Applications of RSA and AES256 in End-to-End encryption using Diffie-Hellman Key Exchange

Agoorukasetty Adithya[1], Kunal Kulkarni[2], Sanskrita Saha[2]

[1]*Student, Department of Software Systems, Vellore Institute of Technology, Tamil Nadu, India*
[2]*Student, Department of Analytics, Vellore Institute of Technology, Tamil Nadu, India*

---***---

Abstract – *Diffie-Hellman Key Exchange is an exponential key exchange technique which allows two users to exchange a secret key securely. This technique can further be extrapolated for encryption of messages. In this paper, we use RSA to encrypt the Diffie Hellman key exchange, which prevents the possibility of a man-in-the-middle attack and keeps the entire chat truly end-to-end encrypted. A detailed discussion regarding works related to this field and the novelty provided by this work in particular has also been discussed. In addition to the RSA, the AES256 algorithm is also discussed with reference to the Diffie Hellman Key Exchange. A detailed statistical analysis with a comparison of all the algorithms has also been provided.*

Key Words: (Cipher Text, Plain Text, Encryption, Decryption, Man-in-the-middle (MITM) attacks, Elliptic curve cryptography

1.INTRODUCTION

One of the most important developments in public-key cryptography was the Diffie-Hellman key exchange, which is still used in a variety of security protocols today. It allows two people who have never met before to create a secure key that they can use to encrypt their communications. In hospitals, the medical records of patients are stored in a highly confidential cloud. However, sometimes they have to be sent from one department to another or even from one hospital to another. In this process that data can be hacked and can be corrupted. The most common methodology used to prevent attackers from hacking or getting a hold of a message would be to encrypt the message through code. In such a situation, if no external source can crack the code, the data will be safe, but that cannot be promised or fool-proof; what would happen if one can't arrange for a code beforehand? The Diffie-Hellman key exchange was the first widely utilised solution to this problem. Even over an insecure channel that enemies may be watching, the technique allows persons who have never met before to safely construct a shared key.

Symmetric algorithms, which can be used to encrypt the majority of the data but are less efficient than public key algorithms, are the most effective kind of encryption. Although sharing sensitive data, data breaches have become more frequent. In many fields, this has been the case. Those who demand higher degrees of security yet haven't received a lot of attention; for instance: Healthcare. The goal of this project is to employ blockchain technology to provide safe data sharing and communication. The RSA and AES256 encryption methods, as well as the Diffie-Hellman Key Exchange, encrypt messages from end to end. The Diffie-Hellman key's main purpose is to build shared secrets that can be used to produce keys safely. These keys can then be combined with symmetric-key methods to send information securely.

On a mathematical level, the Diffie-Hellman key exchange's security is based on one-way functions. These are calculations that are straightforward in one direction but far more complicated in the other. While implementing this we are considering the situation where two clients are communicating with each other and sharing sensitive data.

1.1 Related Works

Emmanuel Bresson, Olivier Chevassut, David Pointcheva et al. [1] observed that the third tier in the treatment of group Diffie-Hellman key exchange utilizing public/private key. The first tier was created for a scenario in which the group membership is static, while the second was created as an extension of the first to allow for membership changes. The group Diffie-Hellman key exchange is given new features in this article, including strong corruption, concurrent protocol executions, tighter reduction, and a standard model.

Rewagad, Pawar et al. [2] researched on security of the secrecy of data kept in the cloud by combining digital signatures and Diffie Hellman key exchange with the Advanced Encryption Standard encryption technique. Even if the key in transit is hacked, the Diffie Hellman key exchange facility renders it useless, because the key in transit is useless without the user's private key, which is only accessible by the legitimate user. This proposed three-way technique makes it difficult for hackers to break the security system, securing data stored in the cloud. Chia-Hui Wang, Yu-Shun Liu et al. [3] worked towards considerably stronger privacy protection for an end-to-end VoIP to significantly improve VoIP security and prevent hostile eavesdroppers. This trustworthy solution uses the Elliptic-Curve Diffie-Hellman algorithm for key negotiation as well as the key generation function for dynamically updating keys during a VoIP conversation session. During the end-to-end call session, this 2-tier key distribution technique provides effective and robust security for VoIP voice packets. This proposed system has been tested over the Internet using an open-source SIP-based phone.

Jin Wook Byun, Dong Hoon Lee et al. [4] focused on two secure N-party EKE protocols in which clients can generate a common group session key using only their different passwords, and we proved their security using the group computational Diffie-Hellman, computational Diffie-Hellman, and 3 multi-decisional Diffie-Hellman assumptions. Designing N-Party EKE protocols in a dynamic environment and proving the methods in the standard assumption could be the next step

Radek Fujdiak, Petr Blazek et al. [5] highlighted difficulties are discussed, and three cryptographic encryption solutions (AES, ChaCha, and OTP) are compared in terms of the core IoT triad of performance, security, and cost. In a real-world application, we study encryption techniques and characterize their energy usage. The findings are valuable in determining the cost of enabling security features and in determining the most efficient encryption strategy.

Mohammed Riyadh Abdmeziema, Djamel Tandjaouib et al. [6] worked on Particularly resource-intensive cryptographic primitives are being offloaded to third parties as suggested in the paper. As a result, limited nodes enlist the help of more powerful entities in order to create a shared secret with any remote entity in a secure manner. Formal validation of our protocol's security attributes in order to evaluate it is performed. Both communication and processing expenses to see where we might save energy are also looked at. The results reveal that this approach saves a significant amount of energy while maintaining its security features.

Neetesh Saxena, Narendra S. Chaudhariet et al. [7] in the research, EasySMS, an efficient and secure protocol that enables end-to-end encrypted SMS communication between end users. The protocol's operation is demonstrated using two separate scenarios. The suggested protocol's study reveals that it can protect against a variety of attacks, including SMS disclosure, over-the-air modification, replay attack, man-in-the-middle assault, and impersonation attacks.

Avijit Mathur, Thomas Newe, Walid Elgenaidi, Muzaffar Rao, Gerard Dooly and Daniel Toal et al. [8] said the goal of this study is to look at a secure end-to-end IoT system. It enables wireless sensors/devices to connect to any PC on the planet while maintaining data and network security. The strategy suggested in this paper can defend an IoT solution from a variety of threats, including data breaches, unauthorized access, and denial of service (DoS). The findings suggest that the technologies installed or employed are effective. When compared to their counterparts in previously published studies, they are superior in terms of time and energy consumption.

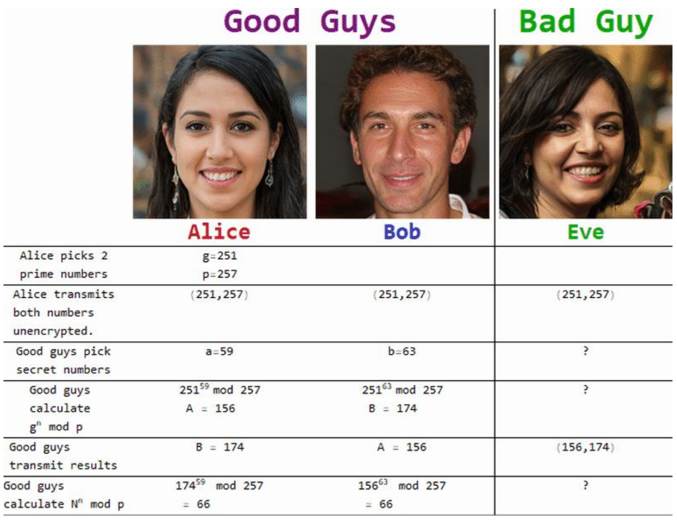
Patgiri et al. [9] said that the main goal is to provide the Diffie-Hellman Algorithm with guaranteed security. As a result, privateDH does not share the data with the designated party in a public manner. Instead, during the key exchange protocol, privateDH encrypts all shareable data with the AES algorithm. To avoid a man-in-the-middle attack, privateDH gets the public key using the RSA technique. As a result, how to secure the Diffie-Hellman algorithm against several types of potential future assaults is shown.

Hasib et al. [10] worked on the fundamental mathematics underpinning the AES and RSA algorithms, as well as a brief discussion of various cryptographic primitives often employed in the field of communication security, is presented in this work. It also covers multiple computational challenges, as well as an investigation of AES and RSA security aspects against various types of attacks, as well as countermeasures.

2. Proposed Work and Algorithms

2.1 Proposed Work

On a mathematical level, the Diffie-Hellman key exchange's security is based on one-way functions. These are calculations that are straightforward in one direction but far more complicated in the other.


Good Guys
Bad Guy

	Alice	Bob	Eve
Alice picks 2 prime numbers	g=251 p=257		
Alice transmits both numbers unencrypted.	(251,257)	(251,257)	(251,257)
Good guys pick secret numbers	a=59	b=63	?
Good guys calculate g^n mod p	251^{59} mod 257 A = 156	251^{63} mod 257 B = 174	?
Good guys transmit results	B = 174	A = 156	(156,174)
Good guys calculate N^n mod p	174^{59} mod 257 = 66	156^{63} mod 257 = 66	?

Fig. 1 Interpretation of the proposed model

While implementing this we are considering the situation where two clients are communicating with each other and sharing sensitive data. The procedure for the proposed model is as follows:

A. Server Initiation

- The process starts with the initiation of a server which handles the duty of exchange of messages between the two clients in an encrypted way.

- Generation of Diffie Hellman parameters, generator and prime are done by the server script itself for enhanced encryption.
- Identify the applicable funding agencies here. If none, delete this text box.

B. Server Connection

Now, we connect both clients to the server using the server's IP address. Once connected, the clients exchange their RSA public keys and on the other hand, the server transmits the Diffie-Hellman parameters to both clients. The users choose a private key for the Diffie Hellman key exchange and compute their respective public keys.

C. Encryption

These public keys are encrypted using the RSA public key of the other users and are shared. Once both clients receive the Diffie-Hellman public keys of the other client, the clients compute the final shared key. The thus obtained key can be utilized for encrypting all communication between the two clients. Since we use RSA to encrypt the Diffie Hellman key exchange, we help prevent the possibility of a man-in-the-middle attack and keep the entire chat truly end-to-end encrypted.

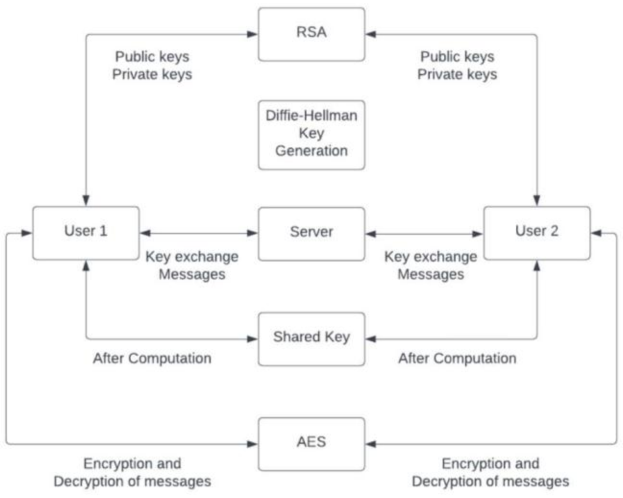


Fig. 3: Flowchart for the proposed system

2.2 Algorithms: Diffie Hellman key exchange

The Diffie-Hellman key exchange, also known as an exponential key exchange, is a technique for digital encryption that makes breaking codes statistically impossible by using numbers raised to particular powers to create decryption keys based on elements that are never physically conveyed.

Elliptic Curve Cryptography, a technique for performing public-key cryptography based on the algebraic structure of elliptic curves over finite fields, is the foundation of the algorithm. The trapdoor function is used by the DH as well, much like many other methods to public-key cryptography is used. The following is a straightforward concept for comprehending the DH Algorithm.

1) The first party chooses the prime numbers g and p and informs the second party of them.

2) The second party then chooses a secret number (let's call it a), computes ga mod p, and transmits the result—call let's it result A—back to the first party. Remember that just the outcome is given to the recipient; the secret number is not.

3) The first party then follows suit, choosing a secret number b and computing the result B in a manner akin to.

4) The second party is then informed of this outcome.

5) The second party calculates Ba mod p using the received value B.

6) The first party computes Ab mod p using the received number A. The solution to step 5 is the same as the answer to step 4, which is where things get interesting. This means that regardless of the sequence of exponentiation, the solution will be the same for both parties.

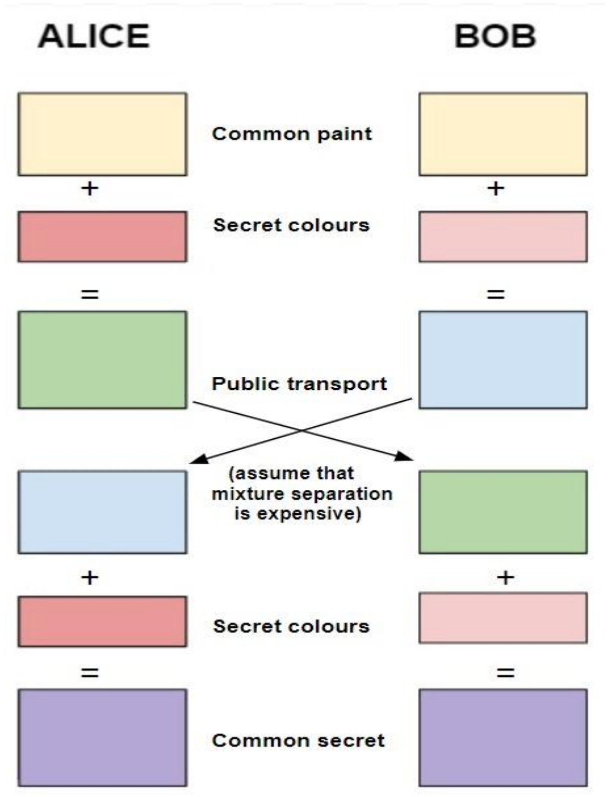


Fig. 3 Diffie-Hellman Key Exchange

2.3 Algorithms: RSA Algorithm

Asymmetric cryptography uses the RSA algorithm. Asymmetric means that it uses both the public key and the private key, which are two separate keys. As implied by the name, the private key is kept secret while the public key is distributed to everyone.

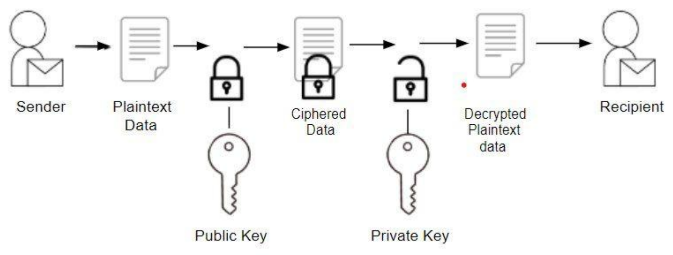


Fig. 4 RSA Encryption Algorithm

Actions required:

MAJOR GENERATION

1. Choose P and Q, two prime numbers.

2. Determine n such that n = P*Q.

3. Determine (n) such that (n) = (P-1) (Q-1)

4. Choose a tiny exponent e such that it is an integer and not a factor of n, and that its value is 1 e. (n)

5. public key is (n,e)

6. Determine d such that it equals (k*(n) + 1) / e for some integer k.

7. The secret code is (n.d)

ENCRYPTION

Text cypher =p e mod n

p = plaintext

DECRYPTION

Plain text is equal to c mod n.

c = cypher text, etc.

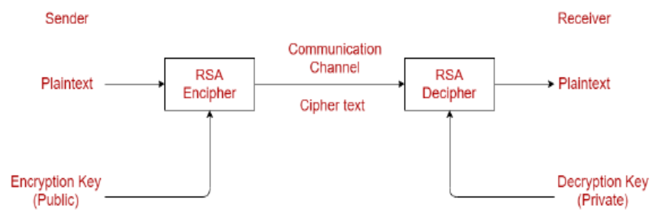


Fig. 5 RSA Decryption Algorithm

2.4 Algorithms: AES256 Algorithm

AES employs symmetric key encryption, which encrypts and decrypts data using just one secret key. The US National Security Agency has authorised the Advanced Encryption Standard as the first and only publicly available cypher for securing top-secret data. Each unit of data is replaced with a different one based on the security key, which is the fundamental tenet of all encryption techniques. AES was created as a substitution-permutation network, to be more precise. AES uses a key expansion mechanism that leverages the initial key to generate a set of new keys known as round keys, adding additional security. Multiple rounds of modification are used to generate these round keys, each of which makes them more difficult to crack the encryption.

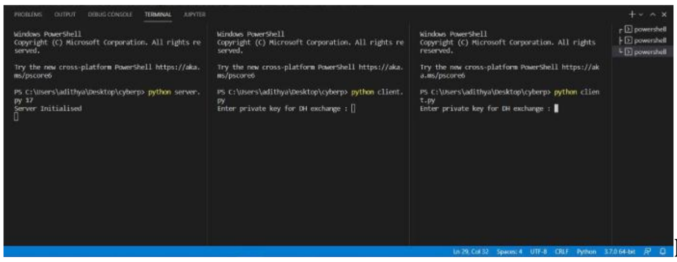
Using an XOR cypher, which is an operation pre-built into processor hardware, the initial key is first appended to the block. After then, each data byte is replaced with a new one according to a specified table. The 44 array's rows are then shifted: the second row's bytes are moved one space to the left, the third row's bytes are moved two spaces, and the fourth row's bytes are moved three places. The four bytes in each column are then combined by a mathematical process that mixes the columns. The next step is to add the round key to the block (just like the first key did), and the procedure is repeated for each round.

This results in ciphertext that differs greatly from plaintext. The same method is used in reverse for AES decryption. The AES encryption method has several stages, each of which performs a crucial task. A far more complex outcome is obtained by using a different key for every round. The relationship between the original and encrypted content is obscured through byte replacement, a nonlinear modification of the data. Data is diffused by moving the rows and columns around, and encryption is made more difficult by bytes being switched around. While mixing diffuses the data vertically, shifting does so horizontally. The end product is an extremely complex form of encryption.

3. Implementation and Performance Evaluation

3.1 Implementation

Server Initiation



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights re
served.

Try the new cross-platform PowerShell https://aka.
ms/pscore6

PS C:\Users\adithya\Desktop\cyberp> python server.
py 17
Server Initialised
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights re
served.

Try the new cross-platform PowerShell https://aka.
ms/pscore6

PS C:\Users\adithya\Desktop\cyberp> python client.
py
Enter private key for DH exchange :
```

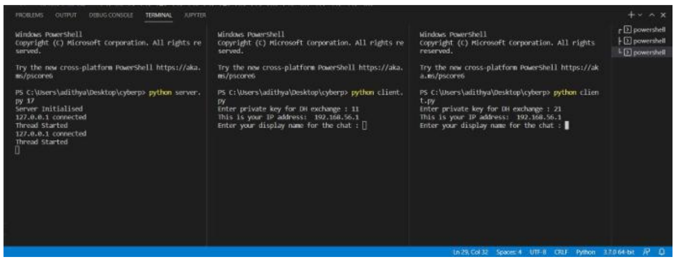
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights
reserved.

Try the new cross-platform PowerShell https://ak
a.ms/pscore6

PS C:\Users\adithya\Desktop\cyberp> python clien
t.py
Enter private key for DH exchange :
```

Fig. 6 Server Initialization

Private Key Exchange



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights re
served.

Try the new cross-platform PowerShell https://aka.
ms/pscore6

PS C:\Users\adithya\Desktop\cyberp> python server.
py 17
Server Initialised
127.0.0.1 connected
Thread Started
127.0.0.1 connected
Thread Started
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights re
served.

Try the new cross-platform PowerShell https://aka.
ms/pscore6

PS C:\Users\adithya\Desktop\cyberp> python client.
py
Enter private key for DH exchange : 11
This is your IP address:  192.168.56.1
Enter your display name for the chat :
```

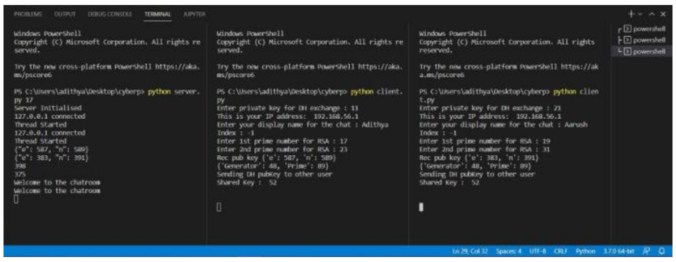
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights
reserved.

Try the new cross-platform PowerShell https://ak
a.ms/pscore6

PS C:\Users\adithya\Desktop\cyberp> python clien
t.py
Enter private key for DH exchange : 21
This is your IP address:  192.168.56.1
Enter your display name for the chat :
```

Fig. 7 Private Key Exchange

Exchange of messages



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights re
served.

Try the new cross-platform PowerShell https://aka.
ms/pscore6

PS C:\Users\adithya\Desktop\cyberp> python server.
py 17
Server Initialised
127.0.0.1 connected
Thread Started
127.0.0.1 connected
Thread Started
{"e": 587, "n": 589}
{"e": 383, "n": 391}
398
375
Welcome to the chatroom
Welcome to the chatroom
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights re
served.

Try the new cross-platform PowerShell https://aka.
ms/pscore6

PS C:\Users\adithya\Desktop\cyberp> python client.
py
Enter private key for DH exchange : 11
This is your IP address:  192.168.56.1
Enter your display name for the chat : Adithya
Index : -1
Enter 1st prime number for RSA : 17
Enter 2nd prime number for RSA : 23
Rec pub key {'e': 587, 'n': 589}
{'Generator': 48, 'Prime': 89}
Sending DH pubKey to other user
Shared Key :  52
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights
reserved.

Try the new cross-platform PowerShell https://ak
a.ms/pscore6

PS C:\Users\adithya\Desktop\cyberp> python clien
t.py
Enter private key for DH exchange : 21
This is your IP address:  192.168.56.1
Enter your display name for the chat : Aarush
Index : -1
Enter 1st prime number for RSA : 19
Enter 2nd prime number for RSA : 31
Rec pub key {'e': 383, 'n': 391}
{'Generator': 48, 'Prime': 89}
Sending DH pubKey to other user
Shared Key :  52
```

Fig. 8 Exchange of messages

3.2 Performance Evaluation

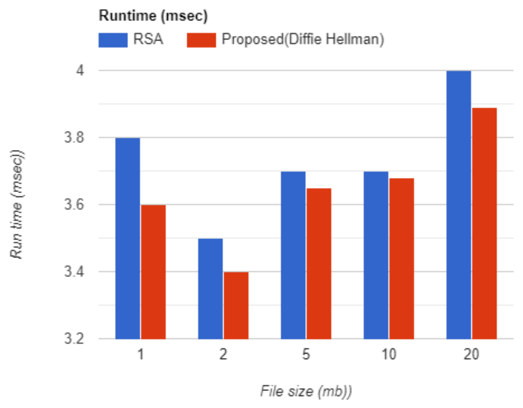


Chart-1: Runtime Complexity Comparison Bar Graph

Comparison between the standard RSA algorithm run time and the runtime of threads in the proposed algorithm has been done.

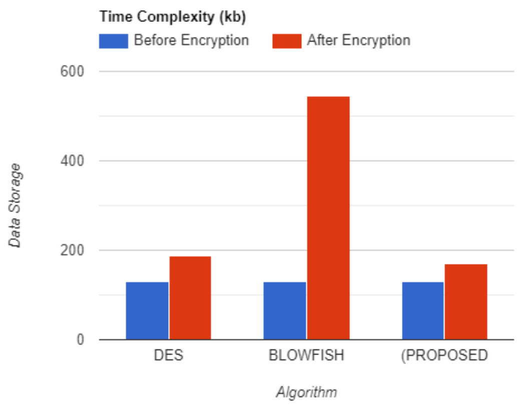


Chart-2: Space-Time Complexity Comparison Bar Graph

Comparison between DES, Blowfish and the proposed algorithm is done on terms of the space occupied before and after encryption.

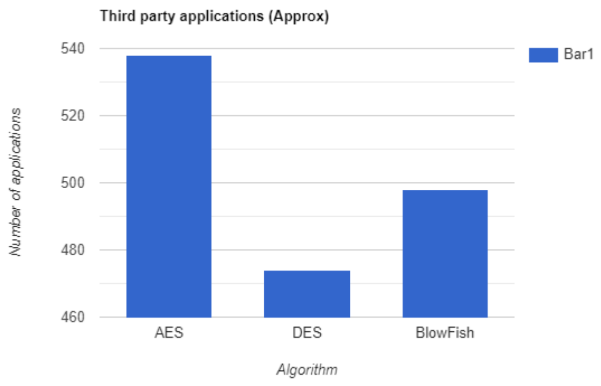


Chart-3: Third-party applications involvement Bar Graph

Our proposed algorithm does not use any third party applications and hence reduces the time and space complexity.

3. CONCLUSIONS

This paper presents a proposed solution of using RSA algorithm to encrypt Dille Hellman Key-Exchange. We also compared the runtime complexity of the standard RSA algorithm with our proposed solution and we found that the proposed solution has a better runtime complexity. Our proposed solution is also most efficient in space-time complexity. Since our proposed solution does not use any third-party applications, it reduces the time and space complexity. Our future work will explore more on this concept and we will apply more algorithms in a sequential or parallel way to ensure secure environment for data storage.

REFERENCES

[1] [1] Nan Li, "Research on Diffie-Hellman key exchange protocol," 2010 2nd International Conference on Computer Engineering and Technology, 2010, pp. V4-634-V4-637,doi:10.1109/ICCET.2010.5485276

[2] Bresson, E., Chevassut, O., & Pointcheval, D. (2001, December). Provably authenticated group Diffie-Hellman key exchange—the dynamic case. In International Conference on the Theory and Application of Cryptology and Information Security (pp. 290-309). Springer, Berlin, Heidelberg. Computer Engineering and Technology, 2010, pp.

[3] Wiener, M. J. (1990). Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information theory, 36(3), 553-558..

[4] Boneh, D. (1999). Twenty years of attacks on the RSA cryptosystem. Notices of the AMS, 46(2), 203-213.

[5] Kocher, P. C. (1996, August). Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Annual International Cryptology Conference (pp. 104-113). Springer, Berlin, Heidelberg.

[6] Boneh, D., & Franklin, M. (1997). Efficient generation of shared RSA keys. In Annual international cryptology conference (pp. 425-439). Springer, Berlin, Heidelberg.

[7] Biryukov, A., Khovratovich, D., & Nikolić, I. (2009, August). Distinguisher and related-key attack on the full AES-256. In Annual International Cryptology Conference (pp. 231-249). Springer, Berlin, Heidelberg.

[8] Al Hasib, A., & Haque, A. A. M. M. (2008, November). A comparative study of the performance and security issues of AES and RSA cryptography. In 2008 third international conference on convergence and hybrid information technology (Vol. 2, pp. 505-510). IEEE.

[9] Kumar, B. S., Raj, V. R., & Nair, A. (2017, April). Comparative study on AES and RSA algorithm for medical images. In 2017 international conference on communication and signal processing (ICCSP) (pp. 0501-0504). IEEE.

[10] Boneh, D. (1999). Twenty years of attacks on the RSA cryptosystem. Notices of the AMS, 46(2), 203-213.