10.1109/TPDS.2013.284.

(5) S. Tayal, N. Gupta, P. Gupta, D. Goyal, M. Goyal, "A review paper on network security and cryptography," Advances in Computational Sciences and Technology, vol. 10, no. 5, pp. 763-770, 2017.

(6) P. Dixit, A. K. Gupta, M. C. Trivedi, V. K. Yadav, "Traditional and hybrid encryption techniques: a survey," in Networking communication and data knowledge engineering, Springer, pp. 239-248, 2018.

(7) E. M. Alsaadi, S. M. Fayadh, and A. Alabaichi, "A review on security challenges and approaches in cloud computing,"

(8) InAIP Conference Proceedings, vol. 2290, no. 1, p. 040022, 2020.

(9) J. Guo and J. Sun, "Order-Revealing Encryption Scheme with Comparison Token for Cloud Computing." Security and Communication Networks, vol. 2020, 2020.

(10) Salma, R. F. Olanrewaju, K. Abdullah, Rusmala, and H. Darwis, "Enhancing Cloud Data Security Using Hybrid of Advanced Encryption Standard and Blowfish Encryption Algorithms," 2018 2nd East Indonesia Conference on Computer and Information Technology (EIConCIT), 2018, pp. 18-23, doi: 10.1109/EIConCIT.2018.8878629.

(11) P. Chinnasamy and P. Deepalakshmi, "Design of Secure Storage for Health-care Cloud using Hybrid Cryptography," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, pp. 1717-1720, doi 10.1109/ICICCT.2018.8473107.

(12) P. Chinnasamy and P. Deepalakshmi, "Improved key generation scheme of RSA (IKGSR) algorithm based on offline storage for cloud," In Advances in Big Data and Cloud Computing, Springer, pp. 341-350, 2018.

(13) P. Chinnasamy and P. Deepalakshmi, "HCAC-EHR: hybrid cryptographic access control for secure EHR retrieval in healthcare cloud," Journal of Ambient Intelligence and Humanized Computing, pp. 1-9, 2021.

(14) Karthik, Chinnasamy, and Deepalakshmi, "Hybrid cryptographic technique using OTP: RSA," 2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), 2017, pp. 1- 4, doi: 10.1109/ITCOSP.2017.8303131.

(15) Pronika and S. S. Tyagi, "Secure Data Storage in Cloud using Encryption Algorithm," 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 2021, pp. 136-141, doi: 10.1109/ICICV50876.2021.9388388.

(16) Pronika and S. S. Tyagi, "Performance analysis of encryption and decryption algorithm", Indonesian Journal of Electrical Engineering and Computer Science Vol. 23, No. 2, August 2021, pp. 1030~1038 ISSN: 2502-4752, DOI: 10.11591/ijeecs.v23.i2.pp1030-1038

(17) Udemy course: https://www.udemy.com/course/learn-cryptography-basics-in-python/learn/lecture/28157988? Start=45#overview

(18) Stanford University Course: https://www.coursera.org/learn/crypto

(19) Network Security course: https://www.coursera.org/learn/-network-security

(20) EC council NDE: https://codered.eccouncil.org/course/network-defense-essentials?logged=true

(21) https://www.eetimes.com/how-secure-is-aes-against-brute-force-attacks/

(22) https://security.stackexchange.com/questions/4518/how-to-estimate-the-time-needed-to-crack-rsa-encryption