

A HYBRID SECURITY FRAMEWORK FOR MEDICAL DATA IN IOT APPLICATIONS

Phani Sridhar Addepalli

Associate Professor, Department of CSE, Aditya Engineering College (A), Surampalem,
Andhra Pradesh, INDIA
Phani.addepalli@aec.edu.in

Dr.P.V.Lakshmi

Professor, Department of CSE, Gitam University, Visakhapatnam,
Andhra Pradesh, INDIA
vpanga@gitam.edu.in

Abstract

When sensitive data stored and transmit over the internet while information is no longer protected based on physical boundaries, security is considered as the significant issue. The secure communication between different entities is ensured by an effective, essential, and efficient component known as cryptography, which transmit unintelligible data and allow the authorized recipient to access the data. A hybrid public key cryptography system is presented in this paper based on the Diffie Helman key exchange algorithm and Elliptic curve cryptography. Elliptic curves are used to generate the private keys at the user end. These private keys are used to compute the public keys which can then be shared between multiple parties forming a group. The proposed modified key generation process helps reducing the key size which helps in effectively exchanging the keys over internet. The proposed algorithm achieves this without compromising the cryptographic efficiency. Advance Encryption Standard (AES) is used for encryption of message based on the shared secret key. The shared secret key is hashed using SHA256 algorithm to provide an additional layer of security. The proposed hybrid model is compared with conventional Elliptic Curve Diffie Helman (ECDH) key exchange based cryptographic algorithm in terms of key length, run time, throughput and avalanche effect. The proposed method produced better results in terms of key length and run time while maintaining good throughput and avalanche effect.

Keywords: Cryptography system; Elliptic Curve Cryptography; Diffie Helman key exchange algorithm; Advance Encryption Standard (AES), SHA256.

1. Introduction

A new digital existence acquires with the improvements of Internet of Things (IoT) in the physical universe. The expectation of being connectivity and collaboration is included in many objects [1]. The efficient, intelligent, and agile way should adapt in the network architecture to maintain the quality of provided services through the connection of trillions or billions of IoT devices for the cloud in order to exchange, process, and store the data while concerning the networks and devices with heterogeneity [2]. The significant challenges have faced by the future IoT in addition to the benefits of a traditional, and centralized cloud model. Those challenges are mobility support, awareness of location, data volume, velocity, latency, or monopoly versus an open IoT contention [3]. The Internet of Medical Things has become the great area of interest because it uses for predicting the diseases and health monitoring and treatment. Here, these performance parameters have controlled [4-5].

Based on the communication resources, storage, and additional computing for specific operations, the challenges have addressed using edge computing. Thus, both the cloud and IoMT devices liberate and the conventional cloud computing services performance improve [6]. The security is the major concern while using the edge computing. Some of the security challenges in a cloud inherit and attributes to new threats and vulnerabilities by the edge. However, the threats include access control, authentication, and protection of privacy, secure data storage, and data computation [7]. The authors specifically concentrate on data privacy preservation that sent by IoMT devices to the cloud with the implementation of edge computing while the edge devices and the cloud allow the data origin and integrity authentication. The ability of describing who you are and say you are using authentication. If the receiver can detect the message of a sender unequivocally, authentication is required for data exchanging in a communication network. When unauthorized systems or users not creating, modifying, or deleting the information or message, integrity is resulted.

2. Related work

D. Koo and J. Hur (2018) [8] and J. Zhang et al., [9] (2019). In the collaborative social applications, vehicle or drive data is considered for advanced health systems and other cases like intelligent traffic systems (ITS) that manage the data about peoples, data privacy is needed in terms of integrity and anonymity. It has been encouraged that the proposals recommend in the other communication areas. In the related literature, various research works address the data privacy preservation in IoT.

J. Cui et al., (2019), presents a public key ECC-based solution [10] for intelligent transportation environments, where the authentication of vehicles has included with a shared allocation between the road side unit (RSU) and the vehicles. The vehicles nearer to the RSU and with better resources of computation have chosen as edge nodes. The responsibility of these vehicles is to authenticate the transmitted messages by nearby vehicles, embedding batch authentication. Additionally, they sent the results to the RSU, which checks out the earlier processed data. To enhance the speed of process, a cuckoo filter and fuzzy logic have used. The trusted third-party authorities by all entities are there to ascertain the vehicles' real identity. In Z. Guan et al., (2019) [11], the same approach is developed. For authenticating both unmanned aerial vehicles (UAV), and vehicles, various Bloom filter probabilistic data structures have incorporated in S. Garg et al., (2018) [12]. In Bloom filters, the hashing and storing of vehicles IDs have authenticated under the coverage of UAV.

X. Li et al., (2019) [13] has introduced a homomorphic Boneh-Goh-Nissim-based technique to preserve the scenarios of mobile edge computing. From the perspective of a security, the solution is robust and interesting. In F. Wu et al., (2018) [14], this method performance evaluation was presented earlier. H. Wang et al., (2018) [15] and Z. Wang (2017) [16] have proposed same approaches. These proposals have implemented based on the homomorphic encryption for providing the confidentiality. Instead of using the information about device identification, pseudonyms use for privacy if data forward to the cloud from the edge or fog computing device. The authors have shown that the aggregation at the edge or fog device allows to transmit the data for the cloud efficiently in terms of overhead than the other techniques. An intermediate element like fog or edge device includes for data aggregation and data privacy based on the comparable results. The homomorphic encryption has a limitation of energy consumption in IoT device. In M. Alkharji et al., (2018) [17], the authors indicate the improvements incorporate into homomorphic encryption by reducing or resolving the challenges.

The determined contributions in the scientific literature limits the novelty specifically for the IoMT paradigm. Deebak et al., (2019) [18] has presented the technique of anonymous and secure user authentication based on biometric information to achieve the security of communications in healthcare applications. Based on the elliptic cryptography, the proposal was improved by integrating the smart cards with the stored biometric data of users. A key generation process has initiated after authenticating the user and securing the communication channel would be made based on the key. The authors have concluded that the proposal results in two possible limitations like the congestion and necessity of implementing physical smart cards (an active method from the users perspective) if a huge number of IoMT devices.

I. S. Farahat et al., (2018) A novel technique [19] has proposed based on the Advanced Encryption Standard (AES) for encoding and encryption in IoMT. The proposal performance has tested practically and has the benefit that the shorter time requires for encryption and encoding processes than the conventional cryptographic methods. In S. Pirbhulal et al., (2019) [20], the authors propose a mechanism of key generation based on the biometric data that can use as an input. For encryption of medical data, the keys were incorporated and the proposal outperforms compared to the other existing methods.

P. Jyotheeshwari and N. Jeyanthi et al., (2020) [21] have developed an efficient architecture using cloud environment for dynamic data storage. A security model is proposed to provide the data integrity and confidentiality while avoiding security issues. The efficient access control mechanism has developed with the integration of ABE and symmetric cryptography to reduce the computation overhead. The proposed algorithm efficiency has proved in terms of security, scalability, and access control.

Adarsh T. Kalpally and K. P. Vijaykumar et al., (2021) [22] have proposed MEHR algorithm to validate the messages provenance. The health record of patients can encrypt using ABE approach. Excellent security and privacy in IoT have resulted based on the experimental results.

Kedir Memo Bashir et al., (2020) [23] has proposed an encrypted for IoT sensor devices. A cryptography algorithm has proposed that incorporates at the sensor device. It has verified the two-level encryption and decryption at the IoT sensor level and the receiving end respectively. The effective security solution has achieved with the proposed technique for healthcare data transmission in IoT.

3. Proposed method

The proposed hybrid encryption model consists of three stages; Elliptic curve based key generation, Diffie-Hellman key exchange and AES encryption. This section presents the details of every individual algorithm used and the proposed modified key generation process which reduces the size of the keys while maintaining

cryptographic efficiency. There are several scenarios where an encrypted message needs to be broadcasted to a group of parties. This task can be accomplished securely using Diffie-Hellman key exchange. Only the members who are a part of the group can decrypt the secret message. This reduces the number of encryption operations that needs to be performed on a broadcast message. The key exchange within a group can be carried out effectively with minimum resources. Once the public keys are generated, each user can generate their own unique shared keys to encrypt and decrypt a message. The figure 1 shows the proposed method.

3.1. Elliptic Curve Cryptography

Elliptic curve cryptography is a mathematical problem called the discrete logarithm problem on an elliptic curve as the basis of security encryption. In 1985, cryptographers Victor Miller and Neal Koblitz devised it at about the same time as their independence. Elliptic curve DSA (ECDSA) and elliptic curve Diffie-Hellman key exchange (ECDH) are often used as specific encryption methods. The figure 2 shows Elliptic Curve Cryptography.

An elliptic curve is a curve expressed in the form of $y^2 = x^3 + ax + b$, and defines a special addition method for the relationship between the coordinates of two points on this curve. "Double" can be obtained by adding itself to a certain point G, but by repeating this, an arbitrary n times (nG) can be obtained. It is also known that if the addition is continued, it will return to G again after p times, and any addition result will point to any of the p points.

When G is multiplied by n to reach the point X, it is easy to find X from n and G, but it is difficult to find n from X and G (X is how many times G is multiplied). No efficient method has been found. This is called the Discrete Logarithm Problem (ECDLP) on an elliptical curve, and even if the public key is known by generating the private key based on n and the public key based on G and X. It is possible to construct public key cryptography in which it is difficult to determine the private key.

This approach uses $\{P, a, b, G, n, h\}$,
P = Field that the curve is define over,
G = Generation point,
a, b = values define the curve
h = co- factor,
n = Prime order of G

However, depending on the parameters of the curve and how to select the point G to start the calculation, n may be easily calculated. Therefore, in practice, it is necessary to set each value so that p is a sufficiently large prime number.

Other public key cryptography is the RSA cryptography, which develops by considering the difficulty in product factoring of huge prime numbers, but the elliptic curve cryptography can obtain the same level of cryptography with a shorter cryptographic key. Since the number of calculations involved in encryption and decryption is small, it has been put into practical use from early on in devices with limited processing power such as IC cards, and it is being introduced and replaced in applications where RSA has been used.

3.2. Diffie-Hellman key sharing

Diffie-Hellman key exchange is one of the procedures for securely exchanging confidential information such as encryption keys over untrusted communication channels. The prototype of public-key cryptography was devised in 1976 by Bailey Whitfield Diffie and Martin Edward Hellman.

Common key encryption (private key encryption), which uses the same encryption key for encryption and decryption, requires that the sender and receiver who exchange messages via a dangerous route such as the Internet share the same encryption key in advance. There was a difficult problem (key delivery problem) that it was not possible to use the same dangerous route as the message for key delivery, and a secure exchange method had to be prepared separately. The below figure 3 shows the Diffie-Hellman key exchange.

Diffie-Hellman key exchange uses a mathematical problem called the discrete logarithm problem. For a large prime number p, select natural numbers g and x that are less than p, and divide g by the x^{th} power to calculate the remainder m. It is easy to calculate m by giving a certain x, assuming that p and g are known, but it is difficult to calculate the corresponding x based on a certain m.

Now, when A and B want to exchange keys, they first decide and share p and g (these two may be known to a third party), and A and B are their own. Determine the secret natural numbers a and b. A sends B the remainder $i (= g^a \bmod p)$ of g divided by a, and B also sends A similar value $j = g^b \bmod p$. If A performs the same calculation $k = j^a \bmod p$ for the sent j, and B also performs the same calculation for i, both have the same value k, and this is used as the encryption key.

If the value of the prime number p is sufficiently large with respect to the computing power of the computer, even if a third party eavesdrops on the information transmitted and received during the key exchange process, it

will be difficult to determine the key itself. However, the Diffie-Hellman key exchange itself cannot prevent a **man-in-the-middle attack** that interrupts the communication between the two parties and replaces the information to impersonate the other party. In addition, it has been reported that the same prime number is repeatedly used in many systems in practice, and it has been pointed out that the difficulty of decoding is reduced if the prime number is fixed.

3.3. Elliptic Curve Diffie-Hellman

Elliptic Curve Diffie-Hellman (Elliptic Curve Diffie-Hellman Key Exchange, ECDH) is, without a share of the pre-secret, eavesdropping the communication channel with the possibility of it is a public key cryptography cryptographic protocol that enables sharing of cryptographic keys. The secret value shared by both parties can be used as it is, or after some conversion, it can be used as a key for symmetric key cryptography. Diffie-Hellmann key sharing is one of the elliptic curve cryptosystems modified to use elliptic curves. The figure 4 shows the Elliptic Curve Diffie-Hellman.

Alice wants to build a common key with Bob, but suppose that the communication between the two has only a line that could be eavesdropped by a third party. First, the elliptic curve used between two people (that is, the finite field K , the cubic expression that determines the curve, the base point G , its order n is decided. And both are on this elliptic curve, the secret key d ($[1, n - 1]$ Randomly selected integer from) and public key $Q(Q = dg$, That is, multiplication on an elliptic curve) is generated. Here, Alice's key pair d_A, Q_A , Bob's (d_B, Q_B). Then exchange public keys with each other.

Next, Alice $(x_k, y_k) = d_A Q_B$ Bob said $(x_k, y_k) = d_B Q_A$. The calculation is performed. $d_A Q_B = d_A d_B G = d_B d_A G = d_B Q_A$ So, Alice and Bob are the same x_k (X coordinate on the elliptic curve) can be obtained, so this is a shared secret. Most standardization protocols based on ECDH use a hash function or the like to generate a common key based on this secret.

Since Alice's public key is the only information that Alice sends on the communication path during the key exchange process, only herself can know Alice's private key (unless she can solve the discrete logarithm problem on the elliptic curve). Similarly, for Bob, the private key is never known to a third party. Also, unless the Diffie-Hellmann problem on the elliptic curve can be solved, it was shared with a third party who was eavesdropping on the communication path and x_k cannot be calculated.

The public key can be static or temporary (ephemeral, especially abbreviated as ECDHE in this case). The temporary key is not always authenticated, so if authentication is required, it will be authenticated by another method. Without authentication, it is vulnerable to man-in-the-middle attacks, just like regular Diffie-Hellman key sharing. If one of the keys is static, it will not be attacked by a man-in-the-middle, but it will not be able to participate in other advanced security such as forward secrecy. On the side with a static key, it is necessary to check the public key of the other party and use a secure common key generation function in order to prevent leakage of its own private key. It is possible to use the shared secret as a key as it is, but it is recommended to hash the secret to eliminate the effects of weak bits created by Diffie-Hellmann key sharing.

3.4. Modified Key Generation Process

The proposed modified key generation process is described in the following section. The public keys used in the encryption process are exchanged over the internet. The length of these keys plays a major role in the performance of the entire cryptographic system. Reducing the key size would improve the parameters of the system like key length, key generation time, encryption time, decryption time and the overall memory usage. The algorithm is shown below:

Algorithm I

1. Generate the private keys using EC technique.

2. Truncate the private key using

$$New_Private_key = Private_Key \& \text{\texttt{\$}}$$

Where

$$\text{\texttt{\$}} = 0\text{xx}$$

3. Generate the public key using

$$Public_Key = Private_Key * G.x$$

Where

$G.x$ is the x coordinate of the generator point

4. Truncate the public keys using

$$New_Public_key = Public_Key \& \text{\texttt{\$}}$$

5. The final encryption key should be hashed with SHA256 algorithm to a 64 key before sending it to AES algorithm.

Hash algorithms, for various message digest or secure hash algorithms, a common interface is implemented by this module. They include FIPS secure hash algorithms such as SHA512, SHA384, SHA256, SHA224, and SHA1 along with the RSA's MD5 algorithm. The terms 'message digest' and 'secure hash' can be interchangeable. Earlier, the algorithms were termed as message digests and we call them as secure hash recently.

Hash algorithms are mathematical functions that can be used to convert the data into hash codes, a fixed length hash values, or hashes. The original value summary is the output hash. It's not possible to get the original input data from hash values.

It's essential to prove that no one modifies the sending data although encryption is important for data protection or data confidentiality. Hashing values inform that if a file isn't changed since creation.

SHA256 is a type of secure hash algorithm that produces a fixed length one way string from any input data. The designing of algorithm is made in a way that two different inputs don't result the same hash value practically. The data integrity can verify using this property. By determining the hash again and comparing with the received hash, the data integrity can verify when hash and data is retrieved based on various techniques.

In security protocols and applications, the most common used algorithm is SHA256. Its hash value of a file is computed using the following python program. It's important to make a note that the conversion of computed hash into a readable hexadecimal string is carried out.

3.5. The AES algorithm

AES algorithm is also called as the Rijndael algorithm which is a symmetrical block cipher algorithm. It converts the plain text of 128 bits blocks into ciphertext based on the keys 256, 192, and 128 bits. AES algorithm considers as the secure standard worldwide.

The working of AES, for generated ciphertext, AES algorithm utilizes a SP network or substitution-permutation with multiple rounds. Depending on the usage of key size, the number of rounds are decided. A 256-bit key size includes 14 rounds, a 192-bit key size has 12 rounds, and a 128-bit key size includes 10 rounds. Figure 5 shows the rounds process. A round key is need for each round, but one key is as input to the algorithm. To get keys for each round, including round 0, this key requires to be expanded.

Steps in each round

Four different steps include in each round of algorithm.

1. Substitution of the bytes

Based on the rules that dictate by predefined S-boxes (short for substitution boxes), the substitution of block text bytes is made in the first step and as shown in figure 6.

2. Shifting the rows

As shown figure 7, all rows shift by one except the first one in the permutation step.

3. Mixing the columns

By mixing the columns of block, the message jumbles up more by using the Hill cipher in the third step and as shown in figure 8.

4. Adding the round key

XORed the message with the respective round key in the final step and it is shown in figure 9.

The final ciphertext is secure based on these steps done repeatedly. Brute force attacks try to crack the encryption by trying all the possible key combinations. But the AES algorithm is so secure that it's not been broken yet. With large keys, even modern computers take billions of years to crack an encrypted text. As the proposed method used keys generated by elliptic curves, the proposed framework is secure against brute force attacks.

4. Results

This section presents the experimental results conducted to verify the proposed method.

The parameters used for encryption/decryption are key generation time, total key length, encryption memory usage/peak usage, encryption time, decryption memory usage/peak usage, decryption time, throughput, avalanche effect and finally check the resultant decrypted message same as encrypted message or not. By comparing all the parameter's proposed method is better than existing method Elliptic curve cryptography key generation with AES.

5. Figures, Tables and Photographs

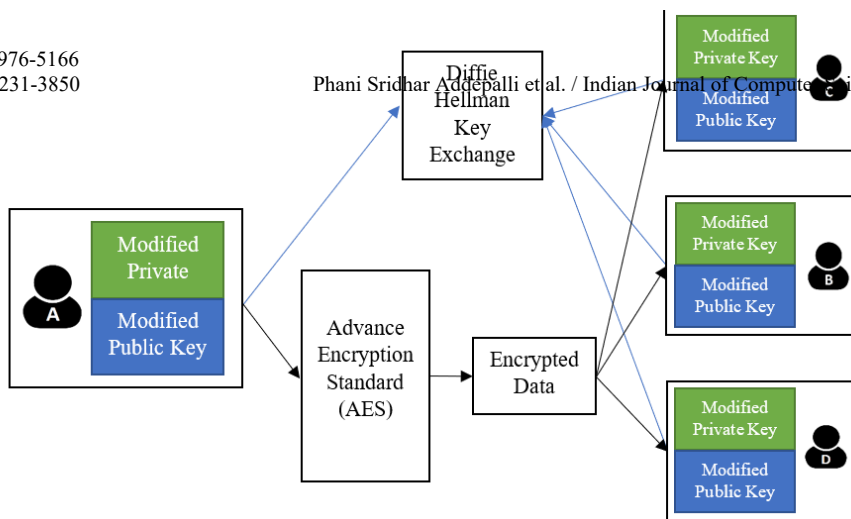


Figure 1: Proposed block diagram

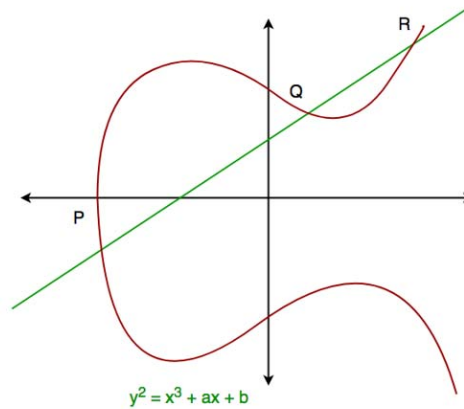


Figure 2: Elliptic Curve Cryptography

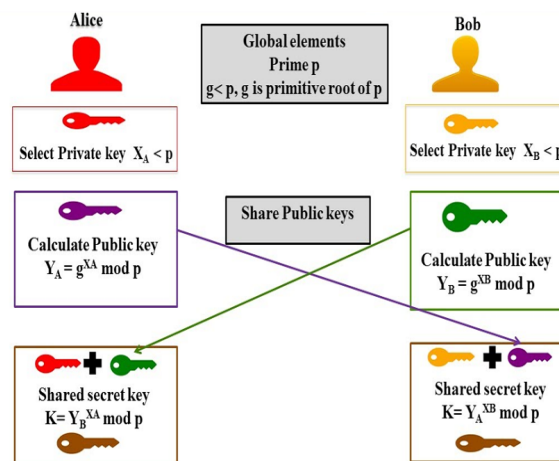


Figure 3: Diffie-Hellman key exchange

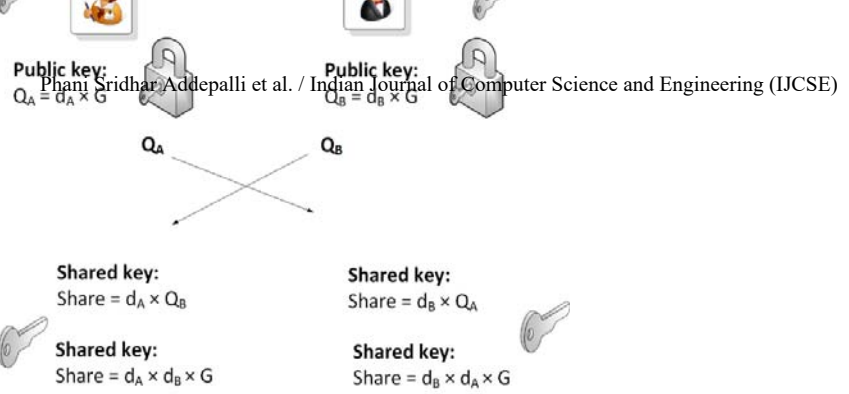


Figure 4: Elliptic Curve Diffie-Hellman

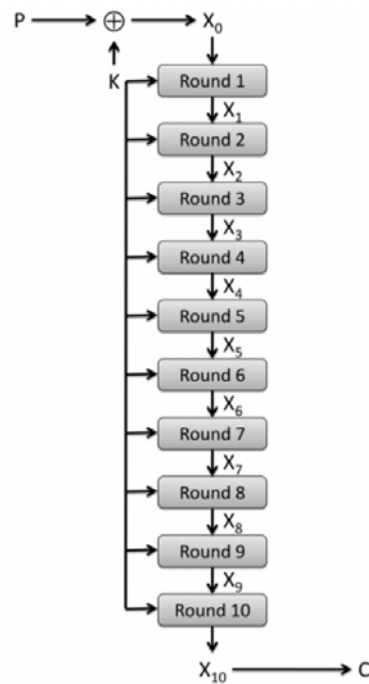


Figure 5: The number of rounds process in AES

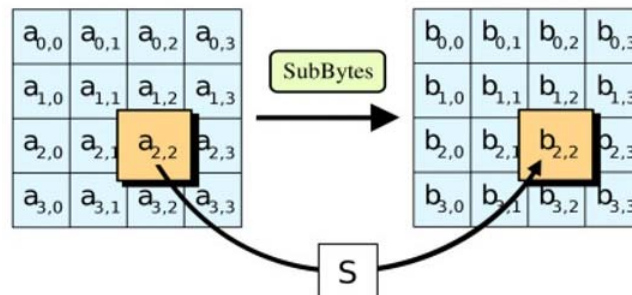


Figure 6: Substitution of the bytes

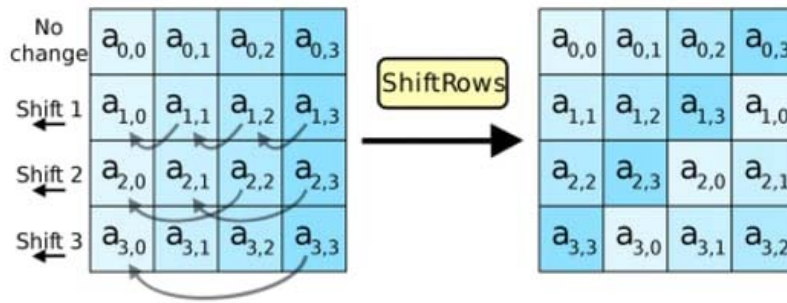


Figure 7: Shifting the rows

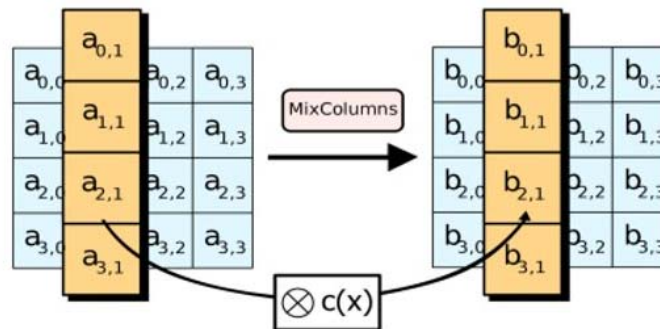


Figure 8: Mixing the columns

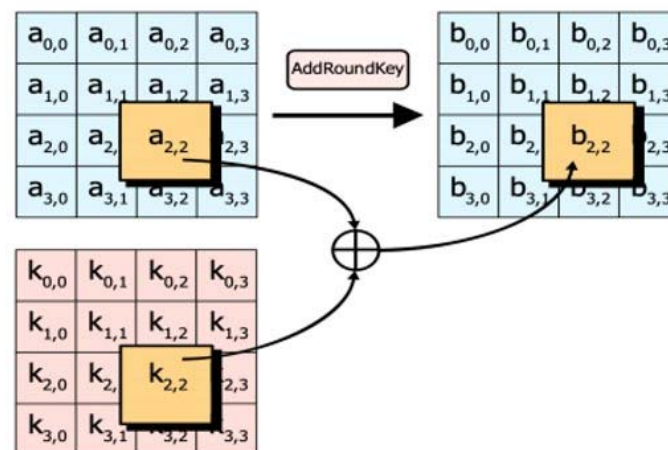


Figure 9: Adding the round key

The table 1 represents comparison results for medical data encryption/decryption with 3 people in the group.

Parameters	Elliptic curve cryptography key generation with AES	Proposed method
Key Generation time	0.680652 seconds	0.0 seconds
Total Keys length	717.75 bytes	296.125 bytes
Encryption memory usage / peak	0.07938 MB 0.082393 MB	0.03782 MB; 0.039985 MB
Encryption time	1.858308 seconds	0.009749 seconds
Decryption memory usage / peak	0.073107 MB 0.0821 MB	0.013515 MB 0.014479 MB
Decryption time	1.568212 seconds	0.0 seconds
decrypted message same as encrypted message	True	True
Throughput	1.3143046322245828 Kilobytes/sec	250.52649630731355 Kilobytes/sec
Avalanche effect	0.49708469721767595 %	0.5076207037643208 %

Table 1: Data encryption results on medical data with 3 people in the group

The table 2 represents comparison results for medical image encryption/decryption with 3 people in the group. The algorithms applied on 4 different medical images for encryption/decryption.





Input image	Parameters	Elliptic curve cryptography key generation with AES	Proposed method
	Key Generation time	0.692095 seconds	0.0 seconds
	Total Keys length	717.625 bytes	293.5 bytes
	Encryption memory usage / peak	0.179608 MB 0.260726 MB	0.112512 MB 0.192782 MB
	Encryption time	1.375024 seconds	0.003995 seconds
	Decryption memory usage / peak	0.133948 MB 0.211291 MB	0.091684 MB 0.166763 MB
	Decryption time	1.456036 seconds	0.004019 seconds
	decrypted message same as encrypted message	True	True
	Throughput	57.23053515793179 Kilobytes/sec	19697.962296620775 Kilobytes/sec
	Avalanche effect	0.5000124148034116 %	0.510581335584551 %
Input image	Image 2		
	Key Generation time	0.953811 seconds	0.0 seconds
	Total Keys length	718.625 bytes	295.875 bytes
	Encryption memory usage / peak	0.157659 MB 0.230508 MB	0.104243 MB 0.176244 MB
	Encryption time	1.405255 seconds	0.008 seconds
	Decryption memory usage / peak	0.141183 MB 0.21026 MB	0.083415 MB 0.150228 MB
	Decryption time	1.366671 seconds	0.0 seconds
	decrypted message same as encrypted message	True	True
	Throughput	50.25291784231332 Kilobytes/sec	8827.2705078125 Kilobytes/sec
	Avalanche effect	0.5000345877144439 %	0.5050207526286663 %
Input image	Image 3		
	Key Generation time	0.819991 seconds	0.0 seconds
	Total Keys length	718.125 bytes	294.5 bytes
	Encryption memory usage / peak	0.147875 MB 0.196804 MB	0.080323 MB 0.128404 MB
	Encryption time	1.603285 seconds	0.008004 seconds
	Decryption memory usage / peak	0.110039 MB 0.154284 MB	0.059567 MB 0.10246 MB
	Decryption time	1.458215 seconds	0.0 seconds
	decrypted message same as encrypted message	True	True
	Throughput	29.476224790040447 Kilobytes/sec	5904.396434595202 Kilobytes/sec
	Avalanche effect	0.5014523366418527 %	0.5080594499586435 %
Input image	Image 4		
	Key Generation time	0.905593 seconds	0.0 seconds
	Total Keys length	717.25 bytes	295.875 bytes
	Encryption memory usage / peak	0.181919 MB 0.272188 MB	0.121663 MB 0.211084 MB
	Encryption time	1.422537 seconds	0.008017 seconds
	Decryption memory usage / peak	0.150395 MB 0.235976 MB	0.100835 MB 0.185064 MB
	Decryption time	1.452296 seconds	0.0 seconds
	decrypted message same as encrypted message	True	True
	Throughput	61.60112729053797 Kilobytes/sec	10930.508022015716 Kilobytes/sec
	Avalanche effect	0.5003400222965441 %	0.5106870122630992 %




Table 2: Encryption results on medical images with 3 people in the group

The table 3 represents comparison results for medical data encryption/decryption with 4 people in the group.

Parameters	Elliptic curve cryptography key generation with AES	Proposed method
Key Generation time	1.049636 seconds	0.0 seconds
Total Keys length	1117.5 bytes	460.375 bytes
Encryption memory usage / peak	0.026652 MB 0.029665 MB	0.03782 MB 0.039985 MB
Encryption time	0.007984 seconds	0.008 seconds
Decryption memory usage / peak	0.012171 MB 0.013135 MB	0.013515 MB 0.014479 MB
Decryption time	0.0 seconds	0.0 seconds
Decrypted message same as encrypted message	True	True
Throughput	305.9096709043086 Kilobytes/sec	305.2978515625 Kilobytes/sec
Avalanche effect	0.5067000818330606 %	0.5058694762684124 %

Table 3: Data encryption results on medical data with 4 people in the group

The table 4 represents comparison results for medical image encryption/decryption with 4 people in the group. The algorithms applied on 4 different medical images for encryption/decryption.

Input image	Parameters	Elliptic curve cryptography key generation with AES	Proposed method
	Key Generation time	1.358042 seconds	0.0 seconds
	Total Keys length	1116.0 bytes	460.0 bytes
	Encryption memory usage / peak	0.101344 MB 0.182462 MB	0.112512 MB 0.192782 MB
	Encryption time	0.008004 seconds	0.008 seconds
	Decryption memory usage / peak	0.09034 MB 0.165859 MB	0.091684 MB 0.166763 MB
	Decryption time	0.0 seconds	0.0 seconds
	decrypted message same as encrypted message	True	True
	Throughput	9831.754044852572 Kilobytes/sec	9836.669921875 Kilobytes/sec
	Avalanche effect	0.5011505294913655 %	0.5104872810339048 %
	Image 2		
	Key Generation time	1.410706 seconds	0.0 seconds
	Total Keys length	1115.5 bytes	460.5 bytes
	Encryption memory usage / peak	0.093075 MB 0.165924 MB	0.104243 MB 0.176244 MB
	Encryption time	0.008003 seconds	0.008009 seconds
	Decryption memory usage / peak	0.082071 MB 0.149324 MB	0.083415 MB 0.150228 MB
	Decryption time	0.0 seconds	0.0 seconds
	decrypted message same as encrypted message	True	True
	Throughput	8823.96152224166 Kilobytes/sec	8817.350987951055 Kilobytes/sec
	Avalanche effect	0.5005983674598783 %	0.5109840204759269 %
	Image 3		
	Key Generation time	1.133198 seconds	0.0 seconds
	Total Keys length	1114.875 bytes	460.625 bytes
	Encryption memory usage / peak	0.069155 MB 0.118084 MB	0.080323 MB 0.128404 MB
	Encryption time	0.008267 seconds	0.007999 seconds
	Decryption memory usage / peak	0.058223 MB 0.101556 MB	0.059567 MB 0.10246 MB
	Decryption time	0.0 seconds	0.0 seconds
	decrypted message same as encrypted message	True	True
	Throughput	5716.558493105117 Kilobytes/sec	5908.087143705464 Kilobytes/sec
	Avalanche effect	0.502845223325062 %	0.5001783498759306 %
	Image 4		
	Key Generation time	1.441527 seconds	0.0 seconds
	Total Keys length	1114.75 bytes	459.5 bytes
	Encryption memory usage / peak	0.110495 MB 0.200764 MB	0.121663 MB 0.211084 MB


	Encryption time	0.007999 seconds	0.008002 seconds
	Decryption memory usage / peak	0.099491 MB 0.18416 MB	0.100835 MB 0.185064 MB
	Decryption time	0.0 seconds	0.0 seconds
	decrypted message same as encrypted message	True	True
	Throughput	10955.104739654958 Kilobytes/sec	10950.997602161959 Kilobytes/sec
	Avalanche effect	0.5001937012263099 %	0.5090919732441471 %

Table 4: Encryption results on medical images with 4 people in the group

6. Conclusion

The proposed hybrid cryptographic algorithm uses short length ECC keys and uses Diffie Helman key exchange method to broadcast an encrypted message in the group. AES algorithm is used to encrypt the data with the modified short length keys. The proposed method performed better than the conventional techniques in terms of key generation time, total keys length, encryption memory usage, encryption time, decryption memory usage peak, decryption time and Throughput.

References

- [1] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2018.
- [2] H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos, "IoT-based big data storage systems in cloud computing: perspectives and challenges," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 75–87, 2017.
- [3] A. Gatouillat, Y. Badr, B. Massot, and E. Sejdić, "Internet of medical things: a review of recent contributions dealing with cyber-physical systems in medicine," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3810–3822, 2018.
- [4] Z. Liu, Y. Cao, L. Cui, J. Song, and G. Zhao, "A benchmark database and baseline evaluation for fall detection based on wearable sensors for the internet of medical things platform," *IEEE Access*, vol. 6, pp. 51286–51296, 2018.
- [5] C. Wang, Y. Qin, H. Jin et al., "A low power cardiovascular healthcare system with cross-layer optimization from sensing patch to cloud platform," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 2, pp. 314–329, 2019.
- [6] E. Ahmed and M. H. Rehmani, "Mobile edge computing: opportunities, solutions, and challenges," *Future Generation Computer Systems*, vol. 70, pp. 59–63, 2017.
- [7] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: survey and open issues," *IEEE Access*, vol. 6, pp. 18209–18237, 2018.
- [8] D. Koo and J. Hur, "Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing," *Future Generation Computer Systems*, vol. 78, pp. 739–752, 2018.
- [9] J. Zhang, J. Cui, H. Zhong, Z. Chen, and L. Liu, "PACRT: chinese remainder theorem based conditional privacy-preserving authentication scheme in vehicular ad-hoc networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 2019, 2019.
- [10] J. Cui, L. Wei, J. Zhang, Y. Xu, and H. Zhong, "An efficient message-authentication scheme based on edge computing for vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, 2019.
- [11] Z. Guan, Y. Zhang, L. Wu et al., "APPA: an anonymous and privacy preserving data aggregation scheme for fog-enhanced IoT," *Journal of Network and Computer Applications*, vol. 125, pp. 82–92, 2019.
- [12] S. Garg, A. Singh, S. Batra, N. Kumar, and L. T. Yang, "UAV empowered edge computing environment for cyber-threat detection in smart vehicles," *IEEE Network*, vol. 32, no. 3, pp. 42–51, 2018.
- [13] X. Li, S. Liu, F. Wu, S. Kumari, and J. J. P. C. Rodrigues, "Privacy preserving data aggregation scheme for mobile edge computing assisted IoT applications," *IEEE Internet of Things Journal*, vol. 34, pp. 1–9, 2019.
- [14] F. Wu, X. Li, L. Xu, A. K. Sangaiah, and J. J. P. C. Rodrigues, "Authentication protocol for distributed cloud computing: an explanation of the security situations for internet-of-things-enabled devices," *IEEE Consumer Electronics Magazine*, vol. 7, no. 6, pp. 38–44, 2018.
- [15] H. Wang, Z. Wang, and J. Domingo-Ferrer, "Anonymous and secure aggregation scheme in fog-based public cloud computing," *Future Generation Computer Systems*, vol. 78, no. 2, pp. 712–719, 2018.
- [16] Z. Wang, "An identity-based data aggregation protocol for the smart grid," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2428–2435, 2017.
- [17] M. Alkharji, H. Liu, and M. Al Hammoshi, "A comprehensive study of fully homomorphic encryption schemes," *International Journal of Advanced Computer Technology*, vol. 10, no. 1, pp. 1–24, 2018.
- [18] B. D. Deebak, F. Al-Turjman, M. Aloqaily, and O. Alfandi, "An authentic-based privacy preservation protocol for smart e-healthcare systems in IoT," *IEEE Access*, vol. 7, pp. 135632–135649, 2019.
- [19] I. S. Farahat, A. S. Tolba, M. Elhoseny, and W. Eladrosy, "A secure real-time internet of medical smart things (IOMST)," *Computers & Electrical Engineering*, vol. 72, pp. 455–467, 2018.
- [20] S. Pirbhulal, O. W. Samuel, W. Wu, A. K. Sangaiah, and G. Li, "A joint resource-aware and medical data security framework for wearable healthcare systems," *Future Generation Computer Systems*, vol. 95, pp. 382–391, 2019.
- [21] S. Pirbhulal, O. W. Samuel, W. Wu, A. K. Sangaiah, and G. Li, "A joint resource-aware and medical data security framework for wearable healthcare systems," *Future Generation Computer Systems*, vol. 95, pp. 382–391, 2019.
- [22] Adarsh T, Kalpally and K. P. Vijayakumar. "Privacy and security framework for health care systems in IoT: originating at architecture through application." *Journal of Ambient Intelligence and Humanized Computing* (2021): 1-11.
- [23] Kedir Mamo, Beshir, Zareen Subah, and Mohammed Zamshed Ali. "IoT Sensor Initiated Healthcare Data Security." *IEEE Sensors Journal* (2020).

Authors Profile



Phani Sridhar Addepalli received M.Tech (CSE) from Gitam University and currently pursuing Ph.D from Gitam University. He is working as Associate Professor in Department of Computer Science and Engineering, Aditya Engineering College (A). He has a teaching experience of around 11 years. He dealt various subjects and handled projects in both Under graduate and Graduate level. His research interests include Internet of Things, IoT security, Healthcare 4.0, Computer Organization, Embedded Design. He has publications in both National and International Journals and Conferences.



Dr.P. Venkata Lakshmi Professor in Computer Science Engineering Department. Her research areas include Cryptography and network security, Machine learning, Neural nets and deep learning, Image processing. She has publications in various National and International Journals and Conferences. Her research contributions include a research project, books published and various guest lecture talks. As of now she guided seven doctorate students and currently guiding 10 scholars in various areas of research.

s