# EXPERIMENTAL RESULTS AND DISCUSSION:

By the combination of ECC, Diffie-Hellman, AES and SHA-256 the efficiency and individuality of the system is improved in the numerous ways so that for the encryption and decryption of the data we can acquire the secured connections and many of the data over cloud storage are secured. This signifies that by employing these two techniques the user can obtain the original message without difficulty. In the following subsection specific benefits of ECC and AES over RSA are presented.

* 1. *Advantages of ECC and AES*

Over the cloud storage ECC is used to safeguard the security of the data. With reduced key size can help to enhance the storage space along with to obtain the anticipated outcomes the maintenance of the data. As the RSA it uses the same 3072 bits. The ECC is the reduced key size and the encryption of data because of a public key, this is the most advantageous thing about this which is in an boosted approach [[32](#_bookmark57)]. For using the most recent algorithmic techniques employing to the encryption and decryption of data in addition to the accuracy of the decrypted data compared to the RSA, ECC is more beneficial. Operations which are controlling on cloud storage like searching on cloud storage, statistical analysis, and others like these AES has many performance operations. Over cloud computing to enhance the security rules over cloud storage it is the widely used strategic algorithm. The public key is identified by every individual while the encryption and decryption procedure can also be completed by the public key [[33](#_bookmark58)]. Many advantages and key sizes of ECC and AES over RSA are given in the Table [5](#_bookmark13) below.

|  |  |  |
| --- | --- | --- |
| **Table 5.** ECC and AES over RSA. |  | |
| **ECC** | **RSA** | **Key Size Comparison Ratio** |
| 160 bits | 1024 bits | 1:6 bits |
| 256 bits | 3024 bits | 1:12 bits |
| 384 bits | 7068 bits | 1:20 bits |
| 512 bits | 16360 bits | 1:20 bits |

Explanation: As you can perceive in the Table [5](#_bookmark13) and Figure [6](#_bookmark14), as compared to RSA a medium key size is needed for ECC. It provides better security than RSA because of this. As compared to other cryptographic algorithms ECC-AES also provides better security with a smaller key size. It decreases computational complexity and optimizes memory space because of the smaller key size. Thus, a high level of data security can be gained by using a medium key size.

Key Size



**Graphical Representation of Key Size**

18,000

16,000

14,000

12,000

10,000

8000

6000

4000

2000

0

1

2

3

4

Key Ratio

ECC

RSA

**Figure 6.** Graphical representation of key sizes.

* 1. *Various Algorithms Comparisons*

The analysis of different algorithms is compared to check the functionality and space optimization over cloud storage [[34](#_bookmark59)]. The comparison of different algorithms is given below in Table [6](#_bookmark15).

**Table 6.** Comparison of different Cryptographic Algorithms.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Factors** | **Proposed** | **DES** | **3DES** | **Blowfish** | **RSA** | **Diffie-Hellman** |  |  |  |  |  |  |  |
| No. of key | 1 | 1 | 1 | 1 | 2 | Key Exchange |  |  |  |  |  |  |  |
| Key Length in bits | 64–256 | 56 | 112–168 | 32–448 | 1024 | Key Exchange |  |  |  |  |  |  |  |
| Rounds | 10 | 16 | 48 | 16 | 1 | 56 |  |  |  |  |  |  |  |
| Limitation  power | Brute force | Brute force | Computational | Key frequently change | Key generation weak | Cant encrypt daa |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Factors** | **Proposed** | **DES** | **3DES** | **Blowfish** | **RSA** | **Diffie-Hellman** |
| No. of key | 1 | 1 | 1 | 1 | 2 | Key Exchange |
| Key Length in bits | 64–256 | 56 | 112–168 | 32–448 | 1024 | Key Exchange |
| Rounds | 10 | 16 | 48 | 16 | 1 | 56 |
| Limitation | Brute force | Brute force | Computational power | Key frequently changing | Key generation week | Cannot encrypt data |

As you can see in the above Table [6](#_bookmark15), it shows the analysis of different cryptographic techniques based on different factors. Cryptographic algorithms have been compared with each other for performance evaluation on basis of number of keys used, keys in bits, rounds, and limitations.

* 1. *Data Size for Proposed Scheme*

We took three different image datasets for the comparison of our proposed method with other existing schemes because images usually take more time than text data, so we wanted to check the computational cost as well as the time required to complete the encryption and decryption of the images. The reason for taking different images size is to compare performance in multiple scenarios. The size of the image datasets on which the experiments were performed is 3233, 4830, and 6308. Figure [7](#_bookmark16) represents the encryption time comparison of different algorithms.

## *3.4 Data and Performance Metric*

In the implementation of our model would be, the data which were considered are the banking datasets, comprising information of all known bank deposits throughout the world. Further, we will analyzed the performance of our model by determining the execution time for encryption and decryption and their throughputs. The performance metrics are explained as follows:

1. Execution Time (*t* ): To translate plaintext to a ciphertext (i.e., encryption) or vice versa (i.e., decryption) this is the time to taken. It is evaluated in milliseconds (ms). For enhance performance of a cloud computing security system, a slighter execution time is preferred.
2. Throughput (*Th*) : is described as input file size (*Fs*) for each unit execution time (*t*). For encryption, the throughput is evaluated as the file size (i.e., plaintext) divided by the encryption execution time, although for decryption, it is the ciphertext divided by the decryption execution time. For enhance performance of a cloud computing security system, a greater throughput is preferred.

# Results and Analysis

The above methodology is implemented in JAVA 8 using Eclipse an open source platform that allows a developer to create a customized development environment (IDE). The above experiment is conducted on different text files with different sizes (KB). Files with different sizes are taken as input to the system and evaluation is done on the basis of different parameters like correlation, Avalanche effect, storage, Encryption time and Decryption time.

### 5.1. Parameter Evaluation using AES-ECDH

Different text files are taken as input and encrypted using AES-ECDH hybrid approach. Performance of the used hybrid approach is evaluated on the basis of different parameters given in table II below. Three metrics were taken to analyze the proposed methodology storage, encryption time and decryption time. Different text

files are taken as input with varying file size, after encryption we can see file size will increase and encryption and decryption time of hybrid approach is better.

### Table 2. Evaluating Storage, Encryption Time and Decryption Time

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| File Name | Original file Size  (kb) | Encrypted file size (kb) | Encryption Time (ms) | Decryption Time (ms) |
| License.  txt | 7650 | 13811 | 17 | 7 |
| new.txt | 152135 | 275527 | 37 | 27 |
| new1.txt | 581469 | 1057809 | 57 | 54 |
| readme.t  xt | 2878 | 5217 | 10 | 3 |
| example  .txt | 4106 | 7447 | 11 | 4 |

In Table III, evaluation of AES-ECDH is done on the basis of avalanche effect and correlation taking again different files as input. Let’s have a brief look on what is Avalanche effect and Correlation coefficient.

**Avalanche Effect:** Avalanche effect is the property of an algorithm which tells us how much change in bits will be there in the encrypted text if we change one bit of plain text. In Table 3 below every input text file holds good percentage of avalanche effect. if an algorithm holds good percentage of avalanche effect then it’s too difficult for the intruder to break the security. Also enormous number of cipher texts can be generated by making just one bit variation.

Avalanche =

Number of Flipped bits in the Cipher Text Number of bits in the Cipher Text

**Correlation:** Correlation tells about the dependence among two variables or one can say similarity or relationship between the two variables so for system to be more secure correlation must be 0 or nearest to 0 which indicates no similarity between two variables. Here in the proposed work encrypted text and plain text files are checked whether they exhibit some linear relationship or not. Correlation can be calculated using formula given below where x, y are two variables, r be the calculated relationship among them and s is the standard deviation.

## ∑n(xi − μx)(yi − μy)

r = 1

## (n − 1)Sx Sy

Table 3 represents the calculated avalanche effect and correlation among different files. Correlation is almost nearest to 0 which implies that there is almost negligible relationship between plain text and encrypted text, so it is difficult to obtain plain text by encrypted text.

### Table 3. Evaluating Avalanche Effect and Correlation

|  |  |  |  |
| --- | --- | --- | --- |
| File Name | Original File Size | Avalanche | Correlation |
| License.txt | 7650 | 76.32 | 0.3 |
| new.txt | 152135 | 76.99 | 0.33 |
| new1.txt | 581469 | 79.5 | 0.26 |
| readme.txt | 2878 | 77.23 | 0.29 |
| example.txt | 4106 | 74.16 | 0.39 |



1200000

1000000

800000

600000

400000

200000

0

Original File Size

Encrypted File Size

**Figure 4. Encrypted File size of Different Files**

Figure 4, Represents the graphical outputs of storage which is encrypted file size against the original file sizes taken as input. Storage is large than original file size for every set of input file. Figure 5 gives graphical outputs of encryption and decryption time. Computation speed of algorithm is much high as it requires much less time for encryption and decryption.



60

50

40

30

20

10

0

Encryption Time

Decryption Time

### Figure 5. Encryption/Decryption Time of Different files

Figure 6 and Figure 7 gives avalanche effect and correlation against files taken as input to the hybrid system. Avalanche is quite high that enhance the security of the system as number of ciphers can be formed by making a single bit variation in the text without compromising the security. Avalanche for License.txt is 76.32, new.txt is 76.99, new1.txt is 79.5, readme.txt gives 77.23 percentage and example.txt exhibits 74.16. So avalanche is much good in percentage for all inputs files that ensure long term security.



**avalanche**

80

78

76

74

72

70

avalanche

### Figure 6. Avalanche Effect in Different Files

Figure 7 exhibits the correlation that can be seen from Table 3, almost nearest to zero which implies that there’s no similarity between the original text and the cipher text or there is much less dependency between the original input file and cipher file. Correlation for License.txt is 0.3, new.txt is 0.33, new1.txt is 0.26, readme.txt gives

0.29 and example.txt exhibits 0.39. Correlation is almost closer to ‘0’ which signify

no similarity between original and output cipher file.



**Correlation**

0.45

0.4

0.35

0.3

0.25

0.2

0.15

0.1

0.05

0

Correlation

**Figure 7. Correlation in Different Files**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Factors** | **Proposed** | **DES** | **3DES** | **Blowfish** | **RSA** | **Diffie-Hellman** |
| No. of key | 1 | 1 | 1 | 1 | 2 | Key Exchange |
| Key Length in bits | 64–256 | 56 | 112–168 | 32–448 | 1024 | Key Exchange |
| Rounds | 10 | 16 | 48 | 16 | 1 | 56 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Factors** | **Proposed** | **DES** | **3DES** | **Blowfish** | **RSA** | **Diffie-Hellman** |
| No. of key | 1 | 1 | 1 | 1 | 2 | Key Exchange |
| Key Length in bits | 64–256 | 56 | 112–168 | 32–448 | 1024 | Key Exchange |
| Rounds | 10 | 16 | 48 | 16 | 1 | 56 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Factors** | **Proposed** | **DES** | **3DES** | **Blowfish** | **RSA** | **Diffie-Hellman** |
| No. of key | 1 | 1 | 1 | 1 | 2 | Key Exchange |
| Key Length in bits | 64–256 | 56 | 112–168 | 32–448 | 1024 | Key Exchange |
| Rounds | 10 | 16 | 48 | 16 | 1 | 56 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Factors** | **Proposed** | **DES** | **3DES** | **Blowfish** | **RSA** | **Diffie-Hellman** |
| No. of key | 1 | 1 | 1 | 1 | 2 | Key Exchange |
| Key Length in bits | 64–256 | 56 | 112–168 | 32–448 | 1024 | Key Exchange |
| Rounds | 10 | 16 | 48 | 16 | 1 | 56 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Factors** | **Proposed** | **DES** | **3DES** | **Blowfish** | **RSA** | **Diffie-Hellman** |
| No. of key | 1 | 1 | 1 | 1 | 2 | Key Exchange |
| Key Length in bits | 64–256 | 56 | 112–168 | 32–448 | 1024 | Key Exchange |
| Rounds | 10 | 16 | 48 | 16 | 1 | 56 |

1. **Avalanche Effect:** Avalanche effect is the property of an algorithm which articulates us in the encrypted text how much transformation will be there in bits if we modify one bit of plain text. Every input text file should contain suitable percentage of avalanche effect. If an algorithm holds decent percentage of avalanche effect then for the intruder it’s too challenging to break the security. Furthermore by making just one bit variation huge number of cipher texts can be produced.

Avalanche =

In the Cipher Number of Flipped bits

In the Cipher Text Number of bits

**Correlation:** Correlation speaks about the requirement among one or two variables can articulate relationship or similarity between the two variables consequently for system to be extra protected correlation must be 0 or nearest to 0 which specifies no connection between two variables. Here in the proposed work plain text files and encrypted text are verified whether they demonstration some linear relationship or not. Using formula given below correlation can be evaluated where x and y are two variables, s is the standard deviation and r be the calculated relationship among them.

## ∑n(xi − μx)(yi − μy)

r =

(n − 1)Sx Sy