Novel Hybrid Technique Enhancing Data Privacy and Security

Y. Sunil Raj, S. Albert Rabara, A. Arun Gnanaraj, S. Kumar

Novel Hybrid Technique Enhancing Data Privacy and Security

Mr. Sunil Raj Y, Dr. S. Albert Rabara, Dr. A. Arun Gnanaraj, Dr. S. Brito Ramesh Kumar

Research Scholar, Dept. of Comp. Sci, St. Joseph's College (Autonomous), Affiliated to Bharathidasan University, Trichy - 2, India.

Associate Professor, Dept. of Comp. Sci, St. Joseph's College (Autonomous),Affiliated to Bharathidasan University, Trichy-2,India.

Software Project Manager and Solution Architect, Qanawat L. L. C, Dubai Media City - 502299, Soudi Arabia.

Assistant Professor, Dept. of Comp. Sci, St. Joseph's College (Autonomous), Affiliated to Bharathidasan University, Trichy –2, India.

ysrsjccs@gmail.com, a_rabara@yahoo.com, arun.art06@gmail.com, brittork@gmail.com

ABSTRACT: Internet of Things is playing a major role by connecting everything via Internet to the Cloud. Data from things are collected at gateways and processed at the application servers. The processed data is sanitized and masked to preserve it on Cloud. The security and privacy issues such as data misuse, loss or theft make IoT Cloud more untrusted. While providing numerous benefits IoT Cloud help the business organizations grow by limiting the time and economy. Fixing the security and privacy issues would enhance the use of these technologies. This work is aimed at proposing a novel DNA based algorithm to enhance the security and privacy of data on cloud. The detailed study is conducted on the existing security schemes. Finally, clear picture on the pits and holes in existing IoT Cloud based security schemes have been presented. The proposed scheme has been tested by implementing it using opensource tools. Results generated depicts the level of security and privacy provided by the proposed scheme. This would also help researchers to analyse the efficiency of existing hybrid security schemes.

KEYWORDS: *IoT, ECC DNA, Cloud Computing, TPA, Privacy, Security*

1. INTRODUCTION

Internet of **T**hings (**IoT**) connects the devices and sensors over the Internet. Objects with intelligent sensors, and a number of connectivity protocols help connecting things together [1-2]. As the devices are of low capability, the connecting protocols include, CoAP, MQTT, AMQP, and XMPP [3-5]. Data thus transferred by this **IoT** network, will be converted to HTTP by the gateway. This data finally reaches cloud for processing, analysis or storage.

Cloud **C**omputing (**CC**) provides services in a distributed fashion despite of the geographical locations. Therefore business/ firms are saving a huge space and large amount of time. Major transactions are done using the protocols such as HTTP, POP, SMTP, FTP and so on. Most services have direct link with **IoT's**, where they send request using protocols such as CoAP/ MQTT. Working on the request Cloud would respond with its own language (HTTP). In this scenario any point of time eavesdroppers or malicious users may initiate an attack [6].

Availing lot of advantages, greatest problem in cloud is security threats [23-27]. The security and privacy issues that arise as **CC** leads to lack of manual control and implements virtualization. This may lead to integrity, consistency and privacy issues. The attacks such as spoofing, replace, DOS, eavesdropping and man in the middle attack could expose the data privacy. Privacy of data in **IoT C**loud can be achieved by handling privacy at device level, storage level, processing level and transit level [29]. Therefore, proper security mechanism is to be devised to enhance the security by strengthening the authentication and encryption techniques [7].
User authentication combined with audit mechanisms and proper encryption could enhance the security and privacy [29 - 32] to a larger extent. The encryption schemes in Hybrid form perform better. **DNA** coding-based encryption is performing better with **ECC** [8-12]. It is revealed that **DNA** is more efficient and would be predominantly used in **IoT C**loud [13]. This strengthens the security using subsequent substitution and **ECC** does the rest. For enhancing the security but reducing the computing power the **DNA** encoding mechanism stands first. As the research is on the go **DNA** may be undertaking every area. This work is intended to have a detailed review on the security mechanisms employed in **IoT C**loud and to propose a novel **DNA** based security mechanism.

The paper is organized as follows: Section (ii) presents the review of literature, Section (iii) presents the proposed algorithm, Section (iv) showing the result and analysis, Section (v) concludes the work.

2. REVIEW OF LITERATURE

The review exhibits the recent security techniques that exist to protect the IoT network. As **DNA** cryptography requires substitutions as Caesar ciphers, they reduce computational complexity and assure security. Here data is transformed to genomic sequence which resembles a **DNA** strand.

In [25] a lightweight encryption based on **DNA** sequence for **IoT** device's is proposed by the authors. Here authors have used **DNA** sequence to generate secret key, that is used to encrypt images. Hence mixing it with the classical cryptographic scheme such as symmetric or asymmetric would provide strong security [14-15, 22].

Novel methods have been proposed to enhance security of data, using Huffman with **DNA** cryptography [4, 16]. The proposal exhibits multiple level of encoding. **DNA** coding scheme include an advantage of better authentication, storage, and digital signatures. In [9] authors have tried securing the data using digital signatures and **DNA** cryptography. Here public-key algorithm is implemented in the form of **DNA** sequencing. In [10] a **DNA** based method is realized on **IoT's**. After implementation of proposal efficiency in energy consumption was detected.

[11] Presents **DNA** and **ECC** based hybrid scheme. After sequence selection, sorting is applied and then resultant is replaced with **DNA** codes. Now the data is converted into binary after which they are encrypted using **ECC**. To provide to **IoT** environment another approach by [12], uses **DNA ECC** for reducing the processing time in **IoT** devices and reducing the size of memory in **IoT** devices.

In [24] author have designed an algorithm based on composite chaos map for securing image. After XOR operation using composite chaotic map, **DNA** encoding which results in image being less prone to attacks. [28] have proposed a system to secure the data in transit with integrity and availability mechanisms. To enhance the data confidentiality a **DNA** cryptography is used along with AES.

The Table 1, accounts few recent proposals **ECC** and **DNA** based hybrid algorithm. In [17] authors have found that **DNA** and **ECC** make difference in **IoT**. The framework is employed with double layered security, first is **DNA** followed by **ECC**. On combining cryptography with steganography [18] have used SHA512, **ECC**, **DNA** and CM-CSA. To hide data behind video, initially the data is compressed using IHE. Multilevel encrypted data is put in pixel points behind frames. The steganographic process is done to enhance security of data.

On suggesting a double level of encryption authors in [19] authors have used two keys for encoding. The first key is generated using **ECC**, and Gaussian kernel (GKF). The other key being generated based on random injective mapping. In [20] authors have proposed **ECC** based hybrid technique. This provides solution for secure communication between nodes. Authors have implemented the scheme as two parts, where the first is encrypting the data with play fairs. And the second is hiding the encrypted data behind **DNA** in a random location using **ECC**.

In [26] authors have proposed an architecture to improve the level of security in **IoT C**loud. They have adopted a hybrid **DNA** based mechanism with **ECC**. Also, authors have implemented an audit mechanism to monitor data integrity.

Tabel 2.1. DNA based Hybrid Security – Existing Approaches

	Proposal	**Technique**
[11]	Framework	DNA & ECC
[12]	Framework	DNA & ECC
[17]	Framework	ECC, RSA, DNA
[18]	Algorithm	DNA, SHA512-ECC
[19]	Algorithm	DNA, ECC, DES, GKF
[20]	Algorithm	DNA, ECC
[26]	Architecture	DNA, ECC, TPA

Hybrid encryption schemes being the security experts, table 2.1, shows the way algorithms are combined for enhancing the security in **IoT C**loud. The attacks that are raised by the anonymous or malicious users are dealt with these schemes. Though the proposals are providing better security, there is no prominent approach to resist data privacy and security attacks on **IoT C**loud. Hence the proposed security mechanism.

3. PROPOSED WORK

The proposed mechanism enhances the security as it is composed of four components including, **DNA P**rocessing **C**enter (**DPC**), **S**ecurity **M**anagement (**SM**), **T**rusted **TPA** on **C**loud (**TPAC**) and Secure **DNA** Bank (**SDB**). The devices such as sensors/ actuators communicate with each other while it is under clouds boundary.

3.1 Functionality of the Proposed Architecture

User Registration and Authentication. Registration is the initial level where service user is adopted to the **IoT C**loud platform through the **DNA G**ateway (**DG**). Such users are allowed to use the services. The registration will be done with credentials including Name, DoB, MSISDN, MAC, E-mail and so on. The User ID is provided for authenticating the user later. For any user to access the service, have to be authenticated which largely increase the security and privacy of user data. The authentication is done by sending and validating the OTP.

Device Registration and Authentication. Any authenticated user allowed to register their **IoT** Device. The required credentials are IMEI, location, service, name and type. Things Manager (TM) receives the credentials and passes it Things Registry (TR). An key is generated and split into two. First few bits are sent via **DNA G**ateway and the remaining sent to users' mobile. If the authentication is successful, the key is stored. As the TR forwards the device id generated to the certificate registry (CR) it generates a Device Certificate (DC) based on ECDSA. Once the device is registered it is ready for use on successful authentication by the user. The process of device authentication is done by the verification of DC.

Data on Transit. After the successful authentication of User at DG and device authentication at TM, on verifying the service certificate the Service Registry (SR) provides the user with the service. The data is encrypted using **ECC DNA** and hash of the data is generated using SHA1. The cypher text and signature merged together is sent to the cloud through the application. The application server after extracting the encrypted text calculates the signature and compares the same with the signature received. This process at the receiving end ensures the privacy and integrity of data. After the user and application specific process the data is again encrypted to be stored at SDB.

DNA Bank/Data at Rest. The data from the Data Owner (Do) is allowed to be stored as blocks. These blocks contain encrypted data, signature of data and signature of the next block of data. As the data enters the cloud the Third Party Audit Manager (TPAm) makes an entry on the device and the user involved in the process. TPAm also records the signature. While a request is received from the Data Users (D_U), TPAm verifies the access rights and transfers the control to Third party Auditor for user (TPAu). TPAu handles the process of requesting the Do through TPAo. Only after recording the credentials such as signature along with device and user credentials the access is provided.

TPAC. Third Part Auditors monitor the access to keep the non-repudiation. The TPA_u hold the user's privileges and other related credentials while monitoring the activities of Du. The TPA_o monitor the activities of Do. Also providers are also included in the process of auditing as they are treated as an user. Thus TPAC by verifying TPAs and CSP's could enhance the privacy and integrity of data on **IoT C**loud environment.

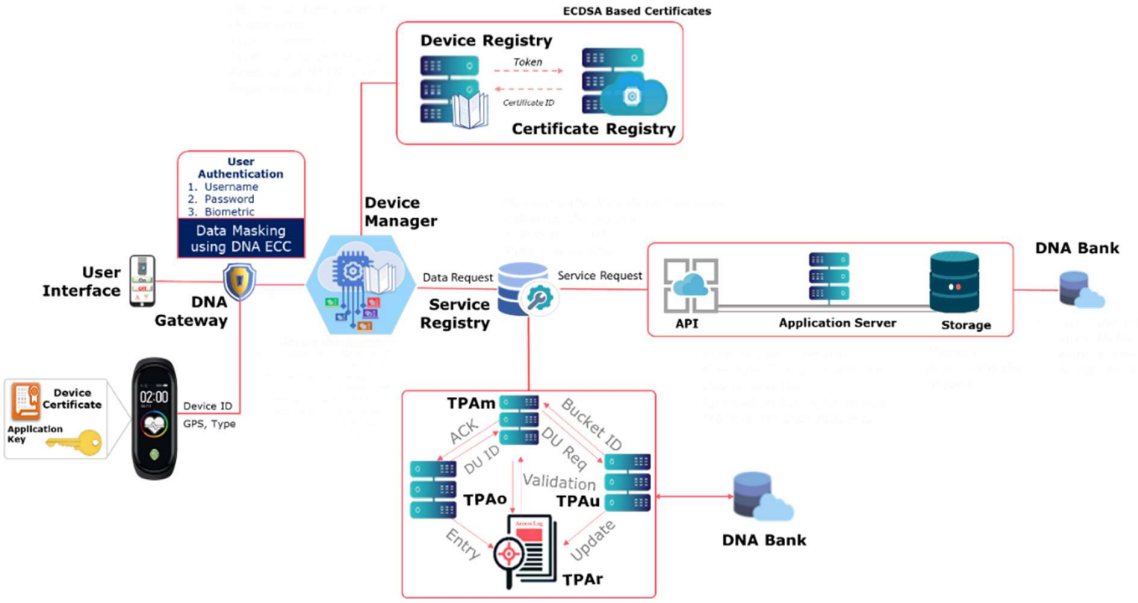


Figure 1. IoT Cloud framework with enhanced Data Privacy and Security

3.2 Encryption Technique

The data is encoded using **DNA** which then transfers it through **DG**. The device authentication is the key to enter the process. After authentication application token is generated. As the authentication becomes successful, the device manager generates a unique session token to handle the user access. The process of encryption is depicted here below and Table 3.1, shows the symbols used and their meanings.

Table 3.1. Symbols used to represent data

Symbols	Meaning
ϑ	Set containing ascii
ω	Variable holding a set of binaries
α	Set containing set of paired binaries
gc	Set containing genomes and associated binary values
β	Resultant of first cycle of substitution
γ	Set holding the grouped pairs of genomes into 4's
∞	Resultant of second cycle of substitution
ε	Vectors containing odd elements
ε'	Vector holding matrix associated ε
δ	Holds even set of data
δ'	Used to store matrix representing δ

Step 1: Initially ASCII equivalent for the input data is retrieved, as depicted in the equation 1. Here d is the data that is received at the **DPC**, f_{asc} is the function that converts it into ASCII, and ϑ represents the array that is generated.

$$\vartheta = fasc(d) \quad (1)$$

Process of Substitution

According to the ANSI standards every character in a given set is assigned with unique numbers as in the following table 3.2.

Table 3.2. ASCII Value Substitution

Characters	Associated Values
0 – 9	48 – 57
A – B	65 – 90
a – b	97 – 122
Symbols and Blank Space characters	Rest of the values between 0 and 256

For example

Let 'DNA', be the input value the resultant value after the process of calculation is 687797 as per the standard. For any integer value between 0 to 9 the ASCII will be between 48 and 57 respectively.

This phase of encoding works at an order of O(n), and at the successful conversion of data ϑ is passed for the further encoding process.

Step 2: After the retrieval of ASCII values, the binary equivalent is retrieved as equation 2, shows the conversion of ϑ into binary sequence. The ω hold the binary sequence retrieved by f_{bin} from the ASCII elements in ϑ.

$$\omega = fbin(\vartheta) \quad (2)$$

Process of Substitution

The binary equivalent of any integer can be calculated easily by subsequent division of the value by 2.

For instance,

Assume the number 687797 the ASCII value generated. This is changed into binary:

- Modulo divide the 68 and store the binary value (remainder) in an array
- Divide the number 68 and store the quotient value to the same variable

- Repeating the process will arrive with a final result, the resultant can be of any size and an eight-bit representation of the given character is 01000100 01001101 00011000.

Higher and random sized binary enhance the security, as it is hard to be guessed. Higher the randomization and bit size also increase the processing time.

Step 3: Binary equivalent is in ω, as in the equation 3, the process of encoding starts by grouping input in pairs of two elements. Here ω, presents its contents to the pairing function f_{par} that chops the entire contents into pair of twos.

$$\alpha = fpar(\omega) \quad (3)$$

The process of conversion is expanded as in equation 4. The sequence of elements represented as, $\omega(i)_j$ to $\omega(n)_m$. Here i represent the starting element in ω and n represent the last element in the set ω. And j count the bits in two's, and m the last bit. The f_{par} could be expanded as, α_i=$\omega_i + \omega_{i+1}$, which forms a pair iterating till ω_n the upper limit reaches. The final set α, could be written as in equation 4.

$$\sum_{i=0}^{n} \alpha = \sum_{j=0}^{m} \omega_j \, \omega_{j+1} \quad (4)$$

where i, the initial value of α and increases until n is reached at ω_n. And j is initialized with 0 stepping higher till m, while passing only two binary elements such as $\omega_j \; and \; \omega_{j+1}$. The routine takes O(n*2) for the successful partitioning of ω.

For example,
If the input value is 01100001, the binary sequence is generated as follows,

01	00	01	00	01	00	11	01	00	01	10	00

Step 4: Now binary pairs are encoded using the fixed set, of values as in the following table. The encoding is done based on the set generated. After the verification of pairs, the associated nucleotide is substituted replacing the binary pair. This the coding table is generated as in equation 5.

$$\sum_{i=0}^{4} gc = \sum_{i=0,j=0}^{1} (i + j) \quad (5)$$

The generated table is depicted in table 3.3. The set actually contain A, T, C, or G, which actually means adenine, thymine, cytosine and guanine respectively.

Table 3.3. Nucleotide Table for Substitution

Code	Genome
00	A
01	T
10	C
11	G

Step 5: The process of substitution $fdse$ is represented in equation 6, where each and every pair of elements is replaced with the genome as in equation 7. The encoded message is at β,

$$\beta = fdse(\alpha) \quad (6)$$

$$\sum_{i=0}^{n} \alpha = \sum_{i=0}^{m} gc \quad (7)$$

To understand the process, let us consider the example here,
If the input value is 01100001, the substitution of the genomes would result in the following set of values,

T	A	T	A	T	A	G	T	A	T	C	A

Step 6: Next process is to group the genome into pairs of four. This is achieved using the algorithm as represented in equation 8, and the process it does is equated in equation 9.

$$\gamma = fnuc(\beta) \quad (8)$$

$$\sum_{i=0}^{n} \gamma = \sum_{j=0}^{m} \beta_j \, \beta_{j+1} \beta_{j+2} \beta_{j+3} \quad (9)$$

For example, let us consider the genomic sequence generated recently,

For the input TATATAGTATCA, the substitution of the genomes would result in the following set of values,

TATA	TAGT	ATCA

Step 7: Generate nucleotide table, for the four-digit genomic sequence. The series is tabulated with an initial value a random number as represented in equation 10.

$$f_{ntbl}(\text{rand}(1, \text{len}(a)), len(\alpha) + 2) \quad (10)$$

Here to construct nucleotide coded set a random number between 1 and length of the actual data is taken. The set generated by f_{ntbl} is as described in Table 3.4. The order of time taken by f_{ntbl} to complete the process is calculated as O (n * 4).

Table 3.4. Nucleotide generated for the message

Text	DNA	Text	DNA	Text	DNA	Text	DNA	Text	DNA	Text	DNA
101	ATAA	109	AGGA	115	CTAA	123	ACAA	131	GATA	139	TAGT
102	TCTA	110	CAAA	116	GCTA	124	CTCA	132	GTAA	140	TACT
103	GCGA	Start	TTGA	117	GTCA	125	TGTA	133	ATGA		
104	GTGA	Stop	TAAA	118	CGTA	126	GAGA	134	AGTA		
105	AGAA	111	ACTA	119	CTGA	127	TATA	135	GACA		
106	CGCA	112	CATA	120	TGCA	128	CACA	136	GCAA		
107	ATTA	113	TCAA	121	TCGA	129	TGAA	137	AGCA		
108	ACCA	114	TACA	122	ATCA	130	TAGA	138	ACGA		

The message encoded using table is more secured, as random number is random. This is because random number depends on the size of the message that changes based on the application.

Step 8: The set β, is grouped in pairs of four by f_{nuc} as in equation 8. This is done by merging the set of elements, as $\gamma_i=\beta_i + \beta_{i+1} + \beta_{i+2} + \beta_{i+3}$, here γ is a set containing grouped message. Thus, the data are grouped and are stored in γ. The message γ thus encoded is ready for further encoding with the generated f_{nuc} by the module f_{enc} as in equation 11.

$$\infty = fenc(\gamma, f\,ntbl()) \quad (11)$$

For each and every entry of γ is replaced with associated index of $f_{nuc.}$ This actually generates the set of digits by fully hiding the message. The time taken for this process is in the order of O(n), this actually sustaining the speed of the process. The process is formulated as:

$$\sum_{i=0}^{n} \infty = \sum_{j=0}^{m} fntbl(j), \quad if\ \ \gamma_j <> \{\theta\} \quad (12)$$

Where, *i* representing the index of the resultant message set ∞, and *j* representing index of input set. Here both *i* and *j*, starts with 0 and the iteration steps with one till the value reaches *m* and *n*. The message is stored in ∞ as in equation 12.

Here let us consider the grouped sequence of genomes,
For the input TATA / TAGT / ATCA, the substitution of the cyphertext would result in a set of values as follows,

127	139	122

Step 9: As in the equation 13 & 14, f_{split}, takes ∞ as input and finds the odd numbers which are then stored into ε. The calculation is done with the simple mathematical calculation of modulo division. After modulo division with the digits, data that do not generate a 0 as result is stored at ε. And for each entry of ε, a separate vector ε' is generated by replace every non zero values with 1's as represented in equation 13.

$$\varepsilon = fsplit(\infty) \quad (13)$$

$$\delta = fsplit(\infty) \quad (14)$$

Thus, f_{split} could separate every odd value in the message. In equation 12 input calculated for separating the even numbers. The f_{split}, takes ∞ as input and finds the even numbers which are then

stored into δ by applying modulo division. The numbers that generate 0 is stored in δ. Parallely each entry in ε, a separate vector $\delta'$is generated by replace every non zero values with 1's as in equation 14. Thus, f_{split} separate every odd value in message.

$$\varepsilon' = fsplit(\varepsilon) \quad (15)$$

$$\delta' = fsplit(\delta) \quad (16)$$

After splitting the converted message in to two set of integers and matrix representation is generated, as in equation 15 and 16, f_{mdd} takes odd set ε, even set δ. The resultant denoted by μ, hold the message ready for undergoing the final phase of encoding process by **ECC**. Data Retrieval Matrix looks as in the following example:

127	139						1	1			0	0			
				122			0	0			1	0			
V	2			V	1										

Example - Data Retrieval Matrix

Step 10: Now the hash for the both the odd (ε) and even (δ) set is calculated using **SHA512** as represented in equation 17 and 18. The DRM is stored with the generated hash as the reference. This DRM set is stored in gateway for the use during decryption.

$$dh_1 = f_{sha}(\varepsilon) \quad (17)$$

$$dh_2 = f_{sha}(\delta) \quad (18)$$

Step 11: Before encryption, the process of key generation is done by f_h, The curve generated based on the curve equation 19, contains y, x, ax and b, of which (x, y) represent a point on the curve.

$$y2=x3 + ax + b \quad (19)$$

The domain parameters of curve are a, b, h, p, N, G, and r. Here 'p' representing prime number, 'a' and 'b' representing coefficients. The 'G' represents point, 'N' represents prime factor, 'h' represent the cofactor and 'r' representing random integer that is less than 'N'. Sinc NIST specifies the 'p' to be greater than 2^160, the equation 19 is rewritten as in equation 20, by assuming.

$$y2 = x3 + ax + b \bmod p \ (p>2\ 160) \quad (20)$$

To encrypt any given input, sender choses a random number 'r'. Using this 'r', point is calculated towards encrypting the text with the receiver's Public Key (**Pu**).

$$C= [(r.G), (M+r.\ Pu)]$$

Example for the key generation is depicted, by assuming the pain text as "W" which is represented by a value 4,433.

$y^2 = x^3$-ax+b (mod 997), a = (-3), b=3
$y^2 = (-3)^3$+1 (mod 997)
G = 17,427
Pr = 11
Pu = 37, 620
r = Random Number = 7

Step 12: The final phase of encryption is done by passing the data $\mu 1$ *and* $\mu 1$ to the f_{ecc} along with the pu is depicted in equation 21 and 22 subsequently. After data is **DNA** encoded and encryption process is over signature (**dnaSig**) is recorded. After the completion of process of encryption of f_{ecc}, the data is sent to the cloud for storage.

$$d_1' = fecc(\mu, pu) \quad (21)$$

$$d_2' = fecc(\mu, pu) \quad (22)$$

After the successful encoding, the data is again passed through another set of encoding process where a random salt (**Es**) is used to generate the code as the user request to store the data. This salt is named here as encryption salt (**Es**), which will be used to construct a sequential array of numbers

associated to the **DNA** codes generated in the previous cycle. The partitioned data will be separated as odd and even set, where the odd set of data will have its associated matrix generated and even set of data will have its own set of matrices generated. After the signature is calculated using SHA 512, the cipher text will be grouped in to two portions and two set of matrices are generated to handle both the portions of data separately.

DNA data is converted into the digital form which is partitioned into two groups $\mathbf{data_{si}}$ and $\mathbf{data_{sj}}$. The encrypted data along with the necessary credentials such as unique signature reach the cloud storage after the validation of **TPAo**. The cloud storage gateway after verifying the data with the received signature (**dnaSig)**, allocates the buckets for the data after processing. Thus, communication between **IoT** and cloud application are encrypted in a much-secured manner.

3.3 Secure Data Decryption

The request from the device at the **IoT** Environment is taken to cloud along with device certificate. The **CR** validates the certificate by extracting the device information, based on the credentials stored in the certificate registry. The firewall after making the entry on its device log, directs the request to the **TPAu**, while **TPAo** after making the entry on the log including the device credentials of the client and user credentials, sends the request to the **Do**. **Do** may provide the **A**ccess **T**oken (**Acct**). Now the **Acct** is transferred to **Du** through **TPAu**, for authorizing the requestor. After making the entry on the **TPAr**, **TPAu** verify the signature in the register with the signature of the data on the storage. After verifying the signature, the data will be transferred from the storage.

To decrypt of the data is done by the receiver, as it multiplies the first point of the ciphertext pair (r.G) with the private key (**Pr**). Now this result is added to the second point of the ciphertext pair.

$$M = (M + r.Pu) - (Pr\ \ (r.G)) = (M + r.\ Pr\ G) - (Pr\ (r.G))$$

After authentication of application token, session token and user credentials the encrypted data is decrypted and send to the client through the firewall reaching the **DNA** Gateway. As the gateway receives the data after verifying the **dnaSig**, it first merges the **DNA** matrix $\mathbf{data_{si}}$ and $\mathbf{data_{sj}}$, together $\mathbf{data_m}$ with the help of matrix created. Once identifying the code pattern generated for the data with the help of application id and the session id stored, it multiplies the even numbered **DNA** matrix with the n that is randomly chosen, where $n > 1 < 10$. Then it merges the **DNA** matrix together so that the actual **DNA** message.

The **DNA** data that is received would never require further processing but decoding based on the standard table that was generated for an application.

$$code_{si} => data_{si} \&\& code_{sj} => data_{sj} \quad \text{---------- 18}$$

$$data_m = data_{si} + data_{sj} \quad \text{---------- 19}$$

The next level of decryption involves merging the data together that can be decoded using that generated table. The **DNA** data thus calculated is combined to form a string altogether, from where the further process of decryption could proceed decoding the **DNA**'s. As the **DNA**'s are calculated using the generated application specific nucleotide code, final phase of decryption is done. After actual data is decoded out, the **dSig** is compared with calculated signature of data that is retrieved.

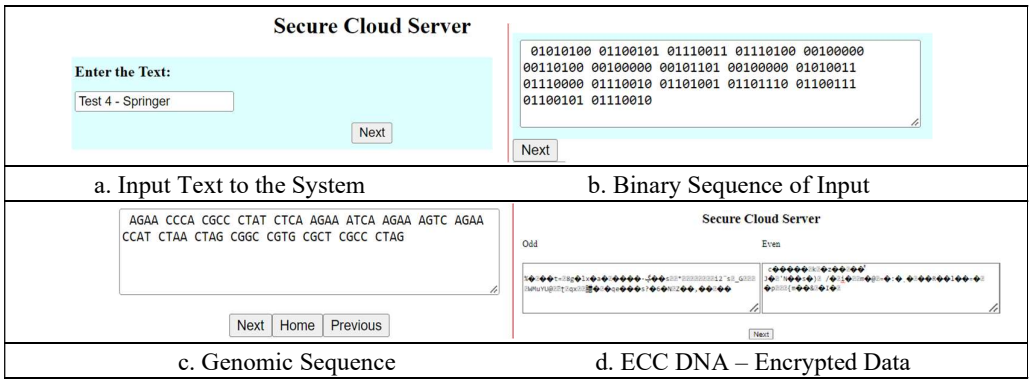
If verification successful the data is sent to client, or an error token *eT* is generated. The *eT* intimates the replace attack to **TPAo**, who further will keep **DOw** informed.

4. RESULT AND ANALYSIS

The implementation is done using opensource tools such as PHP and MYSQL. The interface is designed using HTML5 and CSS3. The cloud sim is used to evaluate the performance of the proposed framework.

4.1 Analysis on ECC DNA

The implementation of the work algorithm is done and results are generated. The input and the output, of the system is recorded to analyse the performance. The sample of the results generated by the proposed algorithm is present here in Figure 7.



a. Input Text to the System

b. Binary Sequence of Input

c. Genomic Sequence

d. ECC DNA – Encrypted Data

Figure 7. Sample Input and Output – Proposed ECC DNA

The figure 7a, shows the inputs text of size 16 characters, figure 7b, displays the binary sequence of every character. On subsequent conversion the binary is converted into nucleotides as in figure 7c. Finally, the figure 7d, shows the final cypher text ready for transmission. The analysis on the proposed **DNA** algorithm is presented here by sampling the data. The efficiency of the algorithm on the basis of time taken is analysed as in Table 4.1.

Table 4.1. Analysis on Encryption of Data- Sample 1

Test	Plain Text	Cipher Text	Key Size	Time (MS)
Test 1	10	80	1024	0.07
Test 2	20	160	1024	0.12
Test 3	50	400	1024	0.17
Test 4	120	960	1024	0.23
Test 5	240	1920	1024	0.32

The key size chosen is 1024 bits and at various input sizes the time taken is analysed. At a maximum of 240 characters of data is encrypted in 0.32 ms. The resultant cyphertext is of size 1920 bits.

Table 4.2. Analysis on Encryption of Data - Sample 2

Test	Plain Text	Cipher Text	Key Size	Time (MS)
Test 6	10	80	2048	0.08
Test 7	20	160	2048	0.12
Test 8	50	400	2048	0.20
Test 9	120	960	2048	0.24
Test 10	240	1920	2048	0.33

The process of decryption is done after the implementation of the algorithm. The sample data set for analysis of the results generated is presented here in table 4.2. The time taken for encryption is analyzed and is summarized as follows:

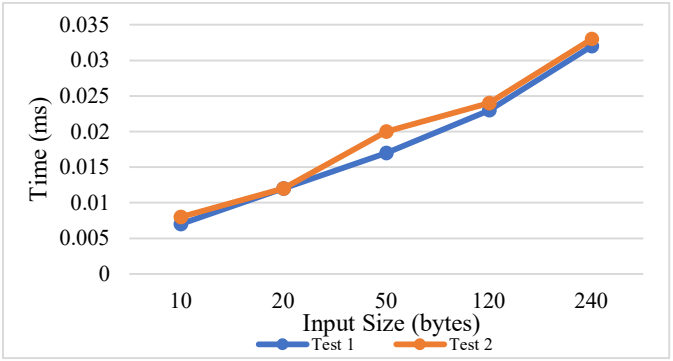


Figure 8. Time taken by the ECC DNA for Encryption

The figure 8, describes the time taken by the encryption algorithm for encrypting the plain text. Test conducted is based on two key patterns, first is with 1024 bit key and next is using 2048 bit key. As per the results generated by the system the average time taken for encrypting 10 byte of data is 7.5 ms. The maximum size reported for this work is 240 byte, for which an average time of 32.5 ms is recorded. Therefore, the **ECCDNA** proves better for encryption.

Table 4.3. Analysis on Decryption of Data - Sample 1

Test	Cipher Text	Plain Text	Time (MS)
Test 1	80	10	0.08
Test 2	160	20	0.14
Test 3	400	50	0.16
Test 4	960	120	0.26
Test 5	1920	240	0.31

The implementation is tested to verify its efficiency during data decryption and results generated are sampled as in Table 4.3. Algorithm performed the decryption process in an efficient manner. The variable length data is used as input to check the efficiency. The minimum size of data is performed at an average of 9ms and the data with about 1920 bytes is performed at an average of 35ms.

Table 4.4. Analysis on Decryption of Data - Sample 2

Test	Cipher Text	Plain Text	Time (MS)
Test 6	80	10	0.09
Test 7	160	20	0.13
Test 8	400	50	0.18
Test 9	960	120	0.23
Test 10	1920	240	0.35

The efficiency of data decryption process is described in Table 4.4, which shows a consistency in the time taken for decryption while compared with the previous set of data. As per the results, it is sure that the encryption algorithm is efficient and faster enough to work with any set of data with a higher size of key.

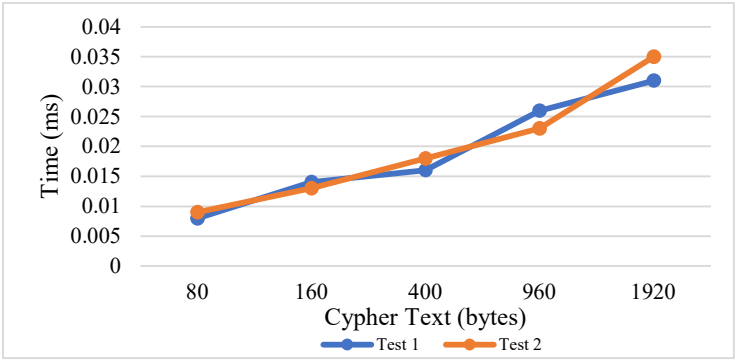


Figure 9. Time taken by the ECC DNA for Decryption

The figure 9, describes the time taken by the encryption algorithm for encrypting the plain text. As per the results generated by the system the average time taken for decryption for 80 byte encrypted input data is 8.5 ms. The time taken is to decrypt the data of size 1920 bytes, for which an average time taken is recorded as 33 ms.

The proposed work while compared with the existing approaches found to have better performance. The approach included RC4 along with **DNA**. The comparison is presented here below in table 4.5.

Table 4.5. Comparison with Existing Proposals

	RC4	**RC4 – DNA**	**Proposed ECC – DNA**
Plain Text (bytes)	128	128	120
Key Size (bytes)	256	256	1024
Encryption Time (ms)	0.414	0.584	0.23
Plain Text (bytes)	256	256	240
Key Size (bytes)	256	256	2048
Encryption Time (ms)	0.576	0.69	0.33

The results depict the efficiency of the proposed algorithm as the key size increases the time taken for encryption also increases. Among the present scenario, such situation the proposed **ECC DNA** after testing with higher key sizes 1024 and 2048 could perform better up to 0.023 ms and 0.033 ms respectively. Therefore, the proposed **ECC DNA** proves performing better at encryption.

4.2 Analysis of Proposed Framework

Test is conducted using an opensource tool Cloud Analyst. The load test, response time and throughput were recorded. Series of requests were generated to test performance of the system as in Figure 9. It shows that the system could accept multiple request if the system is implemented.

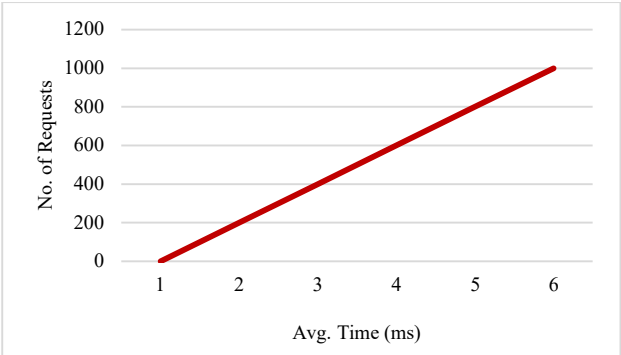


Figure 9. Analysis of the Parrel Requests

As any request to have a task done response time of the system is recorded and found to be reasonable as in Figure 10. After the series of requests are received, the calculation is done with the input and the response is sent back. The minimum time taken to respond is 6.8 ms and a maximum of 16.5 ms.

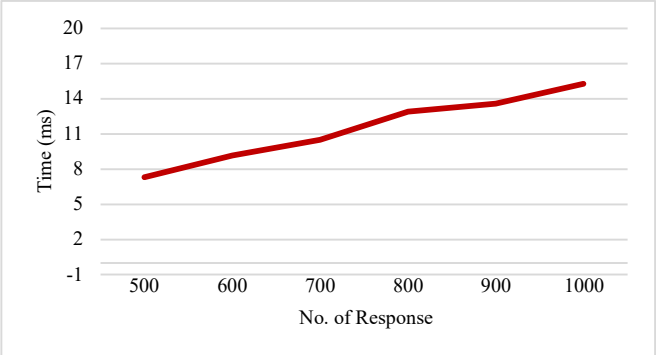


Figure 10. Analysis on the Response Time of the System

During the communication, network show up throughput as the request and response are higher. On the analysis of throughput is depicted as in Figure 11. It also depends on the type of network used, here the throughput recorded is between 2.2 ms to 4.4 ms on an average.

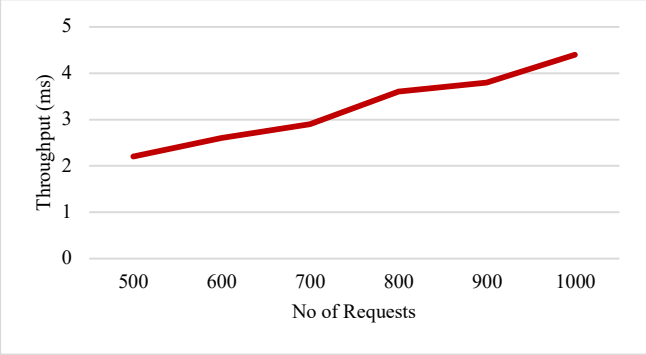


Figure 11. Analysis of Throughput of the System

As per the study it is found that performance is also affected depending on medium that is used and the other aspects of devices being employed. The proposed mechanism performs better on analysing the aspects such as request, response and throughput.

5. CONCLUSION

The analysis reflects the efficiency of the security techniques as the privacy attacks such as data leakage is mitigated. As described in [4, 8-12] that **DNA** based security algorithms with **ECC** provide better security at **IoT** level. Hybrid algorithms proposed could encrypt the data as fast as 0.033 ms as the size of key is 2048. The decryption time stands equally faster up to 0.035 ms. Data privacy and integrity stands high as security is strong. **DNA** is efficient in such a way that it could enhance the security and privacy as it is prominent at data hiding and the proposed architecture enhance the privacy and security by enforcing authentication and repudiation using TPAC. As per the results generated, it is clear that using Hybrid encryption could enhance the security in **IoT C**loud.

REFERENCES

[1] R. Van Kranenburg, "The Internet of Things: A Critique of Ambient Technology and the All-Seeing Network of RFID", Amsterdam, The Netherlands: Institute of Network Cultures, 2007.
[2] L. Tan, N. Wang, "Future internet: The internet of things," in Proc. 3rd Int. Conf. Adv. Comput. Theory Eng. (ICACTE), Chengdu, China, 2010, pp. 376–380.
[3] Y. Wu, Q. Z. Sheng, S. Zeadally, "RFID: Opportunities and challenges," in Next-Generation Wireless Technologies, N. Chilamkurti, Ed. New York, NY, USA: Springer, 2013, pp. 105–129.
[4] Vidhya Vijayan, Eldo P Elias, "Hybrid Method for Securing Data in IoT Cloud", International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2019.
[5] Sowmya Nagasimha Swamy, Dipti Jadhav, Nikita Kulkarni, "Security Threats in the Application layer in IOT Applications", International conference on I-SMAC, 2017.
[6] Joel J. P. C., Dante B. R., Heres A., Murilo H., Rafael M., Jalal Al-Muhtadi, Victor Hugo C., "Enabling Technologies for the Internet of Health Things", IEEE, 2018, ISSN: 2169-3536.
[7] Samhita Kanthavar, "Design of an Architecture for Cloud Storage to Provide Infrastructure as a Service (IaaS)", IEEE, 2017.
[8] Noor A. Hussein, Mohamed Ibrahim Shujaa, "DNA computing-based stream cipher for internet of things using MQTT protocol", International Journal of Electrical and Computer Engineering (IJECE), Vol.

10(1), February 2020, pp. 1035 – 1042.
[9] Naga Saranya Cherukupalli, Sesha Shayee Maruvada, "Securing Data in IoT Devices using DNA Cryptography", International Journal for Modern Trends in Science and Technology, 6(8S), 2020.
[10] Nayna Agarwal, Anand Mahendran, Ramanathan Lakshmanan, "Trusted Third Party Auditing for Cloud Security Using Digital Signature and DNA Cryptography", IJSTR, Vol 8(12), 2019.
[11] Harsh Durga Tiwari, Jae Hyung Kim, Novel Method for DNA-Based Elliptic Curve Cryptography for IoT Devices, ETRI Journal, Vol. 40 (3), 2018. (http://wileyonlinelibrary.com/journal /etrij)
[12] Barman, P., Saha, B. "DNA encoded elliptic curve cryptography system for IoT security". International Journal of Computational Intelligence & IoT. 2, 2019.
[13] D. Prabhu, M. Adimoolam, "Bi-serial DNA Encryption Algorithm" [Online]. Available: https://pdfs. Semantic scholar. org/ 1754/ f0eb5 85 25005 98 a70af 4002 e186 cd2f 3c6 ce.pdf
[14] Kang Ning, "A Pseudo DNA Cryptography Method", arXiv:0903.2693 [cs.CR], Cornell University Library, 2009.
[15] Kritika Gupta, Shailendra Singh, "DNA Based Cryptographic Techniques: A Review", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3 (3), 2013.
[16] Harish Kumar N, Rajshekhar M Patil, Deepak G, Murthy B M, "A Novel Approach for securing data in IoTCloud Using DNA Cryptography and Huffman Coding Algorithm", 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2017.
[17] Malti Bansal, Shubham Gupta, Siddhant Mathur, Comparison of ECC and RSA Algorithm with DNA Encoding for IoT Security, Proceedings of the Sixth International Conference on Inventive Computation Technologies [ICICT 2021], IEEE Xplore, 2021.
[18] Asha Jose, Kamalraj Subramaniam, "DNA based SHA512-ECC cryptography and CM-CSA based steganography for data security", Materials Today: Proceedings, Elsevier, 2020.
[19] Eman I. Abd El-Latif & M. I. Moussa "Information hiding using artificial DNA sequences based on Gaussian kernel function" Journal of Information and Optimization Sciences, 2019, ISSN: 0252-266.
[20] Zena Ahmed, Saher Adil, Saif M. Kh. Al-Alak, ECC Based Blind Steganography-DNA for Hidden Information. Journal of Engineering and Applied Sciences, 14, 2019.
[21] M. Kumar, "Implementation of DNA cryptosystem using Hybrid approach", Research Journal of Computer and Information Technology Services, Vol. 6(3), 2018.
[22] Mansi Rathi, Shreyas Bhaskare, Tejas Kale, Niral Shah, Naveen Vaswani, "Data Security Using DNA Cryptograph", International Journal of Computer Science and Mobile Computing, Vol.5, 2016, pg. 123-129.
[23] Abusaimeh, H., "Security Attacks in Cloud Computing and Corresponding Defending Mechanisms", International Journal of Advanced Trends in Computer Science and Engineering, 9(3), 4141–4148, 2020.
[24] Aditya, K., Mohanty, A. K., Ragav, G. A., Thanikaiselvan, V., & Amirtharajan, R., "Image encryption using dynamic DNA encoding and pixel scrambling using composite chaotic maps", IOP Conference Series: Materials Science and Engineering, 872(1), 12045, 2020.
[25] Al-Shargabi, B., Al-Husainy, M. A. F., "A New DNA Based Encryption Algorithm for Internet of Things", International Conference of Reliable Information and Communication Technology, IRICT 2020, 786–795, 2021.
[26] Sunil Raj Y, Albert Rabara S, Britto Ramesh K. S., "A Security Architecture for Cloud Data Using Hybrid Security Scheme", Proceedings of the Fourth International Conference on Smart Systems and Inventive Technology (ICSSIT-2022), IEEE, 2022, pg. 1803 – 1811, ISBN: 978-1-6654-0117-3.
[27] Sunil Raj Y, Albert Rabara S, "An Integrated Architecture for IoT Based Data Storage in Secure Smart Monitoring Environment", International Journal of Scientific & Technology Research, V0l. 8(10), 2019. ISSN: 2277-8616.
[28] Kolate, V., Joshi, R. B., "An Information Security Using DNA Cryptography along with AES Algorithm", Turkish Journal of Computer and Mathematics Education, Vol, 12(1S), 183–192, 2021.
[29] Dhuha Khalid Alferidah and NZ Jhanjhi, "A Review on Security and Privacy Issues and Challenges in Internet of Things", IJCSNS International Journal of Computer Science and Network Security, VOL.20 (4), 2020.
[30] Y. Yu, Y. Li, J. Tian, and J. Liu, "Blockchain-Based Solutions to Security and Privacy Issues in the Internet of Things," IEEE Wireless Communications, vol. 25, no. 6, pp. 12-18, 2018.
[31] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A Survey on Security and Privacy Issues in Internet-of-Things," IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1250-1258, 2017
[32] A. Assiri, and H. Almagwashi, "IoT Security and Privacy Issues," 2018 1st International Conference on Computer Applications & Information Security (ICCAIS), pp. 1-5, 2018.