File Transfer on Cloud using Diffie-Hellman Key Exchange in Conjunction with AES Encryption

MSc Research Project
Cloud Computing

Ravi Deokar
Student ID: x20207077

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Ravi Deokar
Student ID:	x20207077
Programme:	Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	15/12/2022
Project Title:	File Transfer on Cloud using Diffie-Hellman Key Exchange in Conjunction with AES Encryption
Word Count:	8878
Page Count:	22

Signature:	
Date:	28th January 2023

File Transfer on Cloud using Diffie-Hellman Key Exchange in Conjunction with AES Encryption

Ravi Deokar
x20207077

Abstract

The advent of internet communications has been widely adopted in recent times, with major advancements being made in the field of technology. With such advances in technology evolving at a faster pace, the role of online platforms to fulfill the purpose of remote communications has been successfully established. In light of this fact, communication ought to be established over a wireless channel involving two different parties that wish to send a private message through this medium. Hence, protecting this data and ensuring that the wireless channel is protected from security breaches becomes mandatory. It is at this point that the notion of cryptography is adopted. Cryptography is a branch of cyber security that tends to secure private messages sent over an online platform to protect them from breaches and attacks done by hackers. For its implementation, several encryption algorithms and techniques are used so that a respective approach can be followed and the informational exchange between the communicating parties can be secured. For this purpose, I propose the implementation of encryption algorithms to secure and monitor the online data so that a trust factor between the client and the third party is developed. The AES encryption algorithm is used to generate the existing key to be shared between the two end parties through a mail message sent by the recipient. In addition to the AES algorithm, the ECDHA encryption algorithm is also put forward. The amalgamation of the two mentioned algorithms helps to develop a technique that would imbibe the agreement protocol and execute the encryption and decryption phases without service delay. The generation of respective keys and their sizes is performed by the AES algorithm so that a communication channel can be developed between the client and the server. However, the execution of the ECDHA algorithm is used to secure the data upload and sharing on the cloud. This implementation of a hybrid concept that involves a combination of two algorithms tends to ensure that the overall security of the system model is maintained and authenticated communication is established. In the last stage, the final experiment is carried out, wherein the encryption and decryption times of various key sizes are compared for their evaluation.

1 Introduction

The theory of distributed computing has received widespread acceptance from numerous academics and software developers throughout the globe. The main justification for this is that it represents an innovation where it is anticipated that computations would eventually represent the genuine nature of classification and prediction from a sociological perspective. Despite the fact that the notion of "distributing computing" is implemented, the practice of concentrating computing power in remote data centers run by external administrations is not that reliable. The concept of distributed computing tends to deliver services from the IT sector to clients who demand cloud services with greater flexibility and adaptability because of the registration model of the system. However, the beginning of distributed computing has remained clear concerning the registration process of various models, and the notion of expansion tends to intensify it. Morgado et al. (2018)

Encryption is essential for protecting sensitive data and information kept in key management storage in a world where communication occurs over a public network like the internet. The primary goal of this information security activity is to prevent assaults from being launched by unapproved parties. Protection of digital assets through a communication channel has become necessary due to the server system's reliance on encryption. This reliance limits and eliminates a server system's overall vulnerabilities and stops new cyberattacks from happening. Therefore, creating and establishing a safe model has never been more crucial or difficult than it is currently.

Multiple site-related devices frequently experience communications that are formed between two devices and then interrupted. Any system can easily become subject to a variety of assaults that can be launched by hackers due to such disrupted connectivity Tuor et al. (2017). As soon as the device connects to the appropriate site, the server protocol is decided upon and a common encryption key is obtained, establishing a secured conversation between two people. The common key between the two gadgets is in the hands of the attacker. As a consequence, harmful assaults on the server system are more likely to succeed since the attacker is more likely to be familiar with the encryption technique being used to secure the network. Hence, it becomes vital to create authentication protocols so that hackers cannot access systems by pretending to be someone else. To prevent the attacker from discovering the encryption technique being utilized, it is also crucial to develop a robust and complex encryption algorithm. This also leads to a situation wherein it becomes difficult for the user to trust the source since the source and the server are located remotely. In such a scenario, building a trust factor becomes difficult since modifications are being made online.

Thus, a user's main worry is data security in an environment where he may utilize the resources and trust the service provider. This requires certain algorithms. Cryptography and encryption are used to provide encrypted data to users to prevent attacks and safeguard cloud data. Cryptographic approaches provide user access and safe data encryption. Cryptography is usually associated with data security and clandestine activities. Multiple study academics and software experts claim that covert communications were conveyed utilizing numerous obscure methods to securely transfer and communicate messages between two persons without a third party. Thus, this method enabled communication and cryptography. Cryptography was used to communicate financial transaction information during conflicts. Thus, information on online media was utilized to execute computer programs. Encryption begins with data access. The original data is ciphered using a secret key and decrypted with the produced key. However, the chosen method needs proper algorithms to produce a key that can only be shared by the parties concerned. The communicating parties exchange the key once it is produced, establishing the connection Ahmadian and Amirmazlaghani (2019).

Therefore, it can be believed that the algorithms so involved must be precisely selected so that the process of encrypting and decrypting the data matches the idea of cryptography. In the later stages, multiple encryption keys must be used and broken, respectively, so that cryptanalysis can be performed efficiently. In contrast, cryptanalysis is frequently associated with a study that includes secretive writing with technical analysis involved in the process in order for respective algorithms to be carried out.The implementation of cryptographic algorithms, however, depends on the usage and execution of the keys so generated. The diagram 1 depicts various forms of encryption systems used:

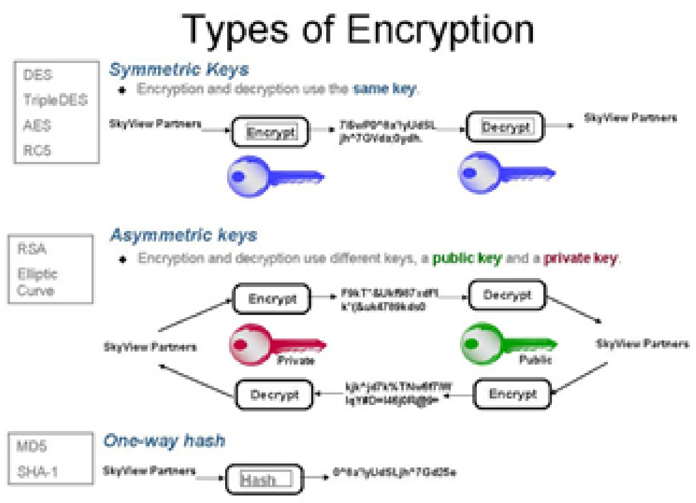


Figure 1: Types of Encryption

- Symmetric Keys: In this method, one key is used by the server and client sides of the secret key cryptography method for both encryption and decryption. The message is subsequently decoded by the recipient using the same key, restoring the message's original data. Secret key cryptography is also known as symmetric encryption due to the fact that it requires just a single key to function. Once the key is generated, it is shared between the communicating parties; further steps are taken to maintain the key's secrecy and security. The system has a number of flaws, the most prominent being the difficulty of safely storing and disseminating the produced key to all of the users. Block encryptions and stream ciphers are the categories to which this technique belongs. For stream ciphers to function, work has to be done byte by byte so that the produced key is always changing, and a feedback mechanism must be established.

- Asymmetric Keys: Symmetric encryption is deployed in a way that is significantly different from how public key cryptography is used. Financial institutions and Government organizations often use symmetric cryptography on a much larger scale. The use of symmetric keys was shown to have several key management issues. As a result, public-key cryptography was created. An "interaction" is a conversation between two people that needs to be changed from plaintext to cipher text. This conversion process is frequently referred to as the encryption stage. In the Asymmetric encryption one public key and one shared secret key is generated.

- One-Way Hash: Hash functions are one-way encryption mechanisms since they employ hash values instead of keys. Plain texts used as model input yield these hash values. Using a digital fingerprint, this approach usually reveals file contents. The digital fingerprint prevents file tampering. This method protects the file against malicious file attacks. Password-encrypted operating systems are popular with businesses. This strategy prevents file corruption. Thus, these qualities are useful for maximum security. Hashing functions are mathematical operations with fixed-size blocks and unique hash codes. Hashing algorithms typically use 256–512-bit blocks. The working implementation of a hash function 2 is shown in the example below:

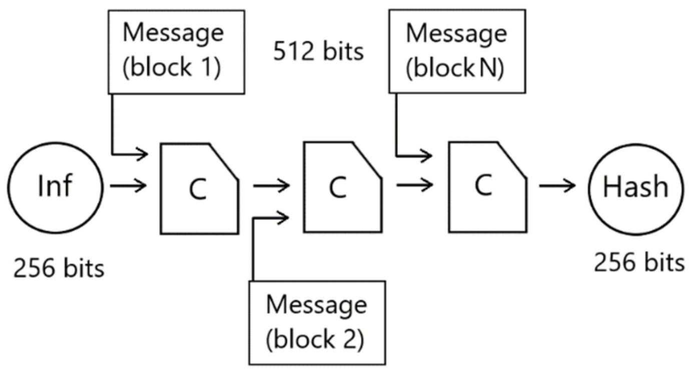


Figure 2: Visual illustration of the hash function

1.1 Motivation

There are numerous other vulnerabilities related to the cryptographic system aside from the concerns with encryption. Attacks on the server system can exploit these weak spots. These are typically referred to as "cryptanalytic attacks." Both the underlying algorithm and the general features of the plain text have a role in the success of these assaults. This simple text happens to be a common sort of contextual document that may include malicious code. As a result, before attempting to attack the system, attackers usually are aware of and study the nature of plain texts. Such cryptographic issues being taken place on server systems might result in trust loss by the users. In such a scenario, the application of the same in the healthcare domain becomes a significant challenge as this might not only result in trust loss but also data loss for patients' health records.

Therefore, protecting this information becomes a task that must be done to prevent the loss of patient files. Information about several patients may also be saved on the cloud in addition to the information about one patient. As a result of storage constraints, attacks like eavesdropping and impersonation may be used to access sensitive data from the records. Additionally, the potential of healthcare systems is sometimes restricted by the absence of interoperability between storage infrastructures and stakeholders. Hence, this consideration should also be taken care of. Finally, existing cryptographic methods and encryption algorithms are still unable to guarantee the privacy and accuracy of healthcare data while reducing its availability. Therefore, the challenges mentioned above have served the purpose of the motivation factor and have resulted in the my contribution to their research in the respective domain.

1.2 Research Question

The suggested research for the thesis is based on the idea of encrypting a system model using the principles of AES encryption and the Diffie-Hellman algorithm. This will be used in the research that will be carried out. Through the use of key generation and encryption, the fundamental purpose of this thesis is to ensure the protection of client data throughout the process of uploading and transferring such data using Microsoft Azure as the cloud model. AES is utilized as the key generation method for this purpose, and Diffie-Hellman is used to encrypt the whole process of implementation on an end-to-end basis that takes place on the cloud. Both of these processes take place on the cloud. The research question listed below will be used to validate the proposed thesis:

How data can be stored on the cloud using the secure model approach with AES and ECDH?

1.3 Structure of the report

A detailed explanation of encryption and cryptography is mentioned in Section 1, along with the motivation and the objectives of the thesis. Section 2 includes a summary of the research being performed by multiple authors who have contributed their work in the respective domain. Section 3 includes the methodologies used for the purpose of implementation along with the methodologies. Section 4provides briefs on the requirements and specifications of the system. Section 5 provides the implementation of the application and section 6 summarizes the results generated by the respective algorithms. The thesis comes to an end with the conclusion in Section 7, followed by references.

2 Related Work

Cryptography's biggest problem is securing data from attackers in a communication channel. To ensure that communications are not compromised, symmetric and asymmetric protocols can be used. Asymmetric key encryption techniques employ RSA cryptography. The symmetric key, on the other hand, is assumed to execute an asymmetric key without generating two random keys on both ends of the communication channel. Thus, the Diffie-Hellman key exchange becomes popular. These algorithms are exactly:

- Elliptic-Curve-Cryptography (ECC)
- Elliptic-Curve-Diffie-Hellman (ECDH)

AES also generates keys for communication encryption. The symmetric method is the safest way to exchange data between two parties. The key agreement protocol involves key exchange and key creation. When this agreement breaks, communication becomes insecure. All such algorithms need better security to safeguard the system from attackers. A recent study suggests RSA cryptography has integer factorization drawbacks. It also develops low exponential computational factors with increased time and space complexity. RSA may also need asymmetric padding. However, several assaults and breaches have disrupted the functional RSA implementation, and researchers have failed to protect the environment through RSA implementation. Thus, cloud-based or internet-based third parties are hard to trust. The third party must release the public key, slowing implementation. Similar to this, brute-force attacks against symmetric key encryption and cryptography are on the rise. The consequences are system failure and server vulnerability. As a result, Diffie-Hellman and execution mitigate the system's shortcomings. The Diffie-Hellman algorithm is considered to be the most recurring algorithm used in Asymmetric key exchange. The most important factor, however, is to secure the process of key generation, which must be accomplished by implementing Diffie-Hellman. Author Rao et al. (2021) tend to find that Diffie-Hellman requires a security mechanism to be activated so that the system can be protected from breaches and attacks. In other words, the implementation algorithm needs security from MITM attacks. In recent times, a survey conducted by a number of research scholars and software developers discovered that Diffie-Hellman, when combined with the implementation of the AES algorithm, is capable of dealing with attacks such as the MITM attack.

2.1 RSA Cryptography

One of the algorithms used in asymmetric cryptography is called RSA cryptography, which processes data by using public and private keys. In the execution process, the receiver generates a public key and retains its private key. Later, to fulfill the purpose of communication, anyone can try to send a message to the other end of the communicating channel by encrypting the generated public key. In this process, only the receiver can decrypt the message using the key that has already been generated. Meanwhile, the public key generated is published and labeled as the key that can be used for the long term. However, in the process, the attacker cannot breach the established system since the key generated is only known to the parties involved in the process of communication.

Author Aggrawal et al. (2018) contributed their work to the research study of implementing RSA and verifying the security of data files stored within the storage network. This storage network consisted of all the files that were associated with the service adopted by the user. The author proposed to secure this data on the storage network by encrypting the messages that were to be transferred through the communicating parties. In this scenario, the RSA algorithm would encrypt and decrypt the data on both sides, maintaining the integrity of the system. The author also proposed that the implementation of the RSA algorithm be further combined with the implementation of digital signatures that verified the process of encryption on both sides.

In a similar work by authors Kleinjung et al. (2010) proposed to generate public keys on both ends, for the client as well as the server. The generation of keys was done using the RSA algorithm so that the keys could give access to the respective user and prevent the system from further attacks. However, the legitimate user's access was gained by using the respective file mentioned in the cloud's data storage. The implementation was further enhanced using an encryption algorithm such as AES. This enabled a secured system to be followed so that the server could be protected from breaches. The overall space complexity obtained by the system was observed to be 49 ms.

In another work wherein the authors Wahab et al. (2021) proposed the implementation of symmetric and asymmetric algorithms so that security breaches could be avoided. His work contributed to the research that focused on the communication that would take place between a client and the server that is deployed in the cloud. The implementation was further combined with the working algorithm of RSA, and the entire system model was executed under the activities of encryption that included the time and space complexity of the system. The proposed model leads to an increase in the security and integrity of the system.

In another survey conducted by authors Gidney and Ekerå (2021) involved cloud deployment using distributed computing. This majorly involved securing the data before it was uploaded so that the overall security of the system could be maintained and privacy couldn't be breached. The authors' technical implementation includes an analysis of securing the privacy of user data throughout the stages of the data life cycle. The same work further included multiple techniques and schemes to secure user data and further protect it from getting hacked on an unsafe medium. However, the proposed implementation by the author had the disadvantage of not being able to store large amounts of data. As a result, the generation of keys for respective algorithms was reduced, allowing the overall data to be managed and communication channels to be established.

2.2 AES Cryptography

A commonly used algorithm for symmetric cryptography is implemented through the Advanced Encryption Standard (AES). The implementation primarily includes the derivation of cipher keys through round keys. The scheduled method of AES also includes an XOR operation that is further combined with a single-byte key of the round key and is generally referred to as the AddRoundKey. The implementation of the round key includes replacing every byte of the round key with a look-up table that could be executed as a substitutional step. This replacement is referred to as the "subbytes." This replacement also includes cyclically shifting the rows attached in the number of states, with certain steps involved in the transposition step. This shifting of rows is termed "shift rows." On the other hand, when such shifting occurs through the columns, it is referred to as "MixColumns."

The authors of the studyZaw et al. (2019), used symmetric encryption techniques that included linear geometry concepts. The entire process of communication was done to maintain the secrecy of the server system so that covert communications could take place over online platforms. For this purpose, both the techniques of substitution and transposition were used, and communication was established so that the channel could not be breached. However, the primary idea behind this was to protect the user's data by using a random number generator so that a matrix could be created and the number of bytes involved in the stream could be calculated

On the other hand, the implementation of asymmetric encryption was done based on a cyclic elliptic curve wherein the author Al-Haj and Aziz (2019) contributed their work in order to transfer data across multiple channels of the neural network in order to establish an authenticated and secure form of method. The model so far contributed by the author executed a non-linear table that consisted of a 32-bit register that was responsible for operating effectively. However, the proposed work could not be used for the implementation of small areas.

Authors Kartit et al. (2015) suggested using an encryption scheme based on AES and RSA to secure the data being saved on cloud services, which allowed them to evade the issue with the model above. AES was used to secure the data encryption process, and keys with key sizes of 128, 196, and 256 bits were used. In circumstances where a key size of 1024 bits was required, the RSA method was also implemented in addition to the AES algorithm. Due to the employment of two encryption methods and the provision of an additional degree of security to store the encrypted keys, the use of HCT improved the overall security of the system model.

The author's of Standard (2001) study proposed the implementation of a similar technique wherein they conducted experiments to control and monitor data that was supposed to be used online. The research work included the implementation of symmetric and asymmetric encryption to protect the data online. This was further followed by encryption algorithms that also included a comparison of multiple research works being conducted in the same domain. The data was further protected using AES encryption, wherein the generated key was to be accessed only by the respective people.

2.3 Diffie-Hellman

The Diffie-Hellman method is widely considered to be one of the most secure algorithms; however, it requires high levels of computational performance to ensure the smooth exchange of data between communicating parties. By transferring a common value of the key that is generated using the principles of data encryption, two computers can communicate with one another. Additionally, the hashing method is used to safeguard the traded data. Practically, keys are never exchanged between the two parties involved; instead, the same key is created on both ends using the keying material's 468 and 1024 bits.

In research by Kumar et al. (2017), the authors recommended the adoption of cryptographic techniques that combined Diffie-Hellman with elliptical curve cryptography. The combination of these two distinct algorithms produced a far better link for establishing the necessary data security over a risky network. However, this approach took advantage of all the security risks related to maintaining the data's integrity via the cloud. The authors' later proposals for a new architecture were successful in achieving data security because they implemented it in four steps, namely:

- Setting up the necessary connection for two communicating parties.
- The data from the recorded photos would subsequently be concealed via steganography.
- Verification of legitimate users.
- Data exchange in a cloud environment.

In a different research work being proposed by authors Al-Mahmud and Morogan (2012), they put forward the implementation of digital signatures to give legitimate authentication to users so that the message sent could be validated by the respective recipient. The execution of the proposed digital signature was based on ECC, wherein the users had a specific time slot for activation to register themselves. The group identities of the users were collected, and services were given to them based on their requested access. Breach types such as DoS attacks were avoided by implementing DH-based ECC. In addition to this, the computational overhead was reduced to a greater extent.

The authors Kavitha et al. (2019) observed a similar application of digital signatures, in which a healthcare application would be authenticated using ECC concepts. Encryption algorithms such as AES and RSA had failed to provide the level of security needed for large volumes of data; hence, the author came up with the ECC implementation, derived from the theory of Diffie-Hellman. ECC was used to generate secret and public keys with a bit size of 256, whereas a hybrid technique was adopted for further securing the system. The hybrid technique, however, included a combination of digital signatures and the Elgamal algorithm and resulted in an accuracy of 91.52 percent.

Real-time research was proposed by authors El Zouka and Hosni (2021), wherein they designed a computationally lightweight security mechanism that could secure and monitor patient record files in real time. The framework included immediate consultation with the doctor, and an approach called machine-to-machine (M2M) was adopted that could enable patient monitoring on the web app. The web app was remotely located, which reduced the overall time for service by quickly verifying the keys.The real-time application of the same was implemented using concepts of Diffie-Hellman and resulted in enhancing the security of the overall system.

The proposed strategy by Abusukhon et al. (2019) did fall short, nonetheless, in terms of protecting cloud data. A DNA-based method was created for this purpose to resolve its associated problems for securely safeguarding the data in the cloud environment. Binary coding and DNA steganography concepts were applied by the authors. The cloud environment now has an additional degree of protection thanks to this binary coding. However, cryptography itself served as the foundation for how steganography was carried out.

3 Methodology

The literature survey that was undertaken revealed that one of the biggest obstacles that has to be overcome is enhancing the system's overall security. To improve the system's overall effectiveness, I propose using block cipher encryption techniques in this thesis. This section of the thesis provides a synopsis of the methodologies used to implement the proposed thesis.

3.1 Taxonomy of Elliptic Curve of Diffie-Hellman

The elliptic curve Diffie-Hellman algorithm, often known as ECDH, is a Diffie-Hellman algorithm that uses the straightforward characteristics of a Diffie-Hellman to simplify the complicated process of data security. When elliptical curves are implemented using mathematical calculations, smooth projective curves with a defined point O are more likely to be produced. It is proven that each curve represents a section of the field that has a common feature and is represented by a planar algebraic curve. Following is the equation for this curve on the algebraic plane:

$$y^2 = x^3 + ax + b$$

Elliptic curve cryptography is gaining traction as a promising public key cryptosystem with potential for use in a mobile environment. When compared to previous systems, this technique offers significantly greater data security and has relatively small key sizes. Fast computational power with minimal bandwidth usage typically follows the operation of ECDHA. As a result, all these characteristics make it simple to install on wireless networks. However, the Diffie-Hellman theory prioritizes using this method even over a dangerous channel. The reason the involved parties can do this is that the generated key is only accessible to and shared with legitimate users. This minimizes the possibility of the produced key being improperly handled by limiting its distribution to the legitimate parties involved. Later phases involve using the secret key that was created for communication purposes that utilize symmetrical encryption techniques.

However, there is one distinction between the two approaches: the algorithm additionally uses algebraic curves once the keys are generated and distributed to the two communication parties. The primary distinction between the two algorithms at work is formed by these algebraic curves. During the key generation phase of the algorithm, the communicating parties must also agree on a desirable elliptical curve. It is important to highlight that when compared to the standard DH, elliptical curve implementations generally operate much more quickly. The AES method works by the sender generating a random key and sending it via a secure channel to the receiving server. However, it should be emphasized that the key that was generated and supplied is simply a plaintext key and was not initially encrypted. If an attempt is made to encrypt this key using a certain algorithm, the encrypted key would then need to be decoded. The need for a second key would then arise in the subsequent stage of decrypting the key on the other end, and an identical problem might occur.

As a result, it is advised to consider creating keys on both ends of the communication channel rather than exchanging them later and encrypting them further using an algorithm. This way, the sharing of these keys can be avoided as attacks are more likely to occur during the key-sharing process. Therefore, the program often use the elliptic curve and Diffie-Hellman techniques to generate a 16-byte key for the AES algorithm. The production of the key process is discussed in the thesis implementation section.

Overall, the use and application of ECDHA have speed up the implementation process significantly. This technique is known to improve the system's overall security and make it operate much more effectively. Thus, this approach is thought to function more effectively than other cryptographic ones. The security of a typical ECDHA implementation is 164 bits, while the security of a similar implementation using an RSA-based method is 1024 bits. However, both approaches typically deliver the same level of security. In later stages, the ECDHA idea offers a wide range of cloud computing solutions, such as the use of less computing power and less battery storage.

3.2 Taxonomy of Cloud

Due to the ideas of cloud computing, a completely new paradigm has been established with the storage and saving of enormous amounts of user information on the cloud. Storing this data on the cloud enables the service provider to securely monitor and maintain the sensitive data of the user by dedicating a private connection on the respective network. This individual storage ensures that data can be accessed indefinitely and virtually scaled. In addition to this, the ability for data to be stored in the cloud eliminates the need for the user to buy a respective amount of network bandwidth to manage his own data infrastructure and thereby allows him to access his data from anywhere in a scalable manner.

Cloud service providers offer a variety of services, including Software as a Service (SaaS) and Platform as a Service (PaaS). Since a cloud's functionality has been expanded to include scalability and efficiency, there are three types of cloud implementations: public, private, and hybrid. A public cloud's fundamental functions are carried out by a third party, as opposed to private clouds, which are controlled by businesses and individuals. On the other hand, a hybrid cloud's networking may be both partially private and partially public. So, employing the cloud as a service has a significant economic and manpower investment benefit. Users using cloud computing do not need to install any special infrastructure, in contrast to traditional computing. As a result, all ongoing costs are reduced by eliminating general technological maintenance. In addition to its benefits, a cloud typically has resource sharing and on-demand self-service as its key features. When a user needs to control his computing resources or request a service, on-demand self-service is primarily claimed. On the other hand, resource pooling refers to the collection of computer resources from distant locations into a single data center.

Although cloud computing as a service has become more mature over time, users have yet to adopt the concept widely because of serious worries about data loss. The foundations of encryption are employed to resolve and overcome this developing problem. Before the essential operation is carried out, this encryption procedure is further integrated with the creation of public and private keys that are shared between the user and the cloud provider. To successfully secure user data in the cloud, I recommend using Microsoft Azure web services. Microsoft Azure is a cloud-based flexible architecture that can provide users with the services they require. It can be considered a hosting service that assists the data center to monitor and store large chunks of recurring data. The proposed cloud architecture is considered secure and thereby designed to provide services through APIs.

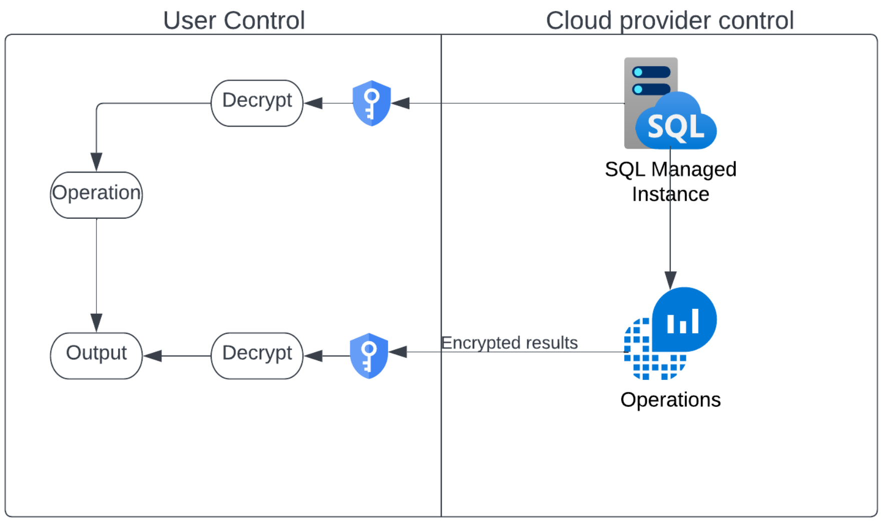


Figure 3: Encryption in Cloud

3.3 Proposed Methodology

I suggest how the model should operate in this area of the thesis that has been presented. It is necessary to assess the cryptographic algorithm's overall evaluation model in terms of resource complexity as well. Consequently, setting its evaluation settings is a necessary function. The assessing model in the suggested thesis is based on the ideas of the AES and ECDHA algorithms. The model is assessed subsequently based on the execution's time and phase complexity. The transformation of plaintext into cipher text starts the model's execution. The assistance of a random number generator is used in conjunction with this conversion. The assessed plaintext and its size are created randomly by keys in the thesis as it has been implemented. The bit size of the cipher text also affects how big the keys are. The AES method is used to parse all of the keys in the following step. It is important to note that the more keys used, the more secure the data is when it is stored on server sites. The process of the suggested system is shown overall in the diagram below 4:

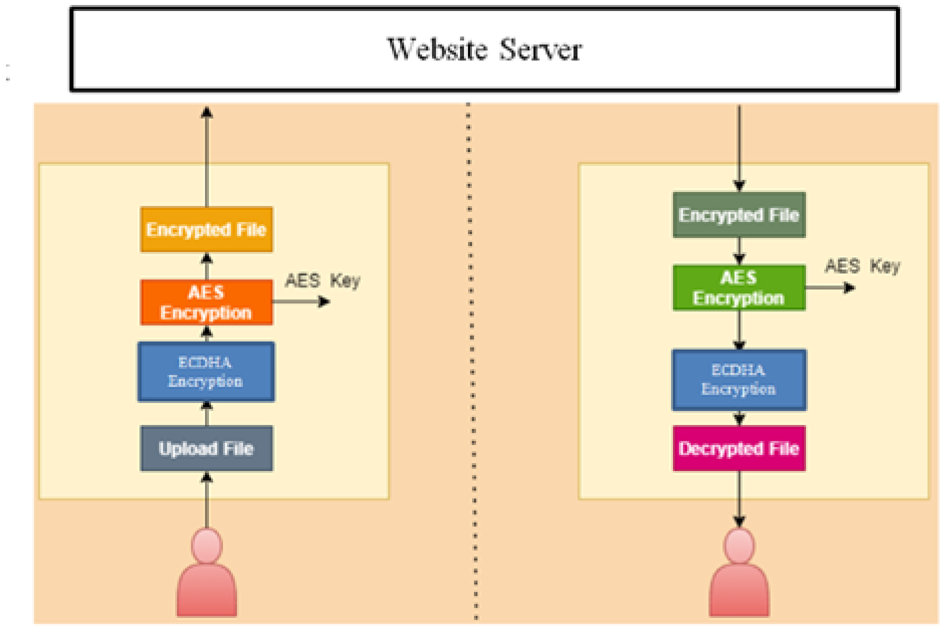


Figure 4: Workflow of the architecture

The figure 4 above shows the systems architecture entire workflow. Three different data security methods are suggested for use in the implementation of the thesis.

- The creation of keys is the first one. This is the key that the two users of the communication channel share, which is present on both sides and can be utilized by both ends in the future. However, the elliptic curve Diffie-Hellman technique is used to generate this key. The keys are sent to the appropriate receiver/end of the communication channel once they have been produced.
- In the second step, which entails authenticating the shared keys on both sides of the communication channel. User authentication credentials (username and password) are sent to the user after verification. By connecting to his email, obtaining his login ID and password, and then entering the site server with those credentials, the user can access these common credentials.
- The last step of the encryption process involves the user decrypting the message using the AES algorithm on their end.

The proposed approach is referred to as the "hybrid-based encryption model" because the server model's execution entails two combined AES and ECDHA algorithms. With the help of this approach, both parties to the information exchange can safely upload and download data. This hybrid model also improves security because it now incorporates the ideas of two algorithms to accomplish the same goal of protecting data being kept and moved across a risky platform. The AES algorithm on the receiving end decrypts the data, and re-encryption is then carried out in the other direction. In order to view the document that has been uploaded or downloaded, both the user and the administrator will need to provide their login credentials onto the server website where this implementation is taking place. The entire procedure contributes to the network's communicating endpoints developing a general sense of trust. However, only one server is kept to maintain the system's confidentiality and integrity, as well as the trust element. However, this single server can be accessed using both the user's and the administrator's login information. The transfer and sharing of data between two parties also take place on this server, and the databases on these servers themselves securely store the shared data. Both individuals can access the website server using the appropriate credentials that were sent to them via their email id.

Therefore, in this situation, the user must first utilize the ECDHA algorithm to upload a file. Key generation for the user's account occurs inside the ECDHA algorithm, allowing him to access his account with ease. Added verification of the user's identity is performed. Since the authorized user was notified of his authentication by email, it is now very difficult for a hacker to get access to the system and take control of the data files. As a result, the entire procedure is thought to be secure from hacker attacks. The data files are then encrypted on the website server using a hybrid encryption method that combines the two ideas from before. The data files are uploaded to the server site after this hybrid encryption algorithm has been used.

The receiver can now download the data files using the credentials that were previously provided once they have been saved in the database. Now that he has logged in, the user downloads all the required data files onto his computer from the server environment. In this way, the system's integrity and confidentiality are kept intact, and the security of the system as a whole is well preserved. The presence of assaults in this thesis implementation is one of the most crucial things to observe. The server model is subject to vulnerabilities and may experience specific assaults that could be initiated by the attackers because key sharing occurs on an insecure platform. On the other hand, unauthorized access to the generated keys could be obtained by intruders using a man-in-the-middle attack or even an eavesdropping assault. Therefore, it is crucial to prioritize sending the keys to the authorized user when distributing these keys. Only after sending the legitimate user's login information can this assurance be made. Diagram 5 makes clear representation of the application environment.

4 Design Specification

The design specifications of overall system will be coded on ASP.NET framework using C# language and then the whole system will be deployed on the Azure platform with the database also deployed in the Azure SQL server. AES algorithm begins with the process of key generation, which occurs by generating random numbers on the sender's side. The generated keys are then transmitted so that communication can take place through the channel and the receiver can receive the respective sent message. Once the generated key for AES encryption is sent to the receiver, plain text is formed that is

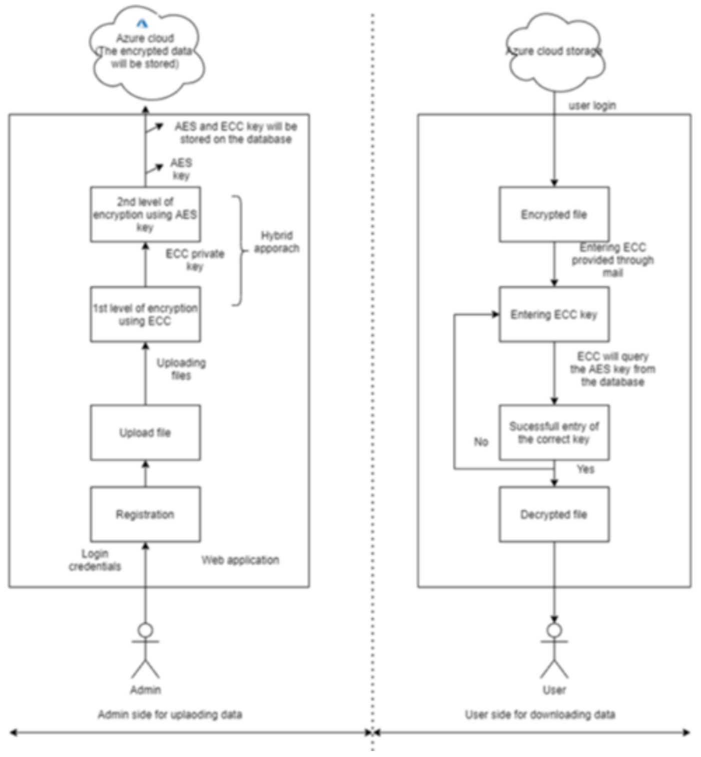

Azure cloud (The encrypted data will be stored)
AES and ECC key will be stored on the database
AES key
2nd level of encryption using AES key
Hybrid apporach
ECC private key
1st level of encryption using ECC
Uploading files
Upload file
Registration
Login credentials
Web application
Admin
Admin side for uplaoding data
Azure cloud storage
user login
Encrypted file
Entering ECC provided through mail
Entering ECC key
ECC will query the AES key from the database
Sucessfull entry of the correct key
No
Yes
Decrypted file
User
User side for downloading data

Figure 5: Architecture of the application

in a readable format that the reader can comprehend. In the next stage, the process of authentication takes place so that the respective message can be decrypted. However, the communication between the two parties occurs through the transfer of keys generated by the actions of the users involved in the communication. The recipient uses the respective login information and credentials to access the mail sent and, therefore, receives the generated key in the mail. Once he receives the generated key, he decrypts them to complete the entire process of communication. The overall security of the system is maintained in this manner, and the transfer of communication tends to take place on the cloud. The web server is deployed in this cloud, and the security of providing respective services is monitored in Microsoft Azure. The overall process, however, requires two login credentials, one for the administrator and one for the user, which eventually could be accessed through their respective dashboards. However, it can be considered that the primary objective of the thesis is to implement the techniques of the ECDH method along with the AES algorithm and secure the entire process of communication that keeps being placed on the cloud.

The model must be able to handle encryption assaults and breaches. Data security is key to successful internet communication because it should all happen online. Thus, the thesis's main goal is to secure cloud communication by generating keys utilizing AES as the encryption algorithm and ECDH to secure the system model. Diffie-Hellman and elliptic curve algorithms enable this strategy. This key was created briefly in the thesis implementation. The thesis hybrid model is created using the ECDH method, simplifying the complicated process. To secure encryption and decryption, both ends of the communication chain employ Diffie-Hellman. Thus, the thesis uses AES and ECDH to secure data files uploaded and shared on the web server. As seen in the 5, the encryption will transit to the cloud, be decrypted by the user, and be downloaded.

5 Implementation

In the following paragraphs, I will talk about the implementation of a hybrid model that uses a symmetric method. Elliptic curve Diffie-Hellman, more commonly known as ECDH, is the complementary algorithm to AES. In order to strengthen the security of client-server communication, AES is combined with ECC. After that, ECDH is used to improve key generation and further strengthen security. The ECDH algorithm will generate the AES key, which the AES algorithm will then use to encrypt and decrypt data. In addition, there will be a key agreement for each session, which will create a common understanding between the client and the server for the duration of that particular session. The client was only able to decode the message after successfully agreeing to the key.

Step 1: The experiment is conducted on several types of files. These files serve as inputs for advanced encryption and decryption inside the system.

Step 2: Elliptic curves will be used to generate keys, Eliptic curves provide two sets of key pairs, one for private use and one for public consumption. Let's say 'g' stands for the server's secret key and 'f' stands for the client's.

Step 3: The "g" character serves as the server-secret key, which AES will use to encrypt the input file. AES will employ the same rounds for permutation and substitution for this hybrid approach; however, the key generation technique for the AES-ECDH model will be different. This means that AES will use an ECDH key for encryption and decryption.

Step 4: ECDH will start the key agreement at the same time by employing the secret keys of both the client and the server; it will generate the shared secret for that particular session. The following are the phases that must be completed for a successful key agreement:

- Both parties need domain parameters (p, a, b, n, G, h). Elliptic curves are defined over a field represented by 'p', where 'a' and 'b' are constants whose changing values generate a variable number of curves. The generator point (G) is a fixed location recognized by both parties. The curve's points are evenly distributed for h=1, and 'G''s prime order is 'n'.

- The server will have 'g' and the client 'f'. Using secret keys, the server and client will generate a public key. They both know generator point "G" initially.

- The client's public key is P(c) = fG. The server's public key, P(s), is generated by the generator point 'G'.

- To create a shared secret, they must exchange public keys. The server sends P(s) to the client, while the client sends P(c).

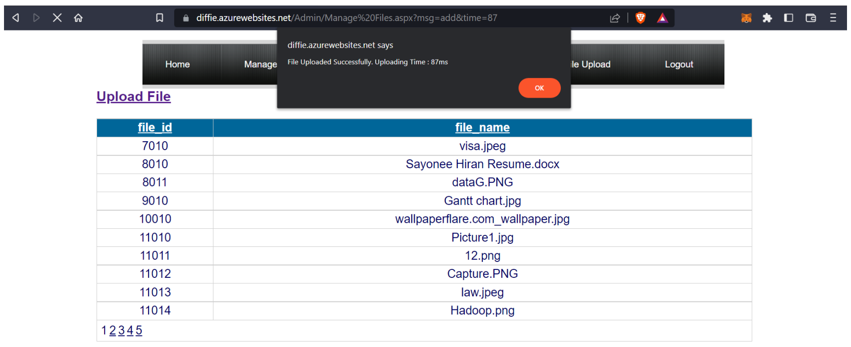
- Server will compute S = gP(c) and client will calculate S' = fP(s) using their shared public keys. They'll compare calculated S and S' values.

- S' = fgG since P(s) = Gg and S = gfG because P(c) = fG. The values of S and S are shared secret values.

- AES-ECDH Encryption and Decryption Model Client-server key agreement will succeed after the development of a shared secret.

Step 5: Client and server now share a secret value. After receiving an encrypted file from the server, the client will decrypt the file using the combined ECC and Diffie-Hellman keys. Sharma and Pokharana (2021)

As seen in Figure 6, after a file has been uploaded to the server, the encryption time is shown on the site. This is the total time, in milliseconds, required by both AES and ECDH to encrypt a file. Therefore, the initial encryption occurs in the ECDH using a shared key produced by the elliptic curve. In comparison to other methods, elliptic curves use less computational resources while maintaining the same degree of security. The shared secret key created by the ECDH algorithm is 256 bits long and offers the same degree of security as the 3027-bit shared secret key obtained by the classic Diffie-Hellman algorithm. The AES algorithm employs a key produced using the ECDH method to encrypt and decrypt data. Alice and Bob will produce their respective keys using the secp256k1 algorithm for the ECDH. Using the algorithm "secp256k1," both sides will produce the same shared secret key on the curves. Therefore, Alice will produce her shared secret key using Bob's public key as an input. Bob will produce the shared secret key using Alice's public key after the key has been generated. Both of the generated keys are identical. Now, when an administrator uploads a file, the file is encrypted using AES. The AES algorithm employs the SHA256 hashing algorithm. Secure hash algorithm 256-bit (SHA-256) is a cryptographic security algorithm. Cryptographic hash techniques generate hashes that are irreversible and unique. This hash has a hexadecimal value of 64 bits. Each character of the hexadecimal value consists of 4 bits; therefore, 64 * 4 = 256.



diffie.azurewebsites.net/Admin/Manage%20Files.aspx?msg=add&time=87
Home
Manage
ile Upload
Logout
OK
Upload File

diffie.azurewebsites.net says

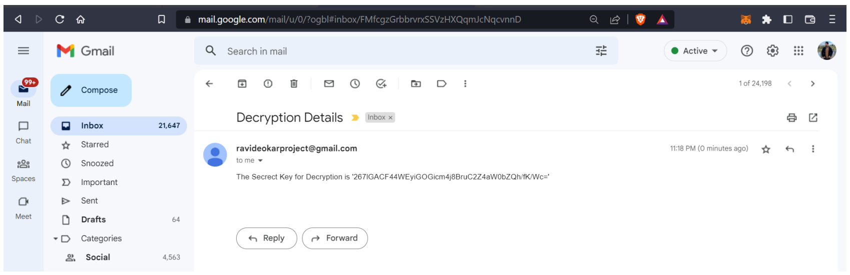
File Uploaded Successfully. Uploading Time : 87ms

file_id	file_name
7010	visa.jpeg
8010	Sayonee Hiran Resume.docx
8011	dataG.PNG
9010	Gantt chart.jpg
10010	wallpaperflare.com_wallpaper.jpg
11010	Picture1.jpg
11011	12.png
11012	Capture.PNG
11013	law.jpeg
11014	Hadoop.png

1 2 3 4 5

Figure 6: Encryption of the file on the Admin side portal

Image 7 represents the ECC key that is sent over the mail to the client for the encrypted file. The administrator can select the user with whom he wants to share the file and send the secret key over email.



mail.google.com/mail/u/0/?ogbl#inbox/FMfcgzGrbbrvrxSSVzHXQqmJcNqcvnnD

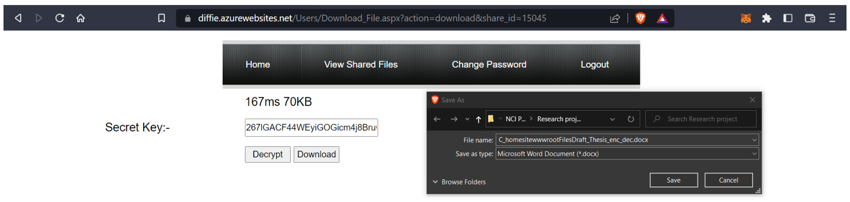
Decryption Details

ravideokarproject@gmail.com
to me
11:18 PM (0 minutes ago)

The Secrect Key for Decryption is '267IGACF44WEyiGOGicm4j8BruC2Z4aW0bZQh/fK/Wc='

Figure 7: Sharing of the secret key via mail

In image 8 we can see in the user portal the shared file is been decrypted and the time and the altered file size is been displayed. This whole process is running on the Azure web services.



diffie.azurewebsites.net/Users/Download_File.aspx?action=download&share_id=15045

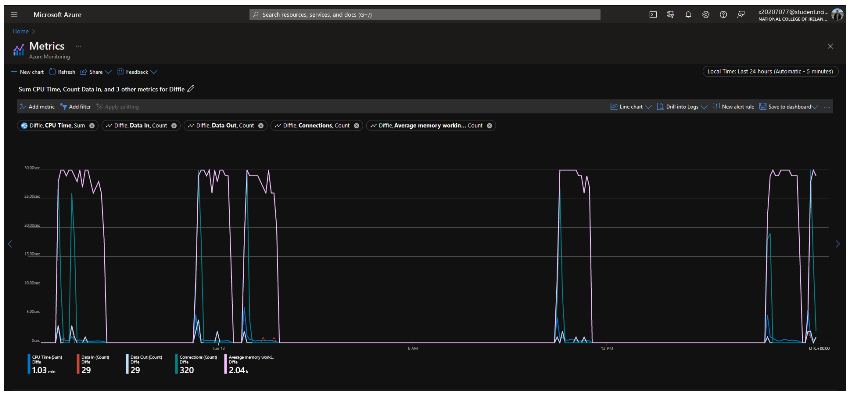
167ms 70KB

Secret Key:- 267IGACF44WEyiGOGicm4j8Bru

File name: C_homesitewwwrootFilesDraft_Thesis_enc_dec.docx
Save as type: Microsoft Word Document (*.docx)

Figure 8: Decryption of the file on the user portal

After implementing the program successfully on localhost, I pushed it to the Azure web services. The server which is utilized for the deployment comprises of shared infrastructure with 1GB of RAM and 240 minutes of computation in a day. As shown in the figure, four application insights are presented. The chart displays parameters such as CPU utilization, data in, data out, connection count, and average memory use. As we can see in the fig 9



Sum CPU Time, Count Data In, and 3 other metrics for Diffie

Local Time: Last 24 hours (Automatic - 5 minutes)

CPU Time (Sum) Diffie	Data In (Count) Diffie	Data Out (Count) Diffie	Connections (Count) Diffie	Average memory worki... Diffie
1.03 min	29	29	320	2.04k

Figure 9: Azure App Insights

6 Evaluation

As observed in the literature survey, the implementation of the RSA algorithm tends to generate certain drawbacks in a model so developed using the same concepts. One of the significant challenges of an RSA algorithm is that its execution involves low exponential computational factors. Due to this drawback, the overall time and space required to store the data become complex in nature. In addition to this RSA algorithm also undergoes the problem of integer factorization that might in turn obscure the security of the system model.

Hence to overcome such issues and challenges; the proposed thesis tends to imply the conceptual theory of AES and ECDHA algorithm. A combination of the above-mentioned encryption algorithm tends to secure the system from attacks and breaches and assist to establish communication effectively. However, multiple parameters are used to assess this approach's performance. To examine the suggested approach, two parameters—namely, encryption time and decryption time—were combined. Multiple file sizes were used as input in the subsequent stages, and this strategy performed better overall in terms of encryption time.

- Encryption time is the length of time it takes to encrypt data before it is submitted by the website administrator to the server.
- On the other side, the time required to download the data from the website server and then decrypt it using the private key is used to compute the decryption time.

Our application is deployed on the Azure web services app deployment service using a shared hosting plan with 1GB of RAM and 1GB of storage space on a Windows server. As this scenario is for academic purposes only, the deployment plan is set to shared to keep costs low; nevertheless, this is not suggested for deploying this application for public usage. In this approach, we often upload many files of various sizes to carry out the thesis and then verify how long it takes to encrypt and decrypt those data. The time it takes to encrypt and decrypt data using the suggested method is shown in the table below, and the storage complexity is further expressed in KB:

6.1 Experiment / Case Study 1

In the first experiment, I will encrypt the data by utilizing image files as the input and record the amount of time it takes to encrypt and decode the data coming from the website server. In addition, I will record the amount of time it takes for the data to be decrypted. I am going to capture images with a variety of file sizes so that I can record the time as precisely as possible. The photographs will be recorded in jpg format, and their sizes will range from very small to very large. This will ensure that a record can be kept of the times at which encryption and decryption occurred. The amount of time needed to encrypt data could shift based on how fast the connection is.

6.2 Experiment / Case Study 2

In the second experiment, I will encrypt the data by utilizing CSV files as input and will record the amount of time necessary to encrypt and decode the data once it has been received from the website server. In addition to that, I am going to keep a record of

Size (Kb)	Encryption time(Ms)	Decryption time(Ms)
137Kb	72	366
172Kb	88	616
259Kb	98	913
605Kb	227	6021
1048Kb	341	15995

Table 1: Time taken for images to the process of encryption and decryption

the amount of time necessary to decode the data. To record the time as accurately as is practically possible, we are going to capture it using a variety of different file sizes. Excel files saved in CSV format may range in size from very small to very large. This will ensure that a record of the times at which the encryption and decryption processes took place can be kept. There may be a correlation between the speed of the connection and the amount of time needed to encrypt data.

Size (Kb)	Encryption time(Ms)	Decryption time(Ms)
123Kb	70	295
366Kb	95	1846
654Kb	150	8448
826Kb	156	12902
1209Kb	211	18722

Table 2: Time taken for to the CSV files process of encryption and decryption

6.3 Experiment / Case Study 3

In the third experiment, I am going to take pdf files as input data and then test how the algorithms handle the data. In addition to that, I am going to keep a record of the amount of time necessary to decode the data. To record the time as accurately as is practically possible, we are going to capture it using a variety of different file sizes. PDF files may range in size from very small to very large. This will ensure that a record of the times at which the encryption and decryption processes took place can be kept. There may be a correlation between the speed of the connection and the amount of time needed to encrypt data.

Size (Kb)	Encryption time(Ms)	Decryption time(Ms)
79Kb	70	149
295Kb	140	1706
878Kb	326	12826
1177Kb	329	17993
1232Kb	368	28525

Table 3: Time taken for to the PDF files process of encryption and decryption

6.4 Application URL

`https://diffie.azurewebsites.net/`

6.5 Discussion

When we consider the facts, we are able to reach the conclusion that the amount of time needed for encryption is much smaller than the amount of time needed for decryption. It has been shown that the amount of time needed to encrypt a file increases proportionally with the file's size during all three of the tests that were carried out. Because the data is first encrypted using the AES technique, and then once it reaches its destination in an unreadable state, it is both decrypted and re-encrypted to ensure that it remains unreadable. The Advanced Encryption Standard, often known as AES, is an example of a symmetric algorithm. This means that it encrypts and decrypts data at the same rate. In certain streaming modes, all that the Advanced Encryption Standard (AES) does is output a stream of bits, which are subsequently xored along with the information that has to be encrypted. In order to decode the information, the receiver must first run the same version of AES that was used to produce the bitstream, which it then xors into the information. The results are reliable due to the fact that they take into consideration the whole amount of time that was spent on both approaches.

7 Conclusion and Future Work

A high scalability and a reduced amount of downtime are the two most important aspects that contribute to the model's improvement. When data is stored on and transferred via a website server while simultaneously using a communication mechanism such as the exchange of keys, data security concerns are certain to occur. In addition, given that this data can be accessed over the internet, it is possible to access and store it regardless of one's physical location in the globe. As a result, the protection of sensitive data is very necessary. In addition, encryption is used in order to prevent unauthorized access to the data. In order to put into practice the model that is discussed in the thesis, the AES algorithm and the ECDH principles have been merged. When compared to the use of only one encryption technique, this combination often results in the data being in a safer state. Due to the fact that it follows the symmetric key algorithm concept, AES encrypts and decrypts using the same key. The fact that the key was used in this manner on both ends requires that it be protected as sensitive information and concealed. When this secret key is utilized to transit between the sender and the recipient, the exchange is then carried out across a communication channel. This mode of communication often lacks the necessary level of safety. The use of the approach that was described, which creates the key using the ECDH algorithm, is thus recommended. Because it has a low computational cost and inherits the properties of the Diffie-Hellman system, the algorithmic model is given a more distinguishable quality. The primary objective of the thesis that has been proposed is to find a solution to the issues that crop up with symmetric algorithms whenever the communication path is not secure. By using AES and ECDH-based algorithms, the necessity for a secured connection may be avoided, which eliminates a potential barrier. It is expected that the cryptographic paradigm that has been proposed would work in any implementation situation, including cloud computing and the Internet of Things. When the Diffie-Hellman algorithm is combined with the

ECDH algorithm, the manufacturing of secret keys is simplified and made more secure. This is due to the elliptic curve features of the ECDH method, which are the algorithm's best feature. However, due to the duration of this key generation and the impact bit size, this procedure uses very little of its available resources. The completion of future tasks related to this project may involve the addition of geographical areas for the purpose of measuring the amount of time required to encrypt and decode files of varying sizes. In the event that there is an emergency at the data center, the model may be constructed in such a way that it is possible to recover the encrypted data from any place. Later rounds of the operation may accommodate the addition of large data files, which can then be stored inside the same location.

References

Abusukhon, A., Anwar, M. N., Mohammad, Z. and Alghannam, B. (2019). A hybrid network security algorithm based on diffie hellman and text-to-image encryption algorithm, *Journal of Discrete Mathematical Sciences and Cryptography* **22**(1): 65–81.

Aggrawal, M., Kumar, N. and Kumar, R. (2018). Optimized cost model with optimal disk usage for cloud, *Big data analytics*, Springer, pp. 481–485.

Ahmadian, A. M. and Amirmazlaghani, M. (2019). A novel secret image sharing with steganography scheme utilizing optimal asymmetric encryption padding and information dispersal algorithms, *Signal Processing: Image Communication* **74**: 78–88.

Al-Haj, A. and Aziz, B. (2019). Enforcing multilevel security policies in database-defined networks using row-level security, *2019 International Conference on Networked Systems (NetSys)*, IEEE, pp. 1–6.

Al-Mahmud, A. and Morogan, M. C. (2012). Identity-based authentication and access control in wireless sensor networks, *International Journal of Computer Applications* **41**(13).

El Zouka, H. A. and Hosni, M. M. (2021). Secure iot communications for smart healthcare monitoring system, *Internet of Things* **13**: 100036.

Gidney, C. and Ekerå, M. (2021). How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits, *Quantum* **5**: 433.

Kartit, Z., Azougaghe, A., Kamal Idrissi, H., Marraki, M. E., Hedabou, M., Belkasmi, M. and Kartit, A. (2015). Applying encryption algorithm for data security in cloud storage, *International Symposium on Ubiquitous Networking*, Springer, pp. 141–154.

Kavitha, S., Alphonse, P. and Reddy, Y. V. (2019). An improved authentication and security on efficient generalized group key agreement using hyper elliptic curve based public key cryptography for iot health care system, *Journal of medical systems* **43**(8): 1–6.

Kleinjung, T., Aoki, K., Franke, J., Lenstra, A. K., Thomé, E., Bos, J. W., Gaudry, P., Kruppa, A., Montgomery, P. L., Osvik, D. A. et al. (2010). Factorization of a 768-bit rsa modulus, *Annual Cryptology Conference*, Springer, pp. 333–350.

Kumar, C., Vincent, P. D. R. et al. (2017). Enhanced diffie-hellman algorithm for reliable key exchange, *IOP conference series: materials science and engineering*, Vol. 263, IOP Publishing, p. 042015.

Morgado, C., Baioco, G. B., Basso, T. and Moraes, R. (2018). A security model for access control in graph-oriented databases, *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, pp. 135–142.

Rao, M. S., Rao, K. V. and Prasad, M. K. (2021). Hybrid security approach for database security using diffusion based cryptography and diffie-hellman key exchange algorithm, *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, IEEE, pp. 1608–1612.

Sharma, S. and Pokharana, A. (2021). Comparative analysis of aes-ecc and aes-ecdh hybrid models for a client-server system, *2021 2nd Global Conference for Advancement in Technology (GCAT)*, IEEE, pp. 1–7.

Standard, A. E. (2001). Federal information processing standards publication 197, *FIPS PUB* pp. 46–3.

Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N. and Robinson, S. (2017). Deep learning for unsupervised insider threat detection in structured cybersecurity data streams, *arXiv preprint arXiv:1710.00811* .

Wahab, O. F. A., Khalaf, A. A., Hussein, A. I. and Hamed, H. F. (2021). Hiding data using efficient combination of rsa cryptography, and compression steganography techniques, *IEEE Access* **9**: 31805–31815.

Zaw, T. M., Thant, M. and Bezzateev, S. (2019). Database security with aes encryption, elliptic curve encryption and signature, *2019 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF)*, IEEE, pp. 1–6.