



A novel approach of asymmetric key generation in symmetric AES via ECDH

Sameer Farooq¹ · Priyanka Chawla¹

Received: 4 September 2019 / Revised: 11 February 2020

© The Society for Reliability Engineering, Quality and Operations Management (SREQOM), India and The Division of Operation and Maintenance, Lulea University of Technology, Sweden 2020

Abstract AES and ECC are considered as the best encryption algorithms, both are approved by NIST and are regarded as next generation algorithm. It is believed that they are quantum computer resistant. The compact ciphertext size of ECC and AES makes them the best choice for cloud computing, sensor networks or small devices where we want resource-saving without compromising the security of the system. But the main issue with AES is key sharing, which is dependent on the secured channel and if that channel gets compromised the communication will also get compromised. To overcome this issue, we are proposing the novel concept, which is generating keys for both AES and ECC using the proposed methodology given in the paper. The algorithm does not need a secured channel at any level as it is inheriting properties of the ECDH algorithm and adding novel concepts with it for key generation. We have observed from the experimental results that our proposed algorithm generates 16-byte key which is inheriting ECC property, so it can with stand with prime factorization attack and QC. The other property is that the key generation algorithm is lesser complex as compared to the existing related algorithms.

Keywords Cloud computing · Fog · IoT · ECDH · AES · ECC · NGE · QCR

Abbreviations

ECDH	Elliptic curve Diffie Hellman
ECC	Elliptic curve cryptography
NGE	Next-generation algorithm
QCR	Quantum computing resistant
DH	Diffie-Hellman

1 Introduction

This work is part of a novel cloud security framework proposed by the author under the name of “Novel Hybrid Encryption Framework to Secure Data Transmission between IoT and Fog systems”. As we know the AES and ECC both are the best encryption algorithm that we have today in the list of symmetric and asymmetric algorithms respectively. The less complexity of the algorithms in terms of the algorithm code and ciphertext makes them the most attractive choice among the cryptographers. In the various fields of IT viz, Wireless sensor networks, wireless body area networks, Cloud computing, Fog computing, etc. where we want the better security of the data with limited resources. The security administrators prefer small compact sized algorithms like ECC and AES which can use fewer resources and provide better security. The resources may include CPU execution time, battery usage, storage memory, network bandwidth, etc. Although the AES algorithm is the best algorithm, still it is not used widely by the security administrators. The reason is its need, of the secured channel for key sharing which creates an issue. As we know in the case of ad-hoc

✉ Priyanka Chawla
priyanka.22046@lpu.co.in

Sameer Farooq
sameerfarooq.lpu@gmail.com

¹ Lovely Professional University, Phagwara, India

networks the broadcasting nature of the packets and decentralized infrastructure makes everything insecure which is sent over the channel. To overcome this issue, we are proposing the new technique of key generation for ECC-128-bit and AES-128-bit algorithms by using a mixed Elliptic curve approach with old Diffie-Hellman and users customized ECDH-AES.

2 Literature review

The proposed algorithm is mainly meant for the novel cloud framework proposed by the author, namely “Novel Hybrid Encryption Framework to Secure Data Transmission between IoT and Fog systems” for encryption of data generated by the IoT devices. But this isn’t limit of the algorithm, we can use it anywhere when we are using AES or ECC algorithms even individually. In this paper a systematic literature review was performed on Cloud computing, Fog Computing and IoT devices like sensors, about their security issues and challenges” and it was found some work has been done earlier related on these fields related to the security issues and challenges but in every topic and in every work, there are pros and cons. So, in this section, we are going to discuss some limitations of these proposed systems and proposed frameworks systematically.

Subasree and Sakthivel (2010) proposed a security algorithm in which the data is enciphered with ECC and the hash value of the plain text is also generated using MD5. After that, both are encrypted again with Dual RSA and then, sent to the destination. The limitation of the algorithm is slow speed due to complexity, the huge size of output generated and the requirement of a secured channel to share private key So, it is not suitable for wireless ad-hoc networks.

In 2011, Dubal et al. (2011) proposed a security algorithm in which the data is encrypted with Dual RSA and the key generated via ECDH. The hash value of ciphertext is generated along with the ECDSA digital signature for more authentication. The whole set is sent via the secured channel to the destination site where the reverse process is done. The limitation of the algorithm is its complexity in structure, slow speed, and resource consumption. Thus, it isn’t suitable for ad-hoc networks or where resources are less and costly.

Rehman et al. (2012) studied different types of security algorithms. The did a comparative analysis of few important algorithms based on their performance and security in different areas. Their analysis work was investigating various cryptographic techniques likes’ asymmetric key symmetric key cryptography and cryptography. Not only this but it also compares different techniques for encryption

like block cipher (viz, RC2, RC5, RC6), stream cipher (viz, RC4), and hashing methods (viz, SHA, SHA1, MD, MD4, MD2).

Kumar (2012) proposed hybrid algorithm architecture for message encryption. The architecture proposes a double encryption framework for more security without considering complexity and resource consumption. The framework encrypts the data first with the AES algorithm and then sequentially the generated output with the ECC algorithm. The hash value of ciphertext is generated and then sent along with ciphertext to destination. Though it is secure due to double encryption the problem is of huge complexity, the requirement of a secured channel for AES key sharing, speed of the algorithm, etc.

Thomas and Janardhanan (2012), proposed the framework in which they were using proxy servers as a third-party server between the cloud user and the cloud server. The aim of placing the proxy server was to detect and prevent the Dos and DDos. the server was preventing the attacker from sending the sham information to the main server of a cloud. if the third-party proxy server had detected any interloper or sham packets it was not allowing them to send it to the main server. so, the proxy server was working as IDPs.

Bhadauria et al. (2012,) presented the study of various issues and threats that effects cloud service users. The author examined the various models of cloud computing like private and public clouds and the security loopholes in them. The examination of the study focuses on the data stockpiling security and the threats related to them the information stockpiling is classified into NAS, SAN, MCC.

Kahanwal and Singh (2013) proposed the detailed study of the pros and cons of cloud computing along with the various security issues in it. The author examined cloud technology in contrast with other technologies like distributed computing, P2P computing, etc. At last, the author draws various conclusions regarding the advantages and disadvantages of cloud computing in comparison to other technologies.

Hashizume et al. (2013) studied the different types of security issues that surround the SPI model in the cloud computing environment. The author identified the various security threats that are present in the cloud computing along with them he studied their effects. He also studied the vulnerabilities of the cloud computing domain SPI model and studied the relationship between them along with their dangers and countermeasures. the author, at last, presents some finds and draws some conclusions regarding that.

Coetzee and Eksteen (2011) proposed the advancement of the utilization of the internet starting from computers to humans and now finally to things, allowing

the development of many new applications and services. Some application areas have also been highlighted. This paper also discusses some challenges that are being faced by the Internet of Things at present and needs to be addressed. The challenges that have been discussed include- privacy, standardization, interoperability, etc. It also highlights the efforts made by some multinational companies like IBM, Microsoft, etc. in recognizing the commercial potential of the internet of things.

Jing et al. (2014) proposed work that has considered the three levels of architecture of the internet of things comprising of perception, transport and application layer and has focused on the security issues of each of these layers. The various issues discussed are as follows-

1. Perception layer-
 - (a) Security issues of RFID technology-uniform coding, conflict collision, privacy protection, etc.
 - (b) Security issues of wireless sensor networks.
 - (c) Problems of heterogeneous integration.
2. Transport layer-
 - (a) Wi-Fi security analysis.
 - (b) Ad hoc security analysis issue.
 - (c) Core network.
3. Application layer-
 - (a) Threats and attack issues.
 - (b) interruption of service.
 - (c) Investigate audit issues.

This paper also discusses the security issues of various applications of the internet of things such as smart homes, smart cities, etc. It also compares the issues of security between traditional networks and the internet of things and finally concludes that the internet of things resides in a more dangerous surrounding environment having very limited resources, thus lightweight solutions should be used for the security of the internet of things.

Madakam et al. (2015) proposed, 'Internet of Things (IoT): A Literature Review' - This paper gives an idea about the different types of architectures of IoT. The main problem of IoT is that it is so vast that there is no uniform architecture has been proposed for it. This paper also gives details of some of the proposed architectures for IoT. Some of the architectures discussed here are:

1. European FP7 Research Project.
2. ITU Architecture: A five-layered architecture.
3. IoT Forum Architecture: It divides IoT into three parts.

Rizk and Alkady (2015), proposed the security framework for wireless sensor networks. In their

architecture, they are using AES, ECC and Dual RSA for key sharing, encryption, and decryption. The data set is divided into two halves the first part is encrypted with AES and second with Dual RSA. The ECC is used for key sharing purposes. The MD5 is used for hashing. The algorithm is good in confusion due to hybrid nature, but when it comes to limitation the problem is in key sharing and vulnerability of RSA with prime factorization attack.

Adnan and Ariffin (2018) proposed a 3-D AES algorithm to secure the data in cloud storage for big data applications. The 3-D AES or key alternating block cipher is composed of rotation key function that has 3 iterations of round function, each with a key mixing operation. The algorithm is good as it sends encrypted data to the server but the issue is when sharing it is private key with the server-side, which has been used for encryption and is a must for decryption on the server-side. Another vulnerability is of an algorithm is, the data is decrypted on the cloud server-side and then it is sent to the respective clients. So, it can be easily attacked in the middle of transmission via server to a client.

3 Proposed methodology

- *Method to generate the secured key for AES and ECC algorithm using ECDH and ECDH-AES algorithms*

Normally in the case of the AES algorithm we generate the random key on the sender side then we send that key to the destination site via a secured channel. The key that we send isn't encrypted it is normally a plain text key. Even if we encrypt it, still we have to send the encryption key of the particular algorithm that encrypted this key to the destination for decryption of the actual key. That second key will still be in plain text form, again the same issue. So, the idea is instead of generating random key and sharing it with destination side via secured channel, which may not be secured, the best idea is to generate key on both sides of sender and destination and use those keys for encryption and decryption of data, without exchanging those keys over network, which is susceptible to attacks.

The motive of this work is to generate the 16-byte key for the AES algorithm via the Elliptic curve and Diffie Hellman techniques. How to generate that key the procedure is given in the paper and it is also simulated via MATLAB along with results. ECDH stands for the elliptic curve Diffie-Hellman algorithm. The algorithm uses elliptic-curve features in simple Diffie-Hellman algorithm, so as to make it complex and hard for prime factorization attack. The Diffie-Hellman is used to generate the key on

Table 1 ECDH domain parameters

P	Field (modulo p)
a, b	Curve parameters
G	Generator (base point)
N	Order(G)
H	Co-factor

both sides, that generated key is later used for encryption and decryption of data. Domain parameters for ECDH are given in Table 1.

All of these domain parameters are public.

Elliptic curve equation is:

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

After generating the ECDH key successfully on both sides, IoT device and base station, the generated key is used in ECC-128-bit and AES-128 bit for encryption and decryption process. For the AES algorithm, the generated key on both sides of the receiver and destination is converted into another key of 16-bytes that fits for the AES algorithm. So, the dependency of the AES algorithm over the secured channel is removed. The generated key doesn't need to be encrypted and there are also no issues of MITM attack or if a key gets exposed somewhere in middle in order to generate that key on both sides of IoT and fog system another ECDH-AES key generation algorithm is executed on both sides.

3.1 ECDH method for generation of the common key on sender and receiver side

At the sink Node:

- Choose random private key α ($1 < \alpha < n - 1$).
 - Now multiply generator point G with private key α , so $S = \alpha G$.
 - Exchange results to the destination side. "There is no need for secure channel".
 - Calculate Final Key, K by multiplying received values from another side with a private key
- $$K = \alpha \cdot \beta \cdot G.$$

At base station Node:

- Choose random private key β ($1 < \beta < n - 1$).
- Now multiply generator point G with private key β , so $B = \beta G$.

- Exchange results to the destination side. "There is no need for secure channel".
 - Calculate Final Key, K by multiplying received values from another side with a private key
- $$K = \alpha \cdot \beta \cdot G.$$

The final key K generated on both sides will consist of x and y coordinates. So,

$$K = K(x, y)$$

In the above procedure sink node refers to the sender end who wants to send data and the base node refers to the receiving end. The sink node is a generator of data who sends data to another end and the base station is the receiving end. Mostly sink node includes an IoT device, and the base station consists of a fog system or a cloud server.

3.2 ECDH method for generation of 16-byte AES key on sender and receiver side

After successfully generating key $K(x, y)$ on both sides of the sender and destination end, we will use that key to generate a 16-byte AES key. In order to generate that key on both sides of the sink node and base node, the steps are given below are going to be followed:

- First the values of x, y will be rounded to the nearest integers, so as to convert these values into integer form if they are in real form.
- The second step will be, if the numbers are less than 1 then add prime number 3 if it is x otherwise add prime number 5 if it is y .

If $[x \leq 1]$ then

$$x = x + 3$$

If $[y \leq 1]$ then,

$$y = y + 5$$

- The third step will include the squaring of both x, y .
- Now the 16-byte key will be calculated by squaring and cubing values of x and y alternatively along with calculating mode of each value.
- The next values will be the alternative squares and cubes of previous values respectively.

$$\begin{aligned}
A1 &= X^2 \bmod 99. \\
A2 &= Y^3 \bmod 99. \\
A3 &= (A1 * X^2) \bmod 99. \\
A4 &= (A2 * Y^3) \bmod 99. \\
A5 &= (A3 * X^2) \bmod 99. \\
A6 &= (A4 * Y^3) \bmod 99. \\
A7 &= (A5 * X^2) \bmod 99. \\
A8 &= (A6 * Y^3) \bmod 99. \\
A9 &= (A7 * X^2) \bmod 99. \\
A10 &= (A8 * Y^3) \bmod 99. \\
A11 &= (A9 * X^2) \bmod 99. \\
A12 &= (A10 * Y^3) \bmod 99. \\
A13 &= (A11 * X^2) \bmod 99. \\
A14 &= (A12 * Y^3) \bmod 99. \\
A15 &= (A13 * X^2) \bmod 99. \\
A16 &= (A4 * Y^3) \bmod 99.
\end{aligned}$$

- (f) These values are first converted into integer values by removing decimal part from **A1–A16** and after that they will be converted into the hexadecimal equivalents so 16-byte key to be used for AES-128 is below (Table 2).

3.3 Working example of algorithms

The algorithm generates key stepwise on both sides. Let us consider common domain parameters are:

Generator point: $G_{(x, y)}$

$$x = 4, y = 3$$

Curve parameters: $x^2 = y^3 + ax + b$

$$a = 2, b = 1$$

Step 1: At Sink node (IoT device end):

1. Choose private key = 5
2. Generate public key $\beta = G_{(x, y)} * 5$
i - e $\beta = G_{(4,3)} * 5$

Table 2 Generated ECDH-AES key

A1	A2	A3	A4
A5	A6	A7	A8
A9	A10	A11	A12
A13	A14	A15	A16

After calculating it using ECC point doubling method

$$\beta_{(x, y)} = \beta_{(9.684, -14.613)}$$

Now send this public key to another end and also do ECC multiplication (point doubling) between the received key public key of another side and own private key to get final key 'K'

$$K = \beta_{(x, y)} * \alpha$$

i-e

$$K = \beta_{(9.684, -14.613)} * 5$$

After calculating it Final Key K

$$K_{(x, y)} = K_{(3.226, 3.300)}$$

Step 2: At Base station (Fog device end):

1. Choose private key = 7
2. Generate public key $\alpha = G_{(x, y)} * 7$
i - e $\alpha = G_{(4,3)} * 7$

After calculating it using ECC point doubling method

$$\alpha_{(x, y)} = \alpha_{(6.0092, 12.538)}$$

Now send this public key to another end and also do ECC multiplication (point doubling) between the received key public key of another side and own private key to get final key 'K'

$$K = \alpha_{(x, y)} * \beta$$

i-e

$$K = \alpha_{(6.0092, 12.538)} * 7$$

After calculating it Final Key K

$$K_{(x, y)} = K_{(3.226, 3.300)}$$

Step 3: Generating 16-byte key for AES:

Now we have final key ' $K_{(x, y)}$ ' on both sides of the IoT and Fog system, the third step is to generate the 16-byte AES. This procedure will be followed on both sides of sink node (IoT device) and base station (Fog system)

Let **A1, A2, A3... A16** are the 16-bytes of AES key.

- (a) First check for $K_{(x, y)}$ is $x \leq 1$ if no then:

$$x = x + 3$$

- (b) Now, check for $K_{(x, y)}$ is $y \leq 1$ if no then:

$$y = y + 5$$

Since, $K_{(3.226, 3.300)}$ thus $x = 3.226$ and $y = 3.300$ so no need to perform above checks.

(c) Now $x = x^2$ and $y = y^2$

So, in our case

$$x = (3.226)^2 = 10.41$$

$$y = (3.300)^2 = 10.89$$

Now,

$$\begin{aligned} A1 &= X^2 \bmod 99 \\ &= (10.41)^2 \bmod 99 \\ &= 9.36 \end{aligned}$$

$$\begin{aligned} A2 &= Y^3 \bmod 99 \\ &= (10.89)^2 \bmod 99 \\ &= 4.47 \end{aligned}$$

$$\begin{aligned} A3 &= (A1 * X^2) \bmod 99 \\ &= 9.36 * (10.41)^2 \bmod 99 \\ &= 24 \end{aligned}$$

$$\begin{aligned} A4 &= (A2 * Y^3) \bmod 99 \\ &= 4.47 * (10.89)^2 \bmod 99 \\ &= 30.86 \end{aligned}$$

$$\begin{aligned} A5 &= (A3 * X^2) \bmod 99 \\ &= 24 * (10.41)^2 \bmod 99 \\ &= 26.64 \end{aligned}$$

$$\begin{aligned} A6 &= (A4 * Y^3) \bmod 99 \\ &= 30.86 * (10.89)^2 \bmod 99 \\ &= 56.76 \end{aligned}$$

$$\begin{aligned} A7 &= (A5 * X^2) \bmod 99 \\ &= 26.64 * (10.41)^2 \bmod 99 \\ &= 15.7 \end{aligned}$$

$$\begin{aligned} A8 &= (A6 * Y^3) \bmod 99 \\ &= 56.76 * (10.89)^2 \bmod 99 \\ &= 43.84 \end{aligned}$$

$$\begin{aligned} A9 &= (A7 * X^2) \bmod 99 \\ &= 15.7 * (10.41)^2 \bmod 99 \\ &= 18.25 \end{aligned}$$

$$\begin{aligned} A10 &= (A8 * Y^3) \bmod 99 \\ &= 43.84 * (10.89)^2 \bmod 99 \\ &= 89 \end{aligned}$$

$$\begin{aligned} A11 &= (A9 * X^2) \bmod 99 \\ &= 18.25 * (10.41)^2 \bmod 99 \\ &= 96.57 \end{aligned}$$

$$\begin{aligned} A12 &= (A10 * Y^3) \bmod 99 \\ &= 89 * (10.89)^2 \bmod 99 \\ &= 81 \end{aligned}$$

$$\begin{aligned} A13 &= (A11 * X^2) \bmod 99 \\ &= 96.57 * (10.41)^2 \bmod 99 \\ &= 69.32 \end{aligned}$$

$$\begin{aligned} A14 &= (A12 * Y^3) \bmod 99 \\ &= 81 * (10.89)^2 \bmod 99 \\ &= 51 \end{aligned}$$

$$\begin{aligned} A15 &= (A13 * X^2) \bmod 99 \\ &= 69.32 * (10.41)^2 \bmod 99 \\ &= 86.51 \end{aligned}$$

$$\begin{aligned} A16 &= (A14 * Y^3) \bmod 99 \\ &= 51 * (10.89)^2 \bmod 99 \\ &= 29.97 \end{aligned}$$

The final 16-byte key will be obtained by rounding all the obtained values to nearest integer:

9	4	24	31
27	57	16	44
18	89	97	81
69	51	87	30

4 Results and discussion

The proposed key generation algorithm was implemented in MATLAB 2012a and Visual Studio 2013. The reason behind using MATLAB for a key generation was actually to measure the time that the algorithm takes to generate the

key. Though we are working with IoT and cloud so measuring time for key generation is very important. In our case sink node is actually IoT device and the base station is Fog Server located at the edge of LAN. So, measuring time is very important because communication between IoT and Fog is time-sensitive as well as resource-sensitive. In MATLAB we can call external libraries as per our requirement. MATLAB permits to test calculations and algorithms promptly without recompilation. In MATLAB once we type anything in the execution section or command-line we quickly observe the outcomes, incredibly encouraging calculation advancement and code development. MATLAB environment permits us to work intelligently with our information, encourages us to monitor records and factors, and improves regular programming/investigating assignments. The whole outcome is the successful generation of the key on both sides of the sink and base station with help of Visual Studio 2013 and MATLAB 2012a. It also includes the time taken for ECDH key generation and ECDH-AES key generation.

The following section shows the results obtained with their explanation.

Figure 1, shows the ECDH key generation process between the sink node and base station. Here the common domain parameters are selected first on both ends. After that, the sink node selects his private key to generate an intermediate key by using the elliptic-curve point doubling method. Similarly, the base station selects his private key

to generate an intermediate key using the same point doubling method. Both ends exchange their intermediate key results with one another in order to generate the final key. The final key on both sides is computed but multiplying the received input key sent from another end with their own private key.

Figure 2, shows the MATLAB implementation of the ECDH algorithm and shows how keys are generated practically using MATLAB.

Figure 3, shows the ECDH-AES key generation process between the sink node and base station. AES-128 16-bytes key is generated from the existing ECDH key. The ECDH key's $[K(x, y)]$ coordinates x and y coordinates will be used to generate a 16-byte AES key. The process of how this key is generated is explained in the ECDH key generation phase of the proposed system.

Figure 4, shows the MATLAB implementation of the ECDH-AES algorithm and shows how 16-byte key for the AES-128 algorithm is generated practically using MATLAB.

Figure 5, shows the 16-byte hexadecimal equivalents of the previously generated key using MATLAB implementation.

Figure 6, shows the ECC and ECDH-AES key generation time, using MATLAB. The figure shows durations of time taken by these key generation algorithms, while the plain text was of different sizes. In the graph, it is clear the time taken by ECC-AES key generation is relatively higher

Fig. 1 ECDH key generation

ECDH KEY GENERATION & EXCHANGE

Common Domain Parameters:

Base Point :

X_g : Y_g :

Curve Parameters : $y^2 = x^3 + ax + b$

a : b :

Base Station

Enter the Private Key:

Public Key: (6.00920649865811, 12.5385095397795)

Base Station Final Key: (3.22585875803946, 3.30095408284525)

Sink Node

Enter the Private Key:

Public Key: (9.68362980001075, 34.6127224850485)

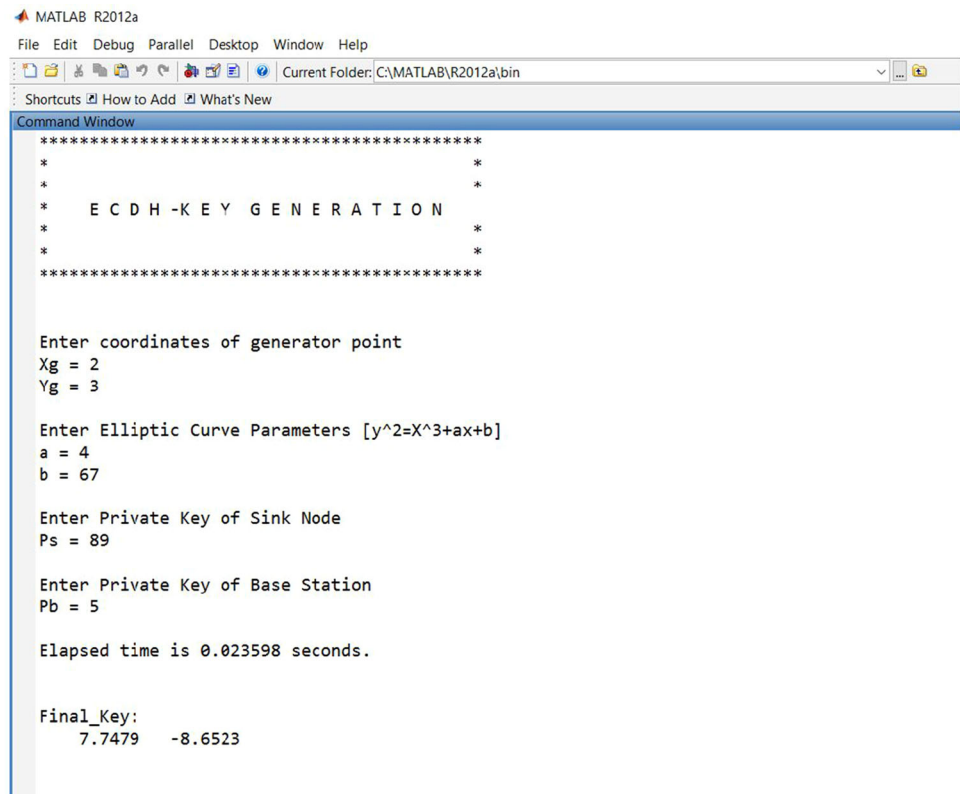
Sink Node Final Key: (3.22585875803946, 3.30095408284525)

Base Station

Enter the Private Key:

Public Key: (6.00920649865811, 12.5385095397795)

Base Station Final Key: (3.22585875803946, 3.30095408284525)

Fig. 2 MATLAB ECDH key generation


```

MATLAB R2012a
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\MATLAB\R2012a\bin
Shortcuts How to Add What's New

Command Window

*****
*                                     *
*   E C D H - K E Y   G E N E R A T I O N   *
*                                     *
*****

Enter coordinates of generator point
Xg = 2
Yg = 3

Enter Elliptic Curve Parameters [y^2=X^3+ax+b]
a = 4
b = 67

Enter Private Key of Sink Node
Ps = 89

Enter Private Key of Base Station
Pb = 5

Elapsed time is 0.023598 seconds.

Final_Key:
    7.7479    -8.6523

```

ECHD KEY GENERATION & EXCHANGE

AES 16-byte key generation from final ECDH Key:

Fetch Generated Public Key :

(3.22585875803946 3.30095408284525)

3	3	9	27
45	45	9	27
45	45	9	27
45	45	9	27

Fig. 3 ECDH-AES key generation

as compared to the simple ECDH key generation algorithm. Since we need both for our algorithm so both have equal priority.

4.1 Comparison with existing algorithms

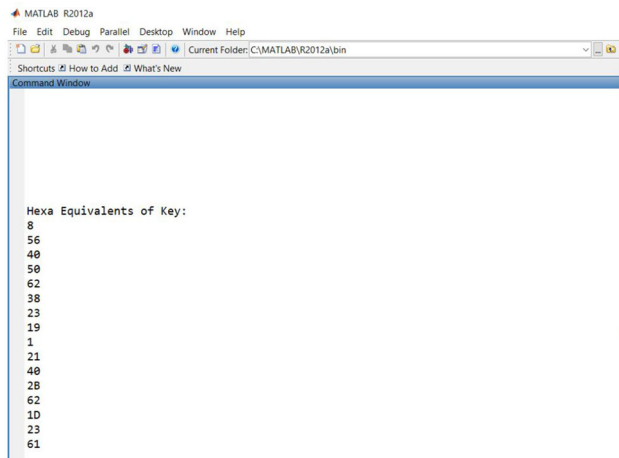
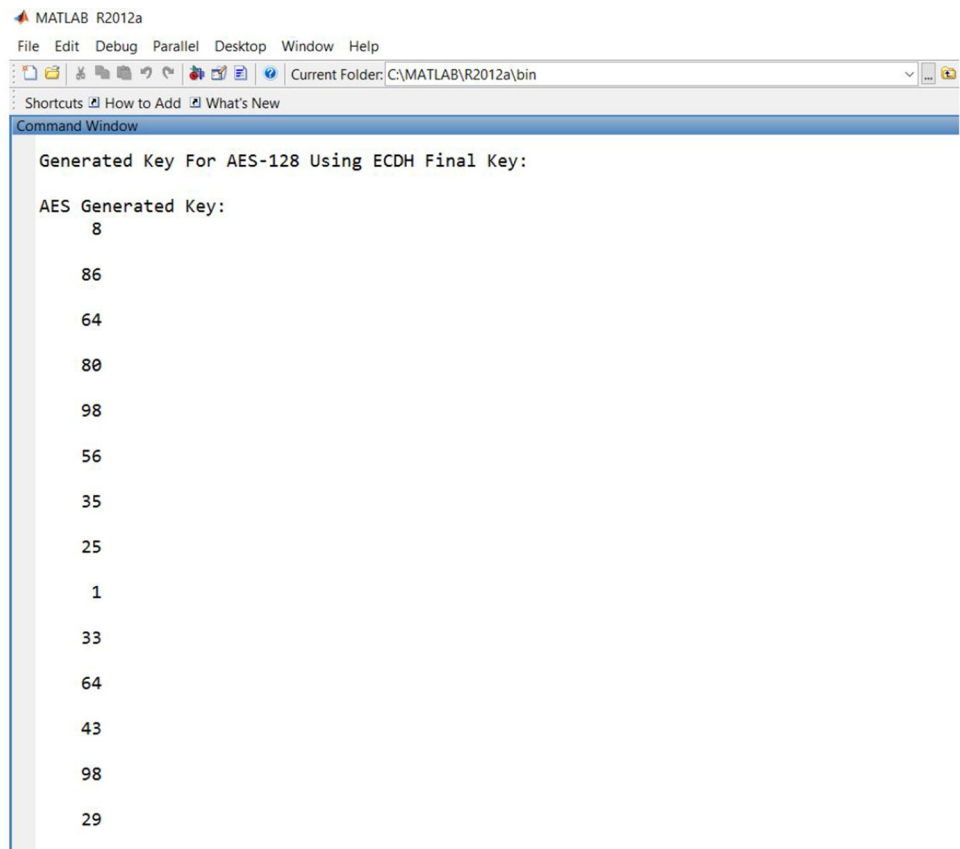
Table 3 given below lists all the related key generation algorithms. It includes different versions of Diffie-Hellman that are of two types. One is a simpler version another one is inheriting discrete elliptic curves. The simple versions of DH differ due to their key sizes which range from 786-3072 bits. The assumption is higher the key size the more it is secure (higher key sizes can withstand with the

prime factorization attack more). But when it comes to QCR or NGE none of them is so, they are vulnerable to quantum calculations and cannot be used in the future.

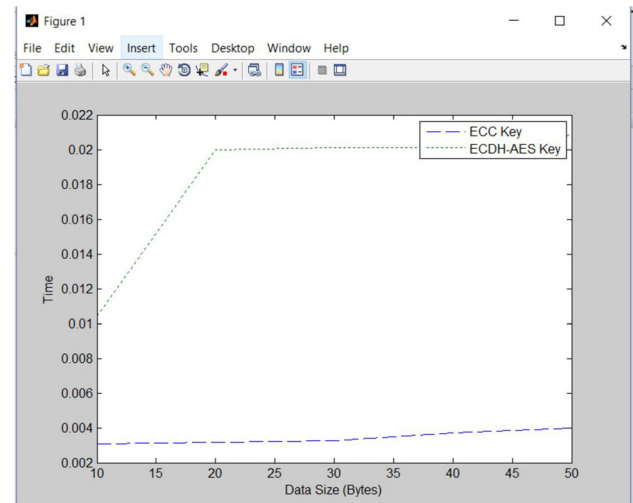
Another version of DH is based on elliptic curve cryptography. As it is inheriting discrete properties elliptic curves, so it can withstand with quantum computing calculations and thus prime factorization attack. ECDH version of DH has two versions here one is using a key size of 256-bits the other is using a 384-bit key. Both versions are QCR and NGE but the problem is increasing complexity of algorithms with larger key size. The algorithm that we are proposing generates the 128-bit key and also inherits discrete properties of elliptic curves. Thus, it balances between complexity and security (as ECC isn't vulnerable to prime factorization attack or MITM). And we can also assume the algorithm is NGE and QCR (Inheriting ECC properties).

5 Conclusion

The work we have done shows how we can overcome the requirement of a secured channel for AES using a proposed scheme. As we know AES is a symmetric key algorithm and requires the same key for encryption and decryption. The key used is secret and needs to be kept confidential.

Fig. 4 MATLAB ECDH-AES key generation**Fig. 5** MATLAB 16-byte ECDH-AES key

So, when sending this key from the sender side to the receiver side, one needs to send this key via a secured channel. To get an appropriately secured channel over the internet is not possible. Here comes the use of the proposed scheme, with that we can generate a key for the ECC algorithm as well as for the AES algorithm and we don't need any secured channel. The proposed scheme inherits the properties of Diffie-Hellman and ECC and we know the

**Fig. 6** ECC and AES key generation time

elliptic curve combination with Diffie-Hellman makes the algorithm QCR as ECC (CISCO).

The motive of the proposed algorithm is to overcome the limitation of the secured channel need in case of symmetric algorithms. Once the barrier of secure channel requirement will overcome in symmetric and asymmetric algorithms like ECC and AES we can use them in any field

Table 3 Comparison table of key generating algorithms

Algorithms	Functionality	Status	Key size	Issues	QCR	NGE	Summary
DH-768, -1024	Key exchange	Avoid	768–1024	Key-size	No	No	Less complexity but susceptible to prime factorization attack
DH-2048	Key exchange	Acceptable	2048–2048	Key-size	No	No	Less complexity but susceptible to prime factorization attack
DH-3072	Key exchange	Acceptable	3072	Key-size	No	No	Less complexity but susceptible to prime factorization attack
ECDH-256	Key exchange	Acceptable	256	Key-size/complexity	Yes	Yes	More complexity but not susceptible to prime factorization attack (ECC property)
ECDH-384	Key exchange	Acceptable	384	Key-size/complexity	Yes	Yes	More complexity but not susceptible to prime factorization attack (ECC property)
ECDH-AES	Key exchange	Acceptable	128	–	Yes	Yes	Balanced complexity (key size 128) but not susceptible to prime factorization attack (ECC property)

where we may need the encryption. The proposed key generation algorithm will work anywhere whether it is WSN, WBAN, Cloud Computing Fog or IoT. The best in an algorithm is its elliptic curve characteristics appended with that of Diffie-Hellman, this make keys generated and exchanged securely from prime factorization or third party MITM attack. The duration of key generation and its compact bit size makes it good for the resource critical environments.

References

- Adnan NAN, Ariffin S (2018) Big data security in the web-based cloud storage system using 3D-AES block cipher cryptography algorithm. In: International conference on soft computing in data science. Springer, Singapore, pp 309–321
- Bhadauria R, Chaki R, Chaki N, Sanyal S (2011). A survey on security issues in cloud computing. pp 1–15. arXiv preprint [arXiv:1109.5388](https://arxiv.org/abs/1109.5388)
- Coetzee L, Eksteen J (2011) The internet of things-promise for the future? An introduction. IST-Africa Conference Proceedings, 2011. IEEE, 2011
- Dubal MJ, Mahesh TR, Ghosh PA (2011) Design of a new security protocol using hybrid cryptography architecture. In: Proceedings of 3rd International Conference on Electronics Computer Technology (ICECT), vol 5, India
- Hashizume K, Rosado DG, Fernández-Medina E, Fernandez EB (2013) An analysis of security issues for cloud computing. J Internet Serv Appl 4(1):5
- Jing Q, Vasilakos AV, Wan J, Lu J, Qiu D (2014) Security of the internet of things: perspectives and challenges. Wirel Netw 20(8):2481–2501
- Kahanwal D, Singh DT (2013) The distributed computing paradigms: P2P, grid, cluster, cloud, and jungle. arXiv preprint [arXiv:1311.3070](https://arxiv.org/abs/1311.3070)
- Kumar N (2012) A secure communication wireless sensor networks through hybrid (AES+ECC) algorithm, vol 386. von LAP LAMBERT Academic Publishing
- Madakam S, Ramaswamy R, Tripathi S (2015) Internet of things (IoT): a literature review. J Comput Commun 3(5):164
- Rehman SU, Bilal M, Ahmad B, Yahya KM, Ullah A, Rehman OU (2012) Comparison based analysis of different cryptographic and encryption techniques using Message Authentication Code (MAC) in wireless sensor networks (WSN). Int J Comput Sci Issues (IJCSI) 9(1):96–101
- Rizk R, Alkady Y (2015) Two-phase hybrid cryptography algorithm for wireless sensor networks. J Electr Syst Inf Technol 2(3):296–313
- Subasree S, Sakthivel NK (2010) Design of a new security protocol using hybrid cryptography algorithms. IJRRAS 2(2):95–103
- Thomas G, Janardhanan PS (2012) Intrusion tolerance: enhancement of safety in cloud computing. Int J Adv Res Comput Commun Eng 1(4)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.