A study on the complexity of Cryptographic Algorithms used for Secure Messaging

Industry Internship
MSc Cybersecurity

Ameet Rahegaonkar
Student ID: X19239122

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ameet Rahegaonkar		
Student ID:	X19239122		
Programme:	MSc in Cybersecurity	**Year:**	2021-2022
Module:	Industry Internship		
Supervisor:	Vikas Sahni		
Submission Due Date:	07/01/2022		
Project Title:	A study on the complexity of Cryptographic Algorithms used for Secure Messaging		
Word Count:	6344	**Page Count**	22

Signature:	Ameet Rahegaonkar
Date:	07/01/2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A study on the complexity of Cryptographic Algorithms used for Secure Messaging

Ameet Rahegaonkar
X19239122

Abstract

Presently, one of the most efficient forms of communication between people is mobile messaging applications, which provide a range of interesting services. Therefore, there are different level of chances that data attacks may occur and there are increase in vulnerabilities.

Therefore, there is a need for secure messaging protocol for an encrypted communication to overcome these shortcomings and risk of data attacks. Many researchers have researched on different messaging protocols and the algorithm that provide end-to-end encryption.

This research is carried out to measure the complexity of the cryptographic algorithms which are used for secure messaging and provide end-to-end encryption. Time and Space complexity of the three cryptographic algorithms used, AES, RSA and Double-Ratchet is measured. The complexity is measured with reference to the parameters of memory usage and the time taken by the algorithm.

Keywords: Cryptographic Algorithms, Time and Space Complexity, End-to-End Encryption

1 Introduction

Instant messaging is a utility that allows people to communicate with each other and transfer messages. Even if you don't understand all of the technicalities on how to use the technologies, when you realize these fundamentals, you can make more sense of it. These are all the basic communicating concepts such as, messages transmit through a channel. To transmit this data, the messages are broken down into packets and then passed through the channel. The delivery of the message is simple as it gets sent from the sender and it is received by the receiver. However, actions are performed when the sender sends the messages and receiver receives it. Sometimes the message has to undergo a routing channel before receiving at the receiver, this is because if the communication is taking place between large entities[1]. Users use messaging applications to communicate between each other. Messaging apps like WhatsApp and Facebook Messenger have become an essential part of people's communication standards, and the applications gained little attention as a tool for data collection so far. As this applications have already been installed in most of the smart devices, individuals have become habitual to use this medium to share information on day-to-day basis with one and another [2]. The strength of interchanging the text messages and the

[1] https://www.enterpriseintegrationpatterns.com/MessagingComponentsIntro.html
[2] https://www.tandfonline.com/doi/abs/10.1080/21670811.2020.1864219

frequent use makes it more insecure. When a message is interchanged it is sent from a device and the cellular tower, and it is then sent to the service provider of the receiver and to the device of the user. End-to-end encryption is a secure messaging channel that protects information from being retrieved by external parties. Only sender and the receiver of data can encrypt and decrypt the information with a key once it is sent through internet. By restricting unauthorized users from accessing user information when data is transferred from one origin to another, E2EE can help mitigate risk and secure sensitive data.
The sender encrypts the information with only a secret key before transferring it. The data can only ever be decrypted by a receiver who now has the corresponding key. Asymmetric and symmetric encryption are the two types of algorithms available. Symmetric encryption uses the same key to encrypt and decrypt the information and it is the most commonly used encryption method. Whereas the asymmetric encryption uses two separate keys.
Previous research was focused mainly on evaluating the performance of symmetric and asymmetric algorithms. The researcher (Khalid et al., 2020) analysed the performance of the encryption and decryption schemes with respect to the time complexity. The study conducted in this research shows the Time and Space complexity of the three cryptographic algorithms. The analysis shows the results of the time taken and memory usage of the algorithms and concludes on which algorithm performs best encryption and decryption in terms of time taken and less memory usage. This paper seeks to address the question, which cryptographic algorithm provides end-to-end encryption for secure messaging with less complexity?

2 Related Work

This section gives a detailed study on the Analysis of the Signal Message Protocol and the various algorithms that the researchers have used to conclude their study. This section elaborates on the ideology, techniques, methodology, limitations and the future work that the researchers have used to provide their research. The researchers have used various methodologies for secure messaging and delivering end to end encryption for the signal message protocol. Various limitations and drawbacks from their conclusion are observed and taken into consideration in articulating the research work. The literature work is categorized into four subsections that is the Security analysis of the Signal Messaging Protocol, Secure messaging testing in applications, End-to-End Encryption and Performance analysis of the algorithms. Each subsection also determines the techniques utilized and the results concluded. The research presented is based on the shortcomings of the research previously conducted by analysing the algorithms for end-to-end encryption by calculating the time and space complexity.

2.1 Security Analysis of Signal Messaging Protocol

Security analysis of the signal messaging protocol researched by (Cohn-Gordon et al., 2017) concluded on getting several observations. The first analysis they carried out showed that the Cryptographic core of the Signal gives important security principles. The model that they proposed comprises of the complex properties of the signal and shows the result that signal satisfies the cryptographic assumptions. These properties are nothing but the secrecy and the authentication of the message keys that the signal possess. The second analysis proved to have six different security properties for the message keys. These keys can be used to strengthen the protocol further. On observing the model there were few limitations to the complex Signal Protocol. The limitations were a part of the components in the protocol. Such as the library components which were not the part of the signal. The key verification of the key was out of band. And the out of order messages need to be stored first by the user after arrival.

Security analysis of the closed source signal protocol was proposed by (Diogo Gaspar Alves, 2018) The researcher performed static and dynamic analysis and tested with an environment. To perform the analysis, an application was chosen to perform the tests. With the help of the Android Emulator the environment for testing was setup. The researcher also used Frida to inject the googles engine to perform various tasks on the target as it is a toolkit. At the beginning the researcher started to perform the analysis on the Signal protocol and identified a problem. The problem was overcome by having the testing approach and the results were efficient.

A study conducted by (van Dam, 2019) on Analysing the Signal Protocol concluded that, the researcher studied on two different questions. First question being, what are the different types of signal protocol, and the second question, how to implement the high-level protocol. By analyzing the implementation, the researcher found that the notion of forward secrecy and future secrecy is too weak. From the paper model, the researcher has done manual fuzzing on the Sesame algorithm but resulted that it was not properly implemented. The researcher suggests on to using the fuzzing method on the IOS version and Desktop version. Fuzzing can be implemented on the server side. But to implement from the client side, one needs to mimic the user.

Another similar research was done by (Rubín, 2018) on the Security Analysis of the Signal Protocol. The researcher used the Mathematical aspects of the protocol functionality and also, they showed how it improves the security of cryptographic operations. He also analyzed the Java implementation of the signal libraries which are used in the applications. The paper mainly focused on the functionality, design, and the implementation of the protocols.

A comparison of chat application in terms of Security and Privacy was researched by (Botha, van 't Wout and Leenen, 2019). They have shown a comparison of various chat application and the type of security features they provide. They concluded by saying that people choose different applications by the means of the security features they provide. They have also concluded by saying the users need to apply correct security settings to the applications in order to achieve the desired requirements. They presented a recommendation by stating that the data generated and transmitted by the applications may be vulnerable to privacy. The recommendation presented is in terms of the applications and the type of user settings that need to be done to achieve the security by the user.

A comparison of Secure messaging protocols and implementations researched by (Mujaj, 2017) conducted two analyses. The first analysis resulted in showing the old and new secure messaging protocols that give the encryption and implement the security and privacy properties to the messaging applications. They also observed that the Signal protocol and Matrix protocol both provide security properties but none of them give the entire properties to the applications. The second analyses showed the applications that use these properties and implement them while using the conversations. The application does provide the usability with these protocols but there are still cope for improvements. The suggestion that the researchers have provided for improvements is to use verification for multiple users and also use verification for matrix protocol which will provide, authenticity, confidentiality, forward secrecy.

2.2 Secure messaging testing in applications

Detailed study on Comparative Survey of Secure Instant messaging applications was given by (Johansen, Mujaj, Arshad and Noll, 2021). The discussion that led upon to the study of results for various applications resulted that, the overall test scenario represents a possible improvement that can be made in the applications, which can be verified critically using modelling and verification techniques. The team concluded on the report for two analyses i.e.., secure messaging protocol and the applications that use these protocols. The first analysis shows the end-to-end encryption that the old and new signal messaging protocols offer and also identified security policies. They also concluded by observing that none of the secure messaging protocols could meet the security properties. The signal protocol does not support multiple devices, but the matrix protocol will support. The second analysis was conducted as an experiment on the applications that support the secure messaging protocol with all the security properties.

Another research for Security Analysis Testing for Secure messaging was done by researcher (Candra, Kurniawan and Rhee, 2016) They carried out the testing on the Instant messaging application called Telegram as a case study. The security analysis gave clear evidence of Cryptographic properties such as the Confidentiality, Authentication and Data integrity. The team defined the attack possibilities and the threat model for the Security analysis. The security testing was done on the application with the approach of Static and Dynamic analysis from the aspects acquired from the threat model. They observed from the Dynamic Testing, that the application was providing secure data-on-transit in the network layer but lacked the security aspects in the data storage and cache. Static analysis showed the potential vulnerabilities in the application by implementing the cryptography. The message had broken cryptography in encryption and decryption of message.

Usability and Security of the Secure Mobile messaging was researched by researcher (Schröder, Huber and Wind, 2016). This team was the first to study the security and usability of end-to-end encrypted messengers. They performed an active MITM attack by compromising the key services. From their research they concluded that, man-in-the-middle attack was simulated, and majority of the users failed to detect the attack in the messenger. They proposed recommendation for the messenger which could detect such attacks was, there should be awareness of the security status in the conversations and the verification should be easily accessible.

A comparison on the security analysis between group messaging and the direct message was studied by (Jansen, 2020). The researcher discussed and studied the implementation of the Cryptographic features that are used to build the Signal protocol. On analyzing this the setting and environment for the direct messaging was studied. He connected with multiple devices in a direct and group setting, applying command-line interfaces to Signal's Java implementation of the protocol, and compared the keys that used move forward in the mechanism, and also to encrypt and decode information.

2.3 End-to-End Encryption

Secure communication in GSM networks was studied by (A.El Zouka, 2015). The study conducted to provide a middleware security system for GSM networks. The middleware will provide Confidentiality, Authenticity and Data Integrity to the networks. The model that the researcher has proposed will have lightweight authentication and cryptographic schemes that will minimize the computational power. With the combination of MT authentication and signal messaging protocol, the intrusion detection accuracy was increased. The middleware systems are efficient enough to prevent and minimize the threats that are detected.

A theory proposed on the blockchain enabled end-to-end encryption for the instant messaging applications by (Singh, Nandan and Tewari, 2021). They proposed framework that was implemented by the Google Firebase. They implemented the blockchain functionality on the application with the help of the Docker platform. From their results they concluded that, the proposed framework does not rely on the server for encryption and decryption. The team is currently working on two aspects i.e.., providing the secure group message protocol and testing the scalability of the framework using the blockchain based environment.

2.4 Performance analysis of the algorithms

Performance Evaluation of Cryptographic Algorithms DES and AES was researched by (Kumar Mandal, Parakash and Tiwari, 2012). Researcher has implemented Data Encryption Standard(DES) and Advanced Encryption Standard (AES) in MATLAB software. The performance of the algorithm was calculated on the basis of Avalanche effect, memory used and the time. They concluded that for secure communication the encryption algorithm is important. Memory usage and Time are the important parameters of the encryption algorithm. Results that they evaluated was that DES and AES are the most widely used algorithms. From their results they analysed that the memory consumption of DES is higher, and the avalanche effect of the AES is high. They suggested the future work to be on the images and to improve the security level of the encryption algorithm.

Similar research on Performance of Cryptographic Algorithms based on Time Complexity was researched by (Ali et al., 2020). They presented a research on the analysis between symmetric and asymmetric algorithms with regards to the performance by measuring the encryption schemes and decryption schemes with respect to the time taken to encrypt and decrypt the data. They concluded by deriving the results with the help of real time software cryptool, by analysing the time taken by the cryptographic algorithm depending on different parameters. The conclusion they derived from the experimental analysis is that there is increase in time complexity with the increase in the data for encryption and decryption.

2.5 Research Niche

The table presented below outlines the literature review on the basis of the different approach used. The table also shows the different methodology used and the motivation behind implementing that technique.

Author	Methodology	Results	Limitation	Motivation
(Cohn-Gordon et al., 2017)	Security analysis of the Cryptographic core Signal Protocol	Cryptographic core of Signal Protocol provides few security properties	Limited analysis of the Security in the Signal protocol	Complexity of the Cryptographic core of the protocol could have been used
(Gaspar Alves, 2018)	Static Analysis with Reverse Engineering and Dynamic Analysis with Frida	Security flaws existing in communication	The code used for implementation contained decomplication error	Observing the erratic behaviour of the algorithm and the signal protocol
(Van Dam, 2019)	Manual fuzzing on the Sesame algorithm	Session recovery is not implemented in the Sesame Algorithm	Manual fuzzing could not implement the entire Sesame algorithm	Protocol state fuzzing can be used in different versions
(Rubín, 2018)	Mathematical aspects of the protocol functionality and signal libraries in Java	Few inconsistencies and operations different from the documented were found in the implementation.	Sesame algorithm and session management in multiple devices could not be implemented	Other languages libraries can be used to analyze the Signal Protocol such as C and JavaScript
(Botha, Wout and Veenen, 2019)	A comparison of various chat application and the type of security features they provide	People choose different applications by the means of the security features they provide.	Comparing the applications resulted in showing that the free applications have more privacy issues	Paid applications provide more security and end-to-end encryption in the applications
(Mujaj, 2017)	The first analysis showed the old and new secure messaging protocols. The second analyses showed the applications that use the properties and	The analysis observed that the Signal protocol and Matrix protocol both provide security properties but none of them give the entire properties to the	The research could be redone with multiple participations to ensure a more generalized result of the test cases.	To use verification for multiple users and also use verification for matrix protocol which will provide, authenticity, confidentiality, forward secrecy.

	implement them while using the conversations	applications		
(Johansen, Mujaj, Arshad and Noll, 2021)	Security analysis of the secure messaging protocol and the applications that use these protocols	The analysis shows the end-to-end encryption that the old and new signal messaging protocols offer and also identified security policies. The signal protocol does not support multiple devices, but the matrix protocol will support	One test scenario was implemented and by observing that none of the secure messaging protocols could meet the security properties	Modelling and verification techniques can be used to improve the security properties
(Candra, Kurniawan and Rhee, 2016)	Testing on the Instant messaging application called Telegram as a case study through static and dynamic analysis	The dynamic testing showed the application was providing secure data-on-transit in the network layer but lacked the security aspects in the data storage and cache. Static analysis showed the potential vulnerabilities in the application by implementing the cryptography.	Threat modelling was used to derive the security aspects for testing	Static and Dynamic analysis for security aspects can be used on other applications as a case study
(Schröder, Huber and Wind, 2016)	An active MITM attack was performed by compromising the key services	Man-in-the-middle attack was simulated, and majority of the users failed to detect the	Participants used to conduct the study were homogeneous and each had to verify first in	There should be a messenger who could detect such attacks and also there should be awareness of the

		attack in the messenger	order to conduct the testing	security status in the conversations and the verification should be easily accessible.
(Jansen, 2020)	Discussed and studied the implementation of the Cryptographic features that are used to build the Signal protocol	The protocol for group messaging rests upon several executions of the direct messaging algorithm	Sending messages to the destination resulted in finding symmetric ratchets and no asymmetric ratchet was found	Replicate the testing methodology over various applications and implement on different versions
(A.El Zouka, 2015)	The model that the researcher has proposed will have lightweight authentication and cryptographic schemes that will minimize the computational power	With the combination of MT authentication and signal messaging protocol, the intrusion detection accuracy was increased	The proposed system does not give results when used with the mobile terminals.	Middleware systems should be used in mobile systems to analyze the other security protocols
(Singh, Singh Chauhan and Tewari, 2021)	Implemented the blockchain functionality on the application with the help of the Docker platform	The proposed framework does not rely on the server for encryption and decryption	Could not provide the secure group message protocol and testing the scalability of the framework using the blockchain based environment	The framework can be used against the scalability of blockchain in terms of fetching time and verification time
(Kumar Mandal, Parakash and Tiwari, 2012)	Implemented Data Encryption Standard(DES)	The memory consumption of DES is higher,	Other parameters and algorithms could	The future work to be on the images and to improve the

	and Advanced Encryption Standard (AES) in MATLAB software	and the avalanche effect of the AES is high	have been used to measure the parameters	security level of the encryption algorithm.
(Ali et al., 2020)	Analysis between symmetric and asymmetric algorithms with regards to the performance by measuring the encryption schemes and decryption schemes	There is increase in time complexity with the increase in the data for encryption and decryption	Time complexity and key size can be reused again for another algorithms	Different of algorithms and parameters can be used to measure the time complexity

From the related work it is concluded that, many researchers have researched in the field of end-to-end encryption for symmetric and asymmetric algorithm by using the methodology of testing the security parameters on the applications. In this research, the time and space complexity of the cryptographic algorithm is calculated to identify which algorithm is the best for encryption and decryption.

3 Research Methodology

The methodology approach used in this paper is based on the extension from the previous work done in the related field. In the present methodology, the comparison and analysis for the key exchanges taking place for secure messaging is examined. The first analysis is done through taking the Public and Private keys generated by the algorithm and calculating the Time Complexity. Time Complexity of any algorithm is calculated by the time taken by the algorithm to run with respect to the input. Second analysis is performed by taking the input as Public and Private keys from the algorithm and measuring the memory that the algorithm takes upon the input. Space Complexity of an algorithm will determine the amount of memory taken to run with reference to the input. The purpose of this research is to determine which algorithm has the best Time and Space complexity to provide End-to-End encryption for secure messaging.

Time and Space complexity of the three cryptographic algorithms are calculated in this research. The three cryptographic algorithms are: AES, RSA and Double-Ratchet Algorithms.

3.1 Advanced Encryption Standard (AES) Algorithm

Advanced Encryption Standard (AES) is a symmetric block cipher used for protecting the personal and private information. AES is implemented to encrypt the subtle data. The AES encryption works using three block ciphers i.e., AES-128, AES-192, and AES-256. Encryption and Decryption of data is done in 128 bits of blocks using the keys of 128, 192 and 256. It is based on the substitution-permutation concepts. It embraces a stream of operations, some involve replacing the inputs with outputs and others involve rearranging the bits. AES performs all its operations on bytes instead of bits. Hence 128 bits of a plaintext is treated as 16 bytes. Therefore, for processing the 16 bytes, the data is arranged in four columns and four rows.

3.1.1 Encryption

AES Encryption operation consists of four sub-operations. The first-round process consists of substituting the 16 bytes of data by looking at the fixed table. The result is placed in four rows and four columns. Second-round process consists of shifting each rows to the left. The first row is not shifted to the left, whereas the second row is shifted one byte to the left and the third row is shifted two byte to the left. And the process continues to the fourth row. Third-round process consists of mixing the columns. Each column is now converted into a mathematical function. This is done by taking the four bytes as inputs and giving the completely new output of four bytes, which will replace the original column. The last round will consider the 128 bits as the 16 bytes and by XORing the bits to the round key the ciphertext is obtained at the output.

3.1.2 Decryption

AES Decryption operation takes place similar to that of the AES Encryption process but in the reverse approach. The four operation is as follows:

- Adding the round keys
- Mixing of the columns
- Shifting the rows
- Substitution of bytes

Encryption and Decryption of the sub-operations need to be separately implemented although they are linked with each other.

3.2 Rivest, Shamir, Adleman(RSA) Algorithm

RSA algorithm is an asymmetric cryptography algorithm, that means it uses two keys i.e., public key and private key. These two keys are mathematically linked keys, and they are different from each other. As per their names, the public key is shared publicly but the private key is not shared publicly and kept private. The measure feature of RSA algorithm is that the key is kept as secured as possible.

3.2.1 Key Generation

- Generate any two prime numbers A and B. Such that the prime numbers should be large.
- Calculate $n=A\mathrm{x}B$ and $\phi= (A-1) \mathrm{x} (B-1)$
- Choose any integer e, $1<e< \phi$
- Choose an exponent d, $1<d< \phi$
- Hence the public key generated will be (n,e) and the private key (d,A,B)

3.2.2 Encryption

- Sender A acquires the public key of the receiver B (n,e)
- Plaintext message is represented as a positive integer $1<m<n$
- Ciphertext is calculated by c=m^e mod n
- Ciphertext is sent to the receiver B

3.2.3 Decryption

- The receiver B uses the private key (n,d)
- Plaintext is calculated by m= c^d mod n

3.3 Double-Ratchet Algorithm

The Double-Ratchet algorithm works and is used between two users where they exchange the messages with the help of the secret key. The two users will use the key management to admit on the shared private key. The Double-Ratchet algorithm will be used to send and receive the messages. Every time new keys will be generated for the users with the Double-Ratchet messages such that the earlier key generated will not be able to calculate. The users also can send the Diffie-Hellman public keys attached with the messages. The results from DH are mixed up with the keys so that the later keys generated will not be able to be calculated. Such properties help in providing the protection to the encrypted messages. KDF chain is the primary theory of the Double-Ratchet algorithm. The KDF chain is a cryptographic function that takes two keys i.e., Secret and the random key along with input data and will generate the output data. The Double-Ratchet is a combination of symmetric-key and DH ratchets. When a message is sent or received, the symmetric key ratchet is enforced to the receiving or the sending chain to get the message. When a public key is received a DH ratchet is performed before to the symmetric key ratchet to replace the chain of keys

4 Design Specification

Recently the researchers have used real time software’s to measure the real time results of the cryptographic algorithms. The results have been found on the basis of the memory size and processing time of the algorithms by selecting different parameters. Performance has been studied on the basis of DES, AES and RSA algorithms. This research comprises of AES, RSA and Double-Ratchet algorithms and their Time and Space complexity which has been calculated.
The below design flow represents the structure for calculating the complexity of the cryptographic algorithms. In this research three cryptographic algorithms were chosen to provide end-to-end encryption for secure messaging. End-to-end encryption is the event of having messages encrypted such that, only to the user that the message is sent will be received and decrypted. The message travels in a secure way providing a secure communication. The major benefit of having end-to-end encryption is that, only the receiver that the message is intended to be will receive the message. To perform the encryption and decryption, three cryptographic algorithms have been selected. The three algorithms namely, AES, RSA and Double-Ratchet algorithms. Each algorithm uses a public key and private key to encrypt and decrypt the message. Time and Space complexity of the algorithms is measured with respect to the two keys while encrypting and decrypting the message, also on the memory that the message consumes. Further on getting the results from the three algorithms, the memory usage and time taken by the algorithms has been compared and used to analyse, whether which algorithm will decrypt the message in less memory usage and with less time difference.

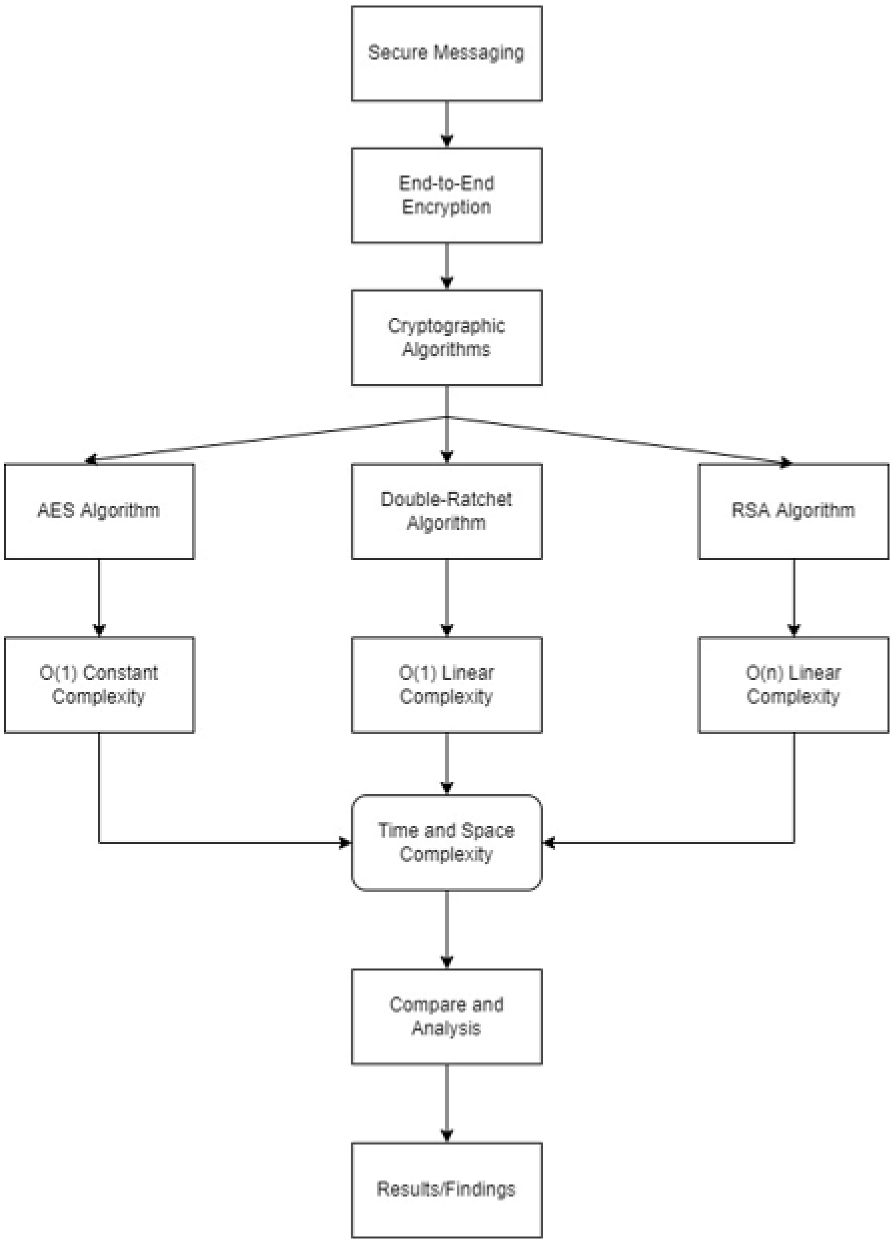


Figure 1: Design flow of the implementation

4.1 Performance Analysis

The necessity of performance analysis of algorithms is required to measure the complexity of the algorithms. It helps the researchers to decide and assess which algorithm suites best for encryption. In this research the analysis is done using the Time and Space complexity of the algorithms. The complexity is measured at the time of encryption and decryption, and the result is measured by taking the difference. Time complexity is the amount of time an algorithm requires to run. Space complexity is the amount of memory space required by the algorithm to run.

4.1.1 Big_O Notation

Big_O notation is the most common standard for calculating the time complexity of an algorithm. It represents the implementation time of a task with respect to the number events that are required to complete the given task. The notation is denoted in the form of O(n) where, O stands for order of magnitude and n represents the parameter that it is being compared to. The task can be done using many algorithms each having different complexity. The common complexities that the algorithm have are, Constant "O(1)", Linear "O(N)", Quadratic "O(N^2)", Logarithmic "O(LogN)" [3].

4.1.2 Comparative approach to calculate Big_O complexity

The comparative approach can be illustrated by the following example for all the algorithms:

Return the character mostly used in the string
maxChar("I loveeee noodles") ==>"**e**"

maxChar complexity analysis
Time complexity- O(n) every character in the string is visited (for loop)
Space complexity- O(1) character count will have minimum 26-key pairs (execution is consistent)

AES algorithm and Double-Ratchet algorithm in the research, are working on the fixed size of the block, so they have the complexity of O(1). Because approximately they take the same amount of time independent of the input. "O(1)" means it will take maximum of 14 nanoseconds to run independent of the data.
RSA algorithm in this research has varying size of input as it uses public key and private key. Both have the key "n" and there size be "k" in bits. Therefore, the complexity is O(n). "O(n)" indicates the time taken by the algorithm will increase linearly with the data.

5 Implementation

This section represents the implementation of Time and Space complexity in the three cryptographic algorithm. The experimental environment utilized is covered in this section. The coding packages are also described.

5.1 Hardware

The environment setup was built on the following specifications:

- Processor: Intel i5-7200 CPU
- RAM: 8192MB
- Operating System: Windows 10 64-bit

5.2 Software

The following software was used on the windows 10 operating system to build the algorithms:

[3] https://towardsdatascience.com/the-big-o-notation-d35d52f38134

- PyCharm: PyCharm is an Integrated Development Environment which is used to code the algorithm.
- Python: Python language was used to code the algorithm.

5.3 Libraries

The following libraries was used for the implementation:

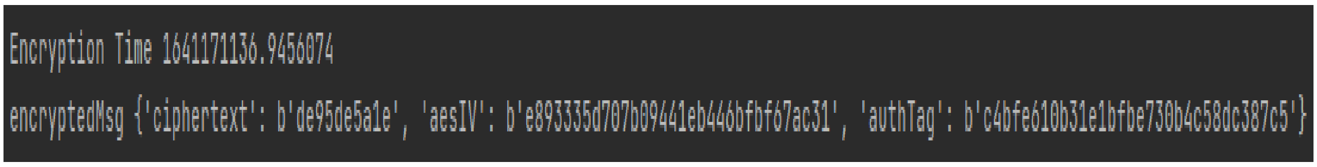
- Big_O: Big_O is a python library used to calculate the approximate time complexity of the code from the execution.
- Time- Time is a function in python which is used to express time in the code.
- Memory_Profiler: Memory Profiler is a python library to observe the memory consumption as line by line in the python program.
- Base64- Base64 is a python library used to encode to decode the data.
- Crypto.Cipher- Crypto.Cipher is a package that contains algorithms that protect the data.

6 Evaluation

This section elaborates the performance evaluation of the AES, RSA and Double-Ratchet algorithms. The performance was calculated using the Time and Space complexity of the algorithms. The performance analysis was also calculated with the help of graphical plot. This section gives the overall results of the implementation followed in the research. The study and analysis of any algorithm is performed by measuring the time and space complexity which is also the performance analysis.

6.1 Advanced Encryption Standard(AES) Algorithm

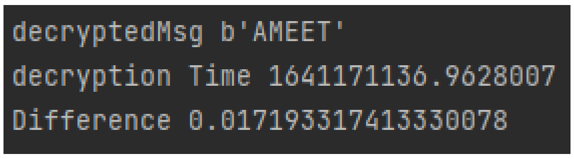
1. Encryption Time: The Encryption time of the algorithm is calculated by the total plaintext in bytes divided by the encryption time in milliseconds.
   In this algorithm, the encryption time was 16411.945 mS.



```
Encryption Time 1641171136.9456074
encryptedMsg {'ciphertext': b'de95de5a1e', 'aesIV': b'e893335d707b09441eb446bfbf67ac31', 'authTag': b'c4bfe610b31e1bfbe730b4c58dc387c5'}
```

Figure 2: Encryption time and Encrypted message

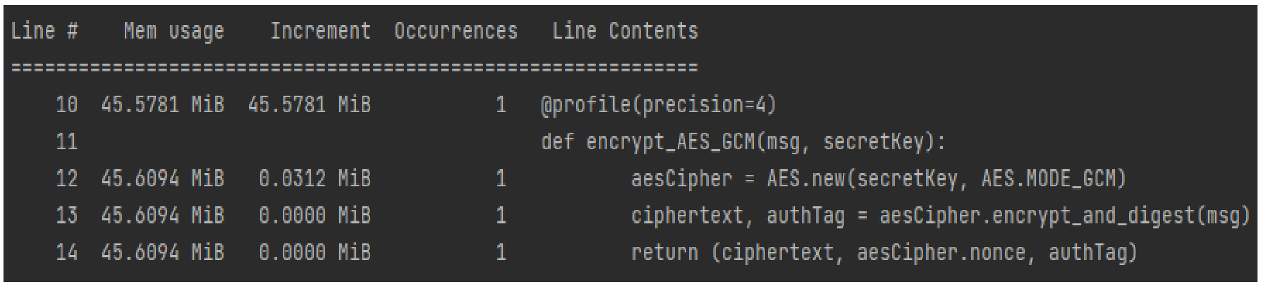
2. Decryption time: Decryption time of the algorithm is the time required to convert the ciphertext to the plaintext in millisecond.
   In this algorithm, the decryption time is 16411.962 mS.
   The difference in time is also calculated by the difference in decryption time and encryption time. In this algorithm, the difference is 0.01719 mS.



```
decryptedMsg b'AMEET'
decryption Time 1641171136.9628007
Difference 0.017193317413330078
```

Figure 3: Decryption time, decrypted message, and difference in time

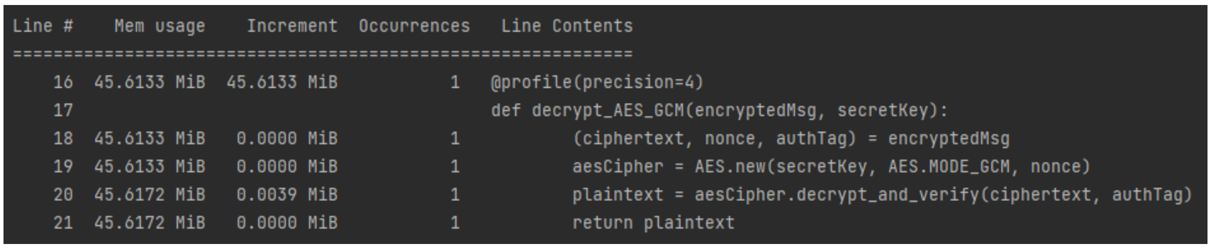
3. Memory Usage: The memory usage is the amount of space required by the algorithm to produce the result.
The memory usage at the time of encryption is as shown in the figure below.



```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    10  45.5781 MiB  45.5781 MiB           1   @profile(precision=4)
    11                                         def encrypt_AES_GCM(msg, secretKey):
    12  45.6094 MiB   0.0312 MiB           1           aesCipher = AES.new(secretKey, AES.MODE_GCM)
    13  45.6094 MiB   0.0000 MiB           1           ciphertext, authTag = aesCipher.encrypt_and_digest(msg)
    14  45.6094 MiB   0.0000 MiB           1           return (ciphertext, aesCipher.nonce, authTag)
```

Figure 4: Line-by-line analysis for memory usage

4. Memory Usage: The memory usage is the amount of space required by the algorithm to produce the result.
The memory usage at the time of decryption is as shown in the figure below.

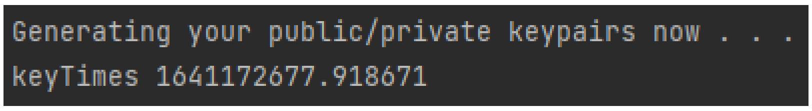


```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    16  45.6133 MiB  45.6133 MiB           1   @profile(precision=4)
    17                                         def decrypt_AES_GCM(encryptedMsg, secretKey):
    18  45.6133 MiB   0.0000 MiB           1           (ciphertext, nonce, authTag) = encryptedMsg
    19  45.6133 MiB   0.0000 MiB           1           aesCipher = AES.new(secretKey, AES.MODE_GCM, nonce)
    20  45.6172 MiB   0.0039 MiB           1           plaintext = aesCipher.decrypt_and_verify(ciphertext, authTag)
    21  45.6172 MiB   0.0000 MiB           1           return plaintext
```

Figure 5: Memory usage for decryption of the message

6.2 Rivest, Shamir, Adleman(RSA) Algorithm:

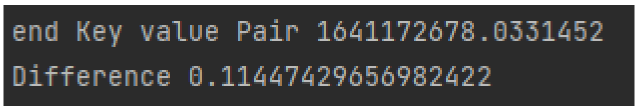
1. Encryption Time: The Encryption time of the algorithm is calculated by the total plaintext in bytes divided by the encryption time in milliseconds.
   In this algorithm, the encryption time was 16411.918 mS.



```
Generating your public/private keypairs now . . .
keyTimes 1641172677.918671
```

Figure 6: Encryption time of generating key pairs

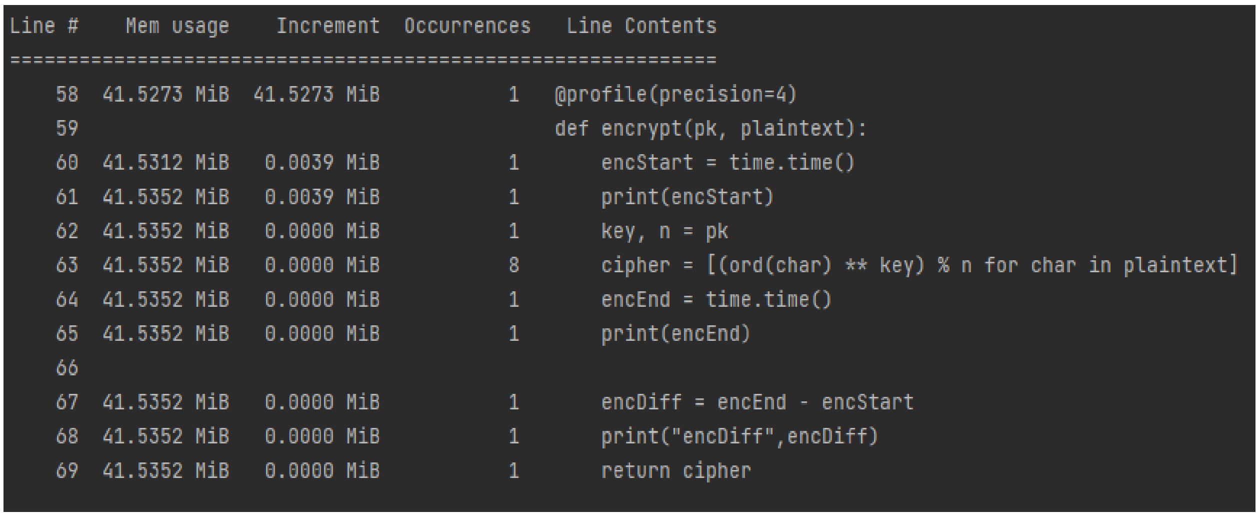
2. Decryption Time: Decryption time of the algorithm is the time required to convert the ciphertext to the plaintext in millisecond.
   In this algorithm, the decryption time at the end of key pairs is 16411.033 mS.
   The difference in time is also calculated by the difference in decryption time and encryption time. In this algorithm, the difference is 0.1144 mS.



```
end Key value Pair 1641172678.0331452
Difference 0.11447429656982422
```

Figure 7: Decryption time at end of key pairs

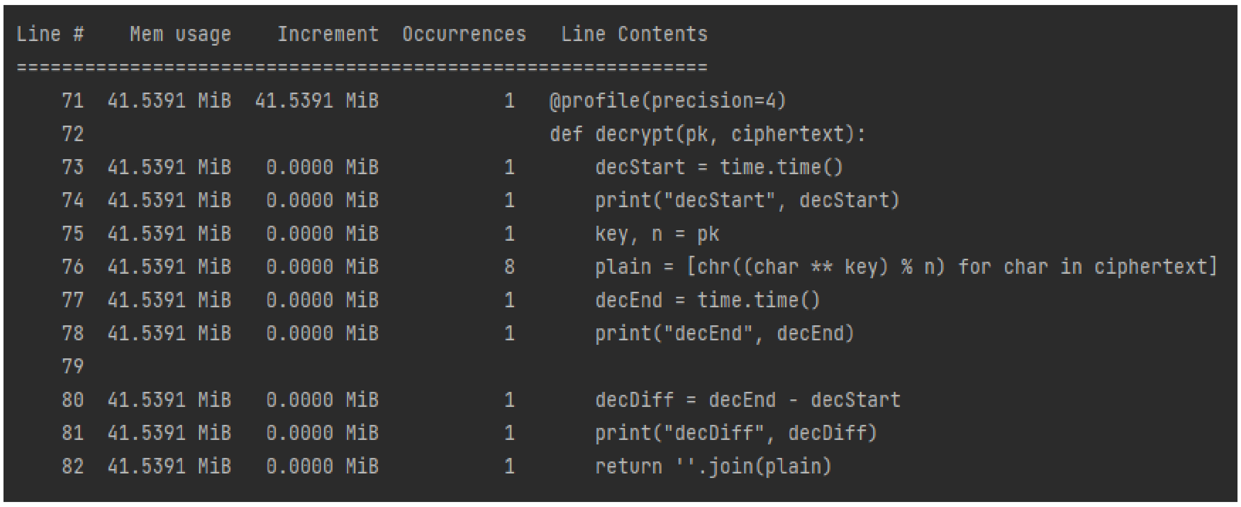
3. Memory Usage: The memory usage is the amount of space required by the algorithm to produce the result.
   The memory usage at the time of encryption is as shown in the figure below.



```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    58  41.5273 MiB  41.5273 MiB           1   @profile(precision=4)
    59                                         def encrypt(pk, plaintext):
    60  41.5312 MiB   0.0039 MiB           1       encStart = time.time()
    61  41.5352 MiB   0.0039 MiB           1       print(encStart)
    62  41.5352 MiB   0.0000 MiB           1       key, n = pk
    63  41.5352 MiB   0.0000 MiB           8       cipher = [(ord(char) ** key) % n for char in plaintext]
    64  41.5352 MiB   0.0000 MiB           1       encEnd = time.time()
    65  41.5352 MiB   0.0000 MiB           1       print(encEnd)
    66
    67  41.5352 MiB   0.0000 MiB           1       encDiff = encEnd - encStart
    68  41.5352 MiB   0.0000 MiB           1       print("encDiff",encDiff)
    69  41.5352 MiB   0.0000 MiB           1       return cipher
```

Figure 8: Memory usage of encryption and time difference

4. Memory Usage: The memory usage is the amount of space required by the algorithm to produce the result.
   The memory usage at the time of decryption is as shown in the figure below.

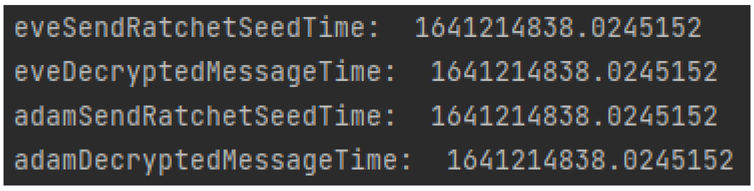


```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    71  41.5391 MiB  41.5391 MiB           1   @profile(precision=4)
    72                                         def decrypt(pk, ciphertext):
    73  41.5391 MiB   0.0000 MiB           1       decStart = time.time()
    74  41.5391 MiB   0.0000 MiB           1       print("decStart", decStart)
    75  41.5391 MiB   0.0000 MiB           1       key, n = pk
    76  41.5391 MiB   0.0000 MiB           8       plain = [chr((char ** key) % n) for char in ciphertext]
    77  41.5391 MiB   0.0000 MiB           1       decEnd = time.time()
    78  41.5391 MiB   0.0000 MiB           1       print("decEnd", decEnd)
    79
    80  41.5391 MiB   0.0000 MiB           1       decDiff = decEnd - decStart
    81  41.5391 MiB   0.0000 MiB           1       print("decDiff", decDiff)
    82  41.5391 MiB   0.0000 MiB           1       return ''.join(plain)
```

Figure 9: Memory usage of decryption

6.3 Double-Ratchet Algorithm:

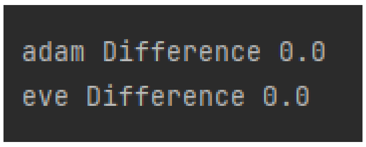
1. The Encryption time of the algorithm is calculated by the total plaintext in bytes divided by the encryption time in milliseconds.
   In this algorithm, there are two ratchets i.e.., sending ratchet and receiving ratchet. As seen from the below figure, there are two users, and each have their sending time and decrypted message time.



```
eveSendRatchetSeedTime:  1641214838.0245152
eveDecryptedMessageTime:  1641214838.0245152
adamSendRatchetSeedTime:  1641214838.0245152
adamDecryptedMessageTime:  1641214838.0245152
```

Figure 10: Ratchet seed time and decrypted time

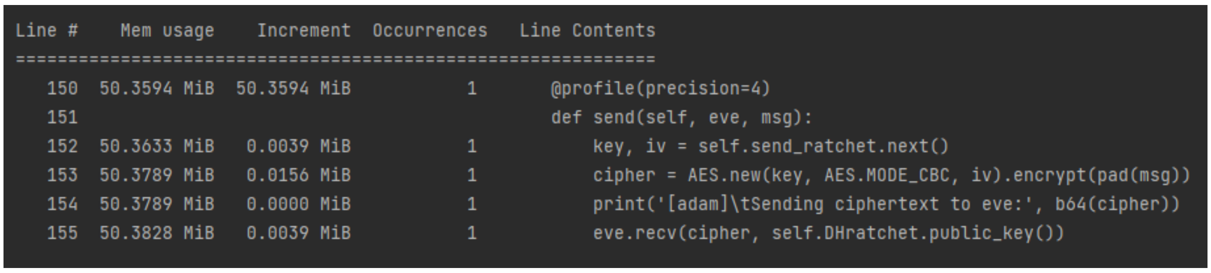
2. Decryption Time: Decryption time of the algorithm is the time required by the ciphertext to the plaintext in millisecond.
   In this algorithm, the difference in time is also calculated by the difference in decryption time and encryption time. In this algorithm, the difference is 0.0 mS.



```
adam Difference 0.0
eve Difference 0.0
```

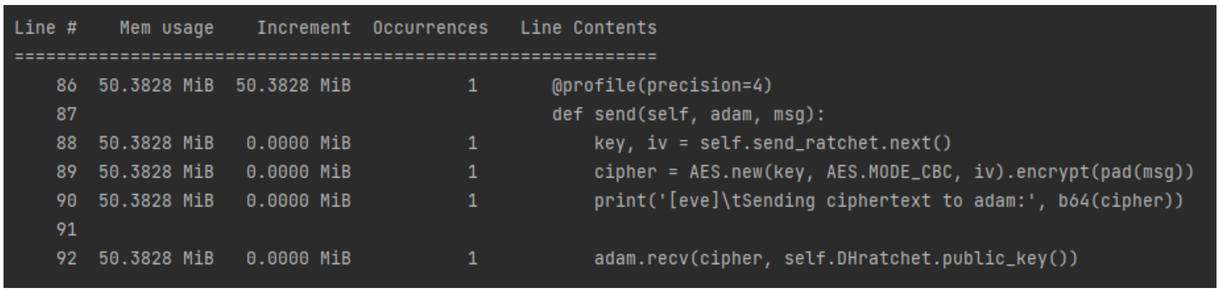
Figure 11: Difference in time calculated for encryption and decryption

3. Memory Usage: The memory usage is the amount of space required by the algorithm to produce the result.
   The memory usage at the time of encryption is as shown in the figure below.



```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
   150  50.3594 MiB  50.3594 MiB           1       @profile(precision=4)
   151                                             def send(self, eve, msg):
   152  50.3633 MiB   0.0039 MiB           1           key, iv = self.send_ratchet.next()
   153  50.3789 MiB   0.0156 MiB           1           cipher = AES.new(key, AES.MODE_CBC, iv).encrypt(pad(msg))
   154  50.3789 MiB   0.0000 MiB           1           print('[adam]\tSending ciphertext to eve:', b64(cipher))
   155  50.3828 MiB   0.0039 MiB           1           eve.recv(cipher, self.DHratchet.public_key())
```

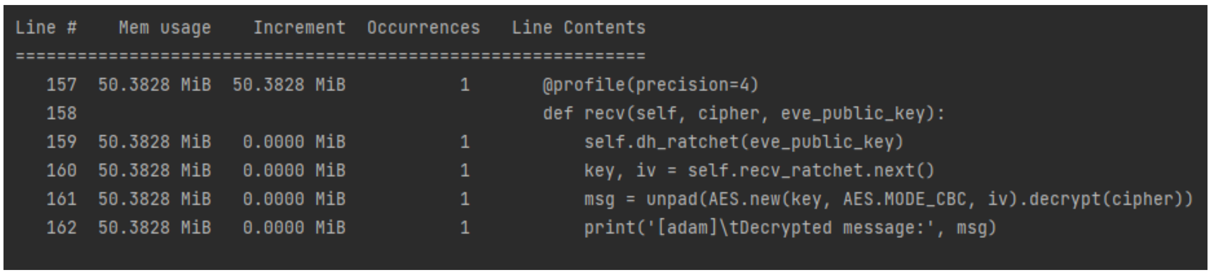
Figure 12: Memory usage of Eve sending the message



```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    86  50.3828 MiB  50.3828 MiB           1       @profile(precision=4)
    87                                             def send(self, adam, msg):
    88  50.3828 MiB   0.0000 MiB           1           key, iv = self.send_ratchet.next()
    89  50.3828 MiB   0.0000 MiB           1           cipher = AES.new(key, AES.MODE_CBC, iv).encrypt(pad(msg))
    90  50.3828 MiB   0.0000 MiB           1           print('[eve]\tSending ciphertext to adam:', b64(cipher))
    91
    92  50.3828 MiB   0.0000 MiB           1           adam.recv(cipher, self.DHratchet.public_key())
```

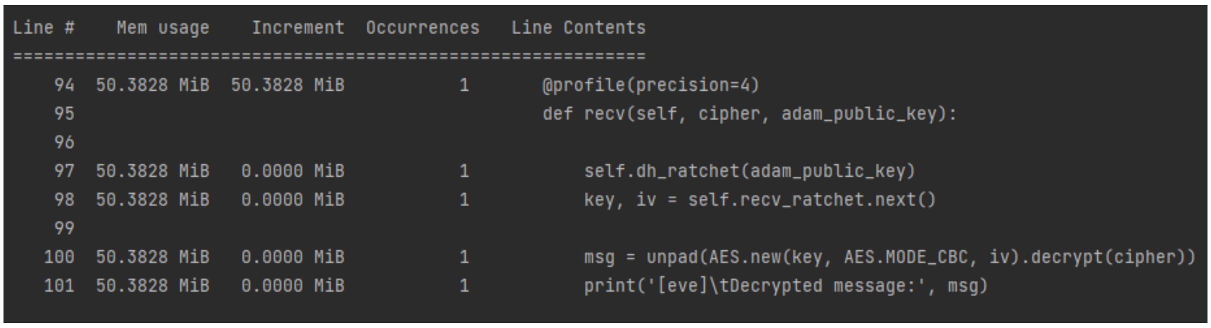
Figure 13: Memory usage of Adam sending the message

4. Memory Usage: The memory usage is the amount of space required by the algorithm to produce the result.
   The memory usage at the time of decryption is as shown in the figure below.



```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
   157  50.3828 MiB  50.3828 MiB           1       @profile(precision=4)
   158                                             def recv(self, cipher, eve_public_key):
   159  50.3828 MiB   0.0000 MiB           1           self.dh_ratchet(eve_public_key)
   160  50.3828 MiB   0.0000 MiB           1           key, iv = self.recv_ratchet.next()
   161  50.3828 MiB   0.0000 MiB           1           msg = unpad(AES.new(key, AES.MODE_CBC, iv).decrypt(cipher))
   162  50.3828 MiB   0.0000 MiB           1           print('[adam]\tDecrypted message:', msg)
```

Figure 14: Memory usage of Adam decrypting the message



```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    94  50.3828 MiB  50.3828 MiB           1       @profile(precision=4)
    95                                             def recv(self, cipher, adam_public_key):
    96
    97  50.3828 MiB   0.0000 MiB           1           self.dh_ratchet(adam_public_key)
    98  50.3828 MiB   0.0000 MiB           1           key, iv = self.recv_ratchet.next()
    99
   100  50.3828 MiB   0.0000 MiB           1           msg = unpad(AES.new(key, AES.MODE_CBC, iv).decrypt(cipher))
   101  50.3828 MiB   0.0000 MiB           1           print('[eve]\tDecrypted message:', msg)
```

Figure 15: Memory usage of Eve decrypted the message

6.4 Experiment 1:

Found the Time and Space complexity of the AES Algorithm which is represented graphically as shown in the figure below.

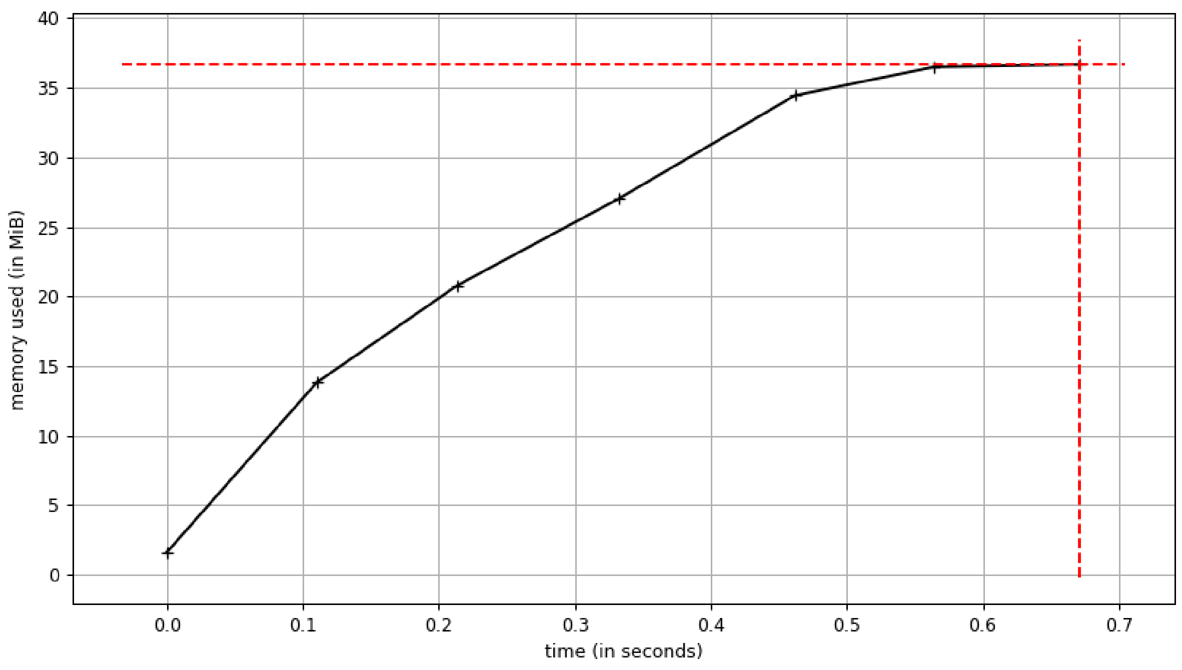


Figure 16: Plot of complexity time vs memory

6.5 Experiment 2:

Found the Time and Space complexity of the RSA Algorithm which is represented graphically as shown in the figure below.

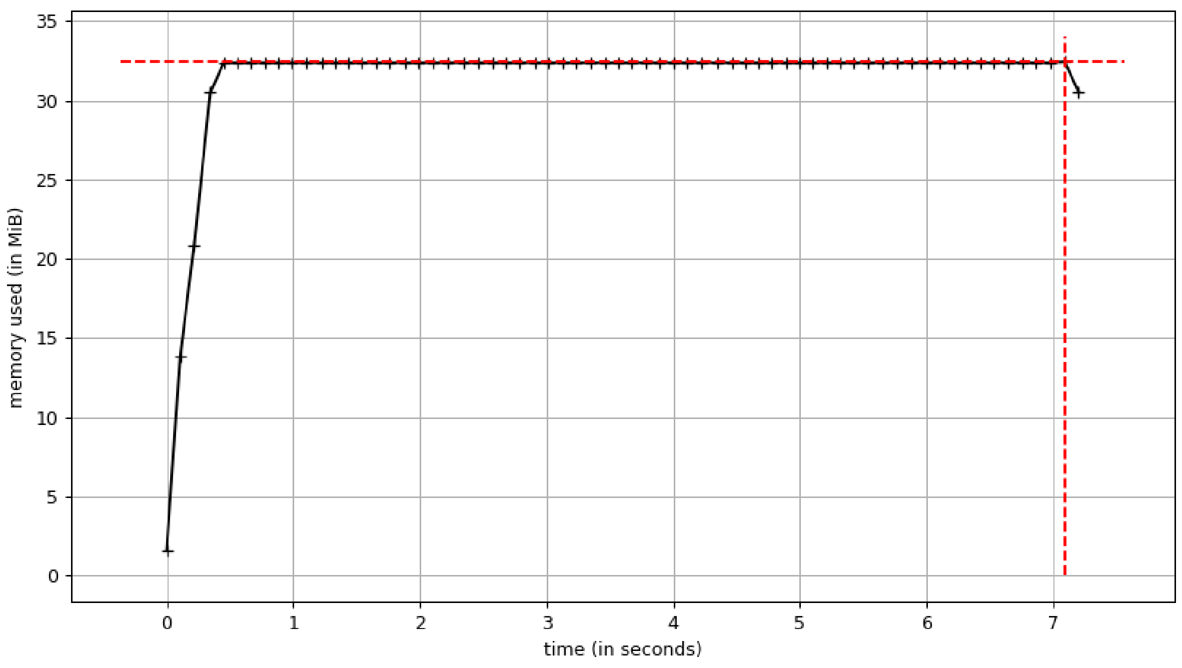


Figure 17: Plot of complexity time vs memory

6.6 Experiment 3:

Found the Time and Space complexity of the Double-Ratchet Algorithm which is represented graphically as shown in the figure below.

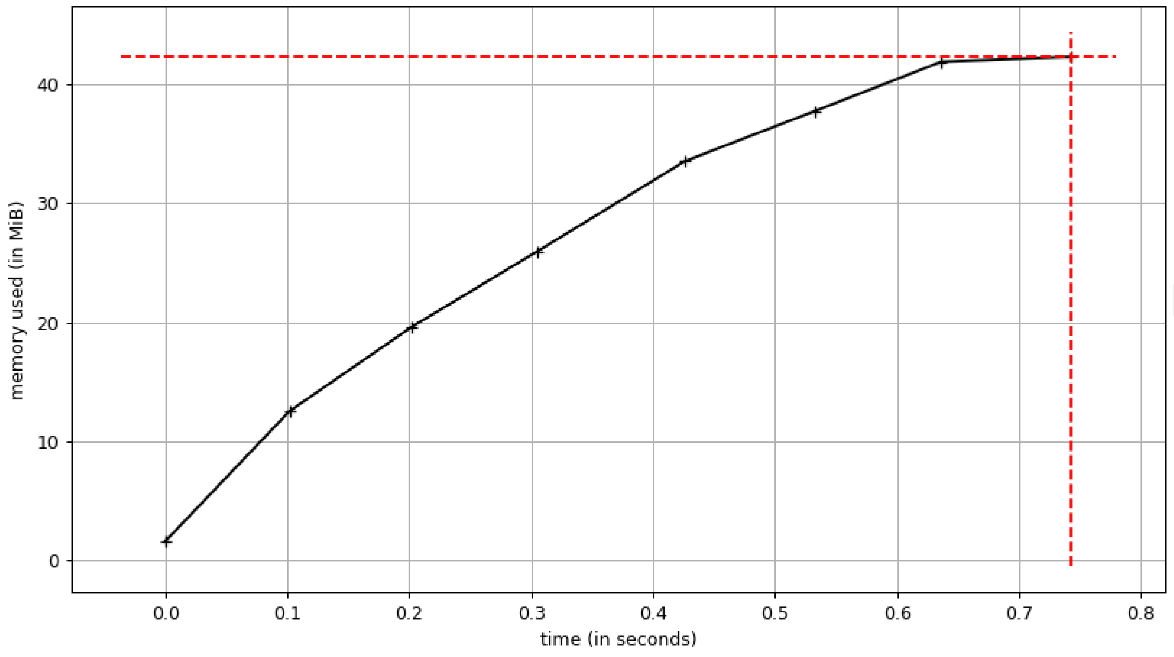


Figure 18: Plot of complexity time vs memory

6.7 Discussion

The research has been implemented and analysed by the complexity of the three cryptographic algorithms. On analysing the performance analysis of the complexity of the algorithms it is observed that the AES algorithm shows a difference in time and has some amount of memory usage while encryption and decryption of message. RSA algorithm has a difference in time while encryption and decryption. Whereas there is a slight increase in the memory usage while encryption but showed no increase while decryption. Out of the three-cryptography algorithm, Double-Ratchet algorithm proved to be the best algorithm while encryption and decryption of the message. As this algorithm showed no difference while encrypting and decrypting the message in the time taken and memory usage. Drawing from the results we can conclude that Double-Ratchet algorithm has proven to provide the end-to-end encryption for secure messaging, as this algorithm protects from the attackers from receiving the clear text as the message is decrypted instantly after sending.

Algorithm	Mathematical Time Taken	Experimental Time Taken	Experimental Memory Usage
AES algorithm (Encryption)	O(1) Constant	16411.945 mS	45.6094 Mib
AES algorithm (Decryption)	O(1) Constant	16411.962 mS	45.6172 Mib
RSA algorithm (Encryption)	O(n) Linear	16411.918 mS	41.5352 Mib
RSA algorithm	O(n) Linear	16411.033 mS	41.5391 Mib

(Decryption)			
Double-Ratchet(Encryption)	O(1) Constant	16412.024 mS	50.3828 Mib
Double-Ratchet(Decryption)	O(1) Constant	16412.024 mS	50.3828 Mib

7 Conclusion and Future Work

The main objective of this research is to determine which cryptographic algorithm out of the three algorithms provides better encryption and decryption. Previously researchers have researched the end-to-end encryption provided by the algorithms and have shown the analysis of two cryptographic algorithms. This research focused onto calculating the complexity of the cryptographic algorithm and stating which algorithm gives the best-case complexity for encryption and decryption of the message. The results of the study have shown that the Double-Ratchet algorithm has encrypted and decrypted the message with no difference, whereas AES and RSA algorithm have shown some difference in encryption and decryption. In the future work, more symmetric and asymmetric algorithms can be used to analyse the complexity. While analysing different set of parameters can be considered and complexity can be measured. Also, a real-time analysis with a real-time software can be implemented and tested to analyse the time and space complexity of the algorithms.

References

Cohn-Gordon, K., Cremersy, C., Dowlingz, B., Garrattx, L. and Stebila, D., 2017. A Formal Security Analysis of the Signal Messaging Protocol. In: 2017 IEEE European Symposium on Security and Privacy. Zurich: IEEE, p.46.

Diogo Gaspar Alves, J., 2018. Security Analysis of a Closed-Source Signal Protocol Implementation. Computer Science, p.88.

van Dam, D., 2019. Analysing the Signal Protocol. Deloitte: IEEE, p.71.

Rubín, J., 2018. Security Analysis of the Signal Protocol. In: Computer Science. Prague: IEEE, p.85.

Botha, J., van 't Wout, C. and Leenen, L., 2019. A Comparison of Chat Applications in Terms of Security and Privacy. In: A Comparison of Chat Applications in terms of Security and Privacy. Coimbra: IEEE, p.8.

Mujaj, A., 2017. A Comparison of Secure Messaging Protocols and Implementations. In: Computer Science. Oslo: IEEE, p.129.

Johansen, C., Mujaj, A., Arshad, H. and Noll, J., 2021. The Snowden Phone: A Comparative Survey of Secure Instant Messaging Mobile Applications. Security and Communication Networks, 2021, pp.1-30.

Candra, A., Kurniawan, Y. and Rhee, K., 2016. Security analysis testing for secure instant messaging in android with study case: Telegram. In: 6th International Conference on System Engineering and Technology (ICSET). Bandung: IEEE, p.5.

Schroder, S., Huber, M., Wind, D. and Rottermanner, C., 2016. When SIGNAL hits the Fan: On the Usability and Security of State-of-the-Art Secure Mobile Messaging. In: 1st European Workshop on Usable Security. Darmstadt: IEEE, p.7.

Jansen, M., 2020. A Security Analysis of the Signal Protocol's Group Messaging Capabilities in Comparison to Direct Messaging. Computer Science, p.29.

A.El Zouka, H., 2015. Providing End-To-End Secure Communications in GSM Networks. International Journal of Network Security & Its Applications, 7(4), pp.31-41.

Singh, R., Nandan, A. and Tewari, H., 2021. the blockchain enabled end-to-end encryption for the instant messaging applications. Dublin: IEEE, p.7.

Kumar Mandal, A., Parakash, C. and Tiwari, A., 2012. Performance evaluation of cryptographic algorithms: DES and AES. In: IEEE Students' Conference on Electrical, Electronics and Computer Science. Bhopal: IEEE, p.5.

Ali, K., Akhtar, F., Ahmed Memon, S., Shakeel, A., Ali, A. and Raheem, A., 2020. Performance of Cryptographic Algorithms based on Time Complexity. In: 3rd International Conference on Computing, Mathematics and Engineering Technologies. Sukkur: IEEE, p.5.