6

Secured Cryptographic Approach for the Outsourced Mobile Device Data over Cloud Storage using the Proposed MSAES Method

6.1 Introduction

To ensure mobile device security, an enhanced malware detection using MSGP-MS classifier is explained in chapter 5. Increase in mobile device sophistication also demands high storage sophistication. To overcome this problem, a mobile device ought to get resources from an external source known as Cloud Storage. With the emergence of Cloud computing in mobile web, mobile users can use infrastructure, platform, software provided by cloud providers on on-demand basis. Emergence of Cloud Computing with mobile devices gave birth to Mobile Cloud Computing.

MCC allows resource constrained mobile users to adaptively adjust processing and storage capabilities by transparently partitioning and offloading the computationally intensive and storage demanding jobs. Also, increase in feature rich mobile applications like mobile wallets, banking apps, and healthcare app etc. expose sensitive data of the users which are always prone to much vulnerability. The value of data is far more important than the value of the device. The main issue in using MCC is securing the user data on cloud storage, since there is a high risk of unauthorized access to the data. So the main concern is to provide data security and at the same time providing ease of access to the authorized user.

In this chapter, the combined algorithm namely, MSAES which is a combination of hybrid cryptographic algorithms such as Signature based Symmetric techniques are used to secure the mobile device data over cloud storage. In order to overcome the security issues of the mobile device data over cloud storage, a three-tier security mechanism using a hybrid cryptographic encryption based approach is proposed to provide security with privacy and integrity.

6.2 Steps of Proposed Contribution Three- MSAES Method

The objective of the contribution three is to enhance security of mobile device data over cloud storage using three-tier security mechanism. The proposed contribution three consists of following steps and are discussed below.

- MSAES Encryption for Uploading process
- MSAES Decryption for Downloading process

To enhance the security of mobile device and data on cloud storage, a combined method called MSAES is proposed. MSAES is a combination of Signature based Digest Advanced Encryption Standard algorithm. The steps of proposed contribution three is shown in figure 6.1.

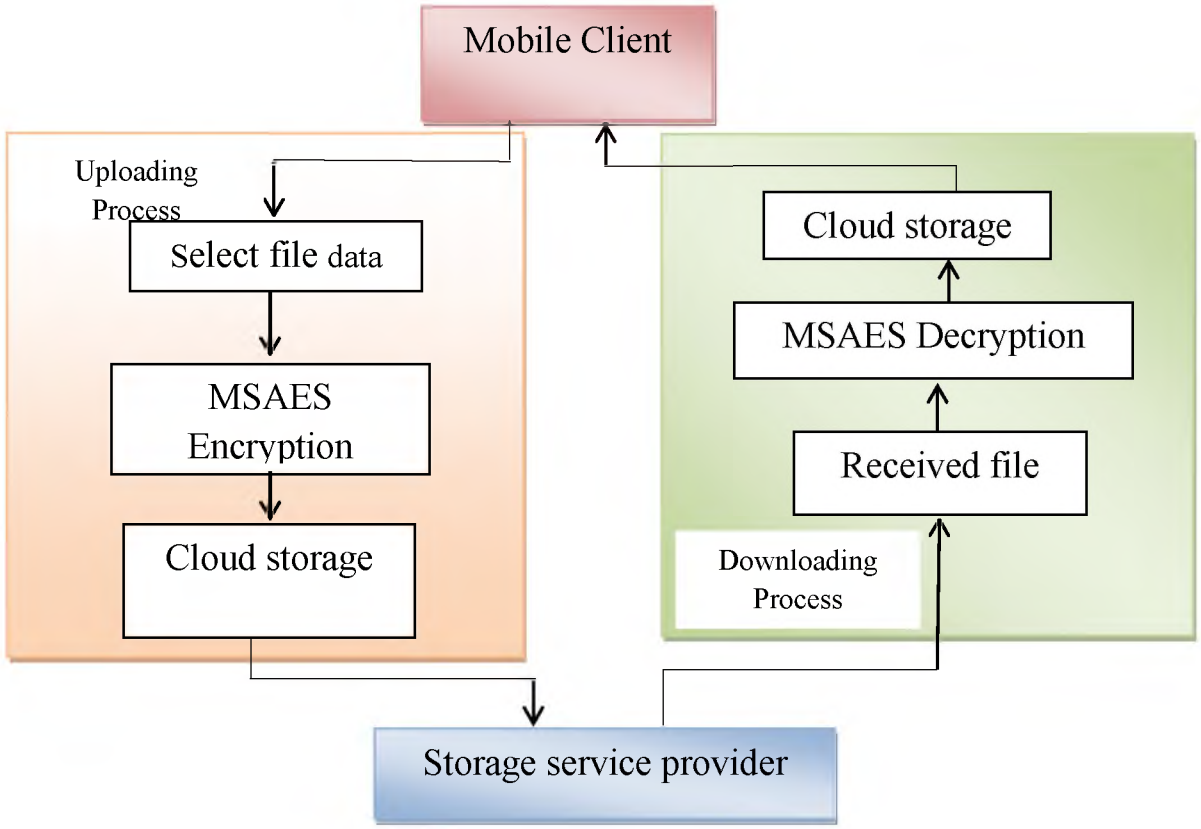


Figure 6.1 Block Diagram of Proposed Contribution MSAES Method

6.2.1 MSAES Encryption for Uploading Process

Encryption solves the data security issues with confidentiality and correctness of the data. It is the most commonly used technique to protect data within cloud environment. While using the cloud storage services on resource constraint mobile device, the mobile user needs to ensure the security of the mobile data before uploading on the cloud storage.

The uploading process of mobile cloud computing for secure data transfer over cloud is discussed below:

Step 1: Before uploading file F into CS, MD prompts for asking U to input a password, denoted as PWD.

Step 2: MD generates encryption key EK= H(PWD || FN) and integrity key IK=H(FN || PWD) , where FN is the name of the file F (character string will be changed to bit string).

Step 3: MD encrypts F with EK as F′ = ENC(F,EK). MD generates file integrity authentication code, denoted as MAC = {H(F,IK)}.

Step 4: MD sends {F′ || H(FN) ||MAC} to portal CS. MD stores T = ⟨FN⟩ locally and deletes EK and IK.

In the encryption process, the user selects the mobile data to be outsourced. A key from the user is obtained and encrypted at client side using the hybrid appoach of MSAES algorithm. After the encryption at local environment, data is outsourced over the cloud environment as shown in figure 6.2.

Figure 6.2 MSAES Encryption for Uploading Process

The cryptographic algorithms used are hybrid combination of Symmetric key and Asymmetric key algorithm with signatures. The hybrid cryptographic algorithms are discussed below.

6.2.1.1 Symmetric Advanced Encryption Standard Algorithm

Symmetric cryptography uses the same cryptographic key for both encryption of plaintext and decryption of cipher text. The proposed framework has been implemented using Advanced Encryption Standard which is discussed below.

Advanced Encryption Standard (AES) is a symmetric encryption algorithm. The ciphers have a 128-bit block size, with key sizes of 128, 192 and 256 bits respectively. AES uses 10, 12, or 14 rounds. The key size can be 128,192 or 256 bits depending on the

number of rounds. AES uses several rounds in which each round is made of four different stages as follows.

Stage 1: Substitute Bytes

Uses an S-box to perform a byte-by-byte substitution of the block. It uses one table of 16x16 bytes containing a permutation of all 256 8-bit values. Each byte of state is replaced by byte indexed by row (left 4-bits) and column (right 4-bits). S-box is constructed using defined transformation of values in GF(2^8).

Stage 2: Shift Rows

A simple permutation with a circular byte shift is done in each row as follows.

1st row is unchanged

2nd row does 1 byte circular shift to left

3rd row does 2 byte circular shift to left

4th row does 3 byte circular shift to left

Stage 3: Mix Columns

The substitution makes use of arithmetic over GF (2^8). Each byte is replaced by a value dependent on all 4 bytes in the column. The coefficients are based on linear code with maximal distance between code words.

Stage 4: Add Round Key

A simple bitwise XOR state with 128-bits of the round key of the current block is done with a portion of the expanded key.

6.2.1.2 Asymmetric Elliptic Curve Cryptography Algorithm

Asymmetric cryptography or public-key cryptography uses a pair of keys to encrypt and decrypt a data. Initially user receives a public and private key pair from a certificate authority. The proposed system implements the Elliptic Curve Cryptography as follows.

The ECC is defined as $a * k = (a + a + \cdots . + a)$ for k times.

Step 1: Encode message

Encode any message M as a point on the elliptic curve P_m.

Step 2: Key

Each user chooses private key and compute public key defined as, $P_A = n_A G$

Step 3: Encryption

To encrypt and send a message P_m to B, A chooses a random positive integer k and produces the ciphertext C_m consisting of the pair of points as, $C_m = \{kG, P_m + kP_b\}$

Step 4: Decryption

To decrypt ciphertext, B multiplies the first point in the pair by B's secret key and subtract result from second point as,

$$P_m + k P_b - n_B(kG) = P_m + k (n_B G) - n_B(kG) = P_m$$

The next section discusses about the asymmetric RSA approach.

6.2.1.3 Asymmetric Rivest Shamir Adleman Algorithm

RSA is a public key algorithm designed by Ron Rivest, Adi Shamir, and Leonard Adleman. It involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The RSA operation can be decomposed into three broad steps as follows:

Step 1: Key generation

The key setup is done once (rarely) when a user establishes (or replaces) their public key. Each user generates a public or private key pair by selecting two large primes at random as p,q. The computation of system modulo using prime variable p and q as, N=p.q. The public encryption key is given as KU={e,N} and private encryption key is given as KR= {d,p,q}.

Step 2: Encryption

To encrypt a message M the sender obtains a public key of recipient KU={e,N}. The ciphertext can be computed as, $C = M^e \bmod N$, where $0 \leq M < N$.

Step 3: Decryption

To decrypt the message ciphertext C obtains the private key KR={d,p,q}. The message can be computed as, $M = C^d \bmod N$.

6.2.1.4 Digital Signature Message Digest Algorithm

A digital signature is computed using a set of rules and parameters that allow the identity of the signatory for verification. The proposed framework implements the message digest algorithm as follows.

MD5 is a widely used cryptographic hash function with a 128-bit hash value that processes a variable-length message into a fixed-length output of 128 bits. The sender uses the public key of the receiver to encrypt the message and the receiver uses its private key to decrypt the message. The following five steps are performed to compute the message digest of the message.

Step 1. Append Padding Bits

The message is "padded" so that its length is congruent to 448, modulo 512. Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

Step 2. Append Length

A 64-bit representation of 'b' is appended to the result of the previous step. In the unlikely event that 'b' is greater than 2^{64}, the low-order 64 bits of 'b' are used. At this point the resulting message has a length that is an exact multiple of 512 bits. Equivalently, this message has a length which is an exact multiple of 16 (32-bit) words. Let M[0 ... N-1] denote the words of the resulting message, where 'N' is a multiple of 16.

Step 3. Initialize MD Buffer

A four-word buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first.

Step 4. Process Message in 16-Word Blocks.

It defines four auxiliary functions which takes input as three 32-bit words and produce output as one 32-bit word.

F(X,Y,Z) = XY v not(X) Z

G(X,Y,Z) = XZ v Y not(Z)

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X v not(Z))

Step 5. The message digest produced as output is A, B, C, D. It begins with the low-order byte of A, and ends with the high-order byte of D.

6.2.2 MSAES Decryption for Downloading Process

The downloading process of the encrypted mobile device data from cloud storage is discussed as follows:

Step 1: Suppose MD wants to fetch F with the name FN, MD then sends H(FN) to CS. CS searches in {F′,H(FN),MAC} sends back {F′ ||MAC} that matches H(FN) to MD.

Step 2: MD prompts for asking U to input corresponding PWD for the FN.

Step 3: MD generates encryption key EK= H(PWD || FN) and integrity key0

IK = H(FN || PWD).

Step 4: MD decrypts out F = DEC(F′,EK), and checks whether MAC = H(F, IK)

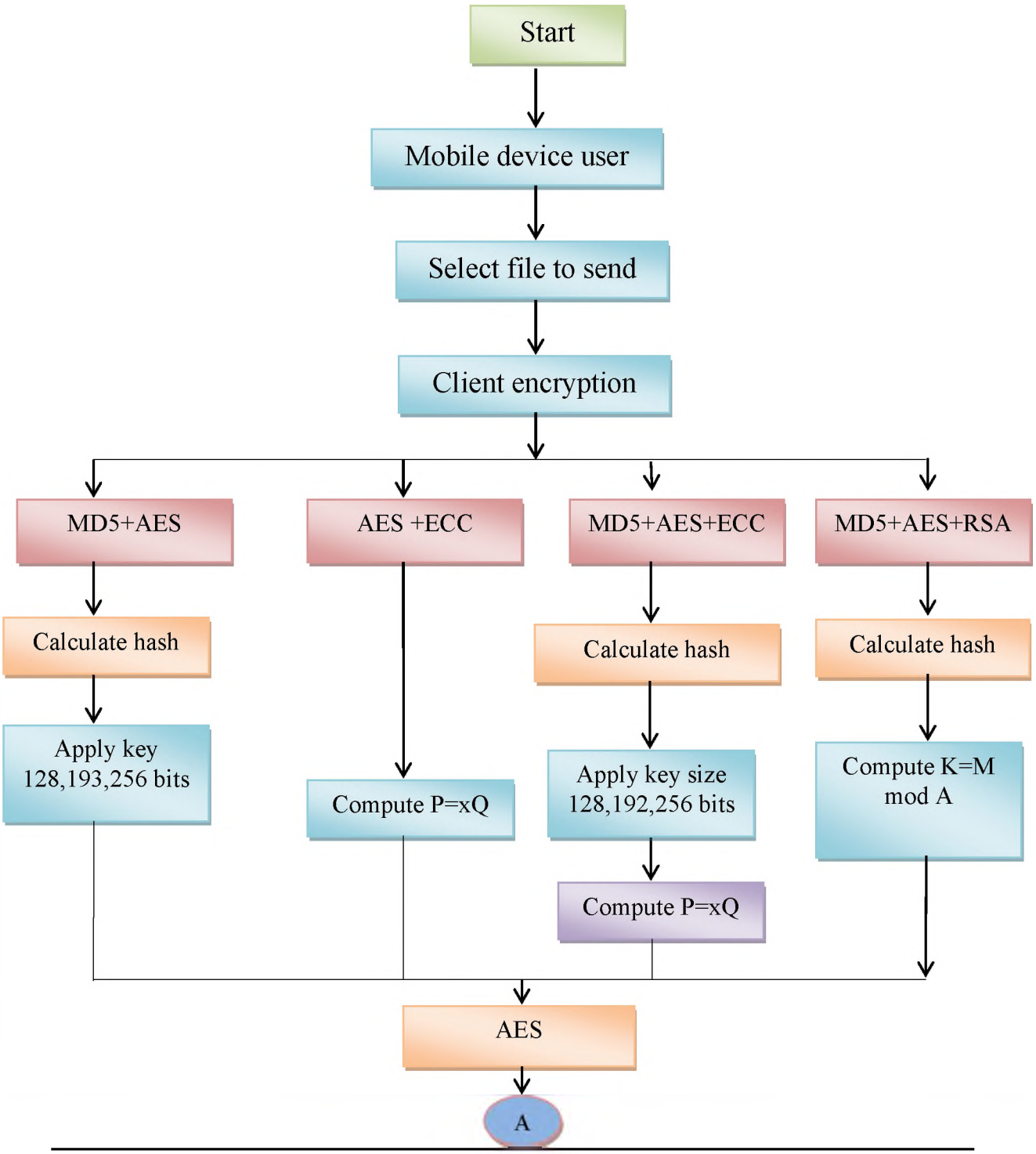
The decryption process of obtaining encrypted outsourced mobile data is decrypted by getting key from the user.

The above discussed cryptographic algorithms in section 6.2.1 is viceversed for the decryption process. The decryption of MSAES algorithm processed as shown in figure 6.3.

Figure 6.3 MSAES Decryption for Downloading Process

6.3 Flow diagram of the Proposed Contribution Three -MSAES Method

The proposed MSAES method secures the outsourced mobile device data over cloud storage using hybrid cryptographic algorithm. The flow diagram of the proposed contribution three MSAES method shown in figure.6.4.



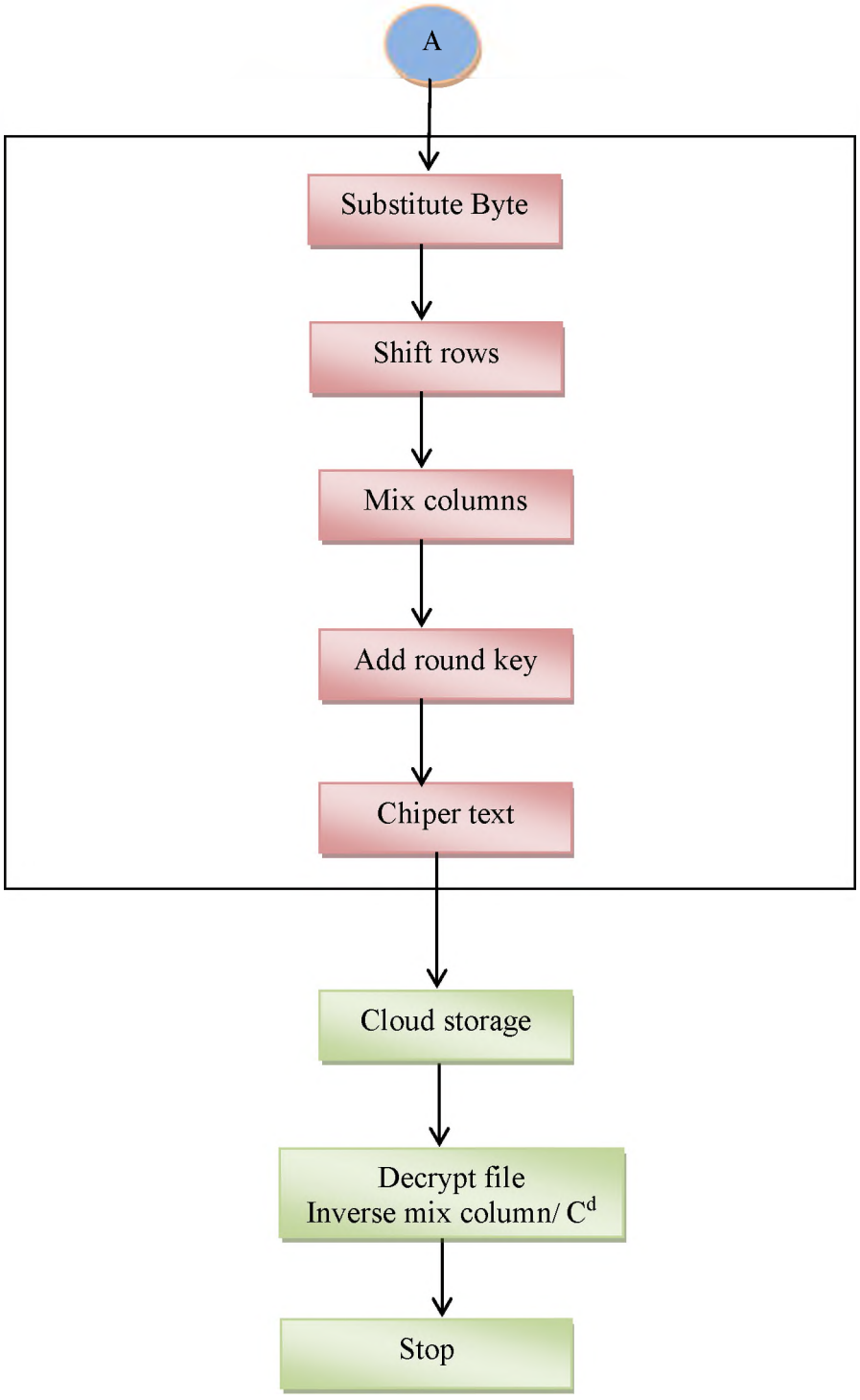


Figure 6.4 Flow chart of Proposed MSAES Method

6.4 Steps involved in the Proposed MSAES Method

The steps used for the secure data over cloud using MSAES algorithm is discussed below:

Step 1: *Create input data samples and select the file for processing*

Step 2: *Run hybrid approach of MSAES method by selecting the files*

Step 3: *Make cloud device instance on application tool and make dynamic web project*

Step 4: *Apply hybrid MSAES algorithm on cloud device with ECC. Actual data is encrypted with MD5 algorithm. The encrypted file is further encrypted with AES and then with ECC ensuring 3 levels of encryption. To calculate the public key in ECC,*

$$P=xQ$$

Step 5: *Apply hybrid MSAES algorithm on cloud device with RSA, the generated AES key is encrypted using RSA rather than encrypting the actual data. The encrypted key is calculated by,*

$$Y=EK(X)$$

where E- Encryption, K- Key, X- plain text, Y-cipher text

Step 6: *Apply decryption to retrieve the file encrypted and compare the results obtained.*

The six steps discussed above are followed to store the mobile device data securely over cloud storage using MSAES approach. The algorithm proposed is discussed in the next section.

6.5 Pseudo Code of MSAES Method

The MSAES method for uploading and downloading process of the outsourced mobile device data over cloud storage is shown in table 6.1

Table 6.1 Pseudo code of MSAES Method

```
Input
array in  of 4*Nb bytes // input plaintext
      array out of 4*Nb bytes // output ciphertext
      array w of 4*Nb*(Nr+1) bytes // expanded key
void Cipher(byte[] in, byte[] out, byte[] w) {
  byte[][] state = new byte[4][Nb];
  state = in; // actual component-wise copy
  AddRoundKey(state, w, 0, Nb - 1);
  for (int round = 1; round < Nr; round++)
 {
    SubBytes(state);
    ShiftRows(state);
    MixColumns(state);
    AddRoundKey(state, w, round*Nb, (round+1)*Nb - 1); // 16 Rounds
{
Four possible functions
A, B, C, D: initialized buffer words
  F(X,Y,Z) = (X AND Y ) OR (NOT X AND Z)
  G(X,Y,Z) = (X AND Z ) OR (Y AND NOT Z)
  H(X,Y,Z) = X XOR Y XOR Z
  I(X,Y,Z) = Y XOR (X OR NOT Z)
for k = 1 to N do the following
    AA = A
    BB = B
    CC = C
    DD = D
    (X[0], X[1], ..., X[15]) = M[k] /* Divide M[k] into 16 words */
      Round 1. Do 16 operations
      Round 2. Do 16 operations
      Round 3. Do 16 operations
      Round 4. Do 16 operations
end for

}}
  SubBytes(state);
  ShiftRows(state);
  AddRoundKey(state, w, Nr*Nb, (Nr+1)*Nb - 1);
  out = state;
}
```

6.6 Experimental Setup and Results

In the experimentation, the selection of data for encryption over cloud is deployed by an android application. The data samples may have varying sizes as 25kb, 50kb, 75kb, 100kb, 125kb and 150 kb. The hybrid cryptographic algorithm MSAES is used for encryption and decryption. For avoiding the security breaches during transit of data from mobile to cloud over network, the proposed method conducts two types of evaluation in both cloud and local environments. The experimentation methodology is shown in figure 6.5.

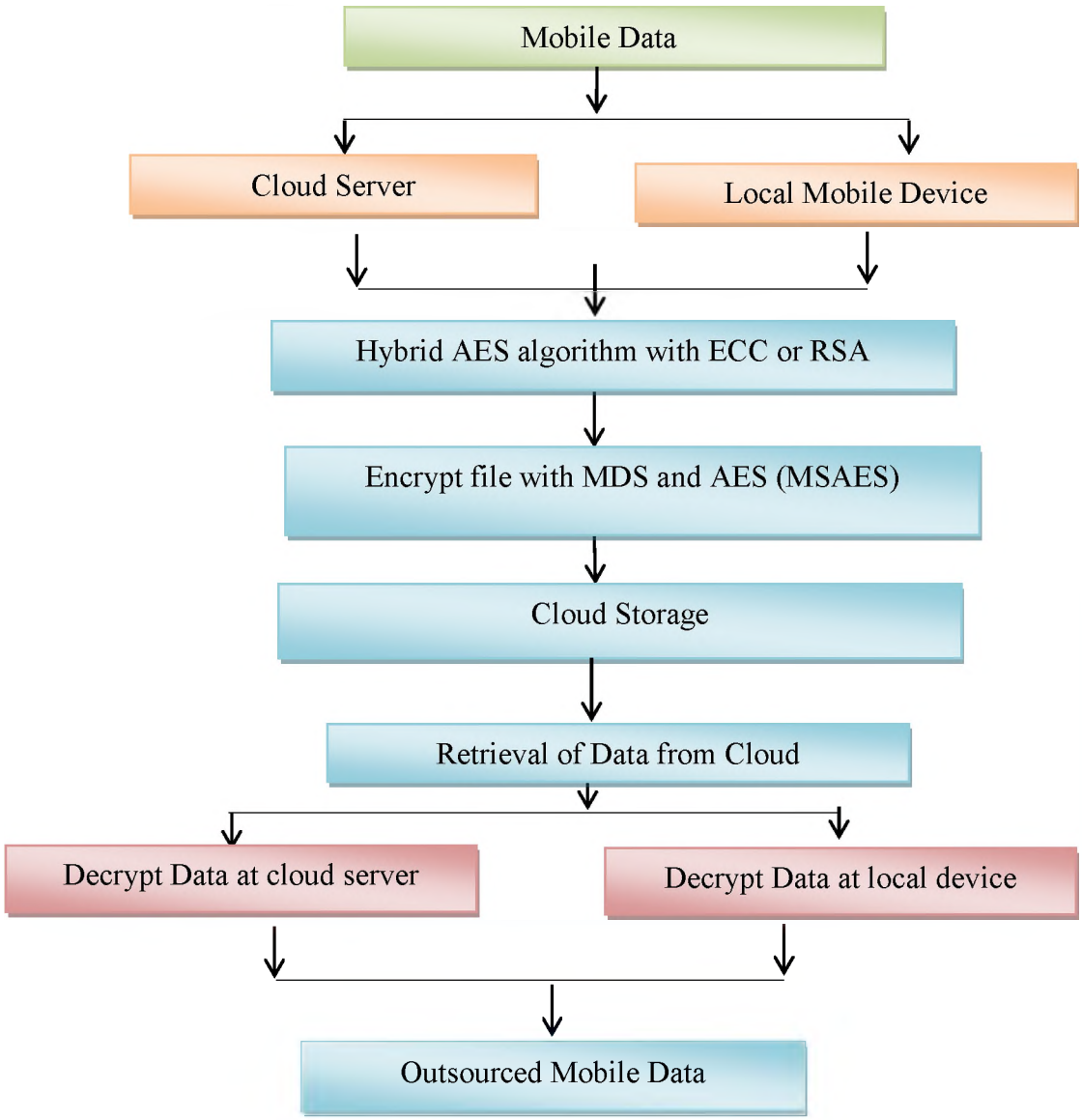


Figure 6.5 Experimentation Methodology for Contribution MSAES Method

For experimentation, the data files like text and image files are downloaded from the mobile device and are stored. The programs that are used for experimentation use two different file formats namely, text files and image files. A sample dataset is shown in Annexure III. To evaluate the performance of the proposed method, the following four parameters listed below are used.

i. Mean Processing Time
ii. Speed Up Ratio
iii. Turn Around Time
iv. Throughput

i. Mean Processing Time

Mean processing time is the difference between the starting time taken to encrypt the data and the ending time. It is also evaluated both on single device and on cloud environment. The MPT is defined by equation 6.1,

$$MPT = End\ time\ to\ encrypt - start\ time\ to\ encrypt \quad (6.1)$$

ii. Speed Up Ratio

It is defined as the difference between the mean processing time of single system and the cloud environment. The speed up ratio is defined by equation 6.2,

$$speed\ up\ ratio = \frac{Mean\ processing\ time\ on\ local\ device}{Mean\ processing\ time\ on\ cloud\ server} \quad (6.2)$$

iii. Turn Around Time

It is defined as the total time taken between starting encryption and time to upload the data as given in equation 6.3

$$TAT = \sum(Time\ of\ starting\ Encryption + Time\ to\ upload) \quad (6.3)$$

iv. Throughput

It is defined as the rate of the input size that can be processed over a certain execution time and shown in equation 6.4

$$Throughput = \frac{Input\ Size(MB)}{Execution\ Time(sec)} \quad (6.4)$$

The performance comparison of the parameters over cloud environment is shown in table 6.2.

Table 6.2: Performance Comparison of Hybrid Approach in Cloud

Algorithms	Size (kB)	Mean Processing Time (ms)	Speed Up Ratio (ms)	Throughput (kB/ms)	Turn Around Time (ms)
ECC+AES	25	147.8	562.2	38.1	2.211
	50	151.2	794.7	117.9	3.66
	75	256.4	951.4	198.4	4.03
	100	328.5	1160.5	265.7	4.06
	125	669.6	1508.2	402.1	5.30
	150	808.6	1808	449.8	4.85
MD5+AES	25	**22.8**	**226.4**	**3.6**	**0.85**
	50	**76.6**	**347.1**	**3.3**	**2.175**
	75	**131**	**349.1**	**5.5**	**2.363**
	100	**163.2**	**492.2**	**3.4**	**2.411**
	125	**408.2**	**763.6**	**4.2**	**3.212**
	150	**490.5**	**976.1**	**4.8**	**2.735**
MD5+ ECC + AES	25	28.5	246.1	18.7	1.078
	50	87.9	409.3	29.6	2.400
	75	157.7	414.2	54.3	2.719
	100	185.8	620.6	59.1	2.636
	125	555.7	983.1	263.7	4.329
	150	702.4	1250.3	355.0	4.148
MD5+RSA + AES	25	20.9	225.4	17.8	0.779
	50	71.3	348.5	18.5	2.068
	75	131.9	344.0	18.5	2.376
	100	150.9	497.7	18.5	2.288
	125	442.0	703.9	27	3.482
	150	561.2	864.7	18	3.207

Table 6.2 provides the comparison of parameters between MD5, ECC, AES and RSA of hybrid approaches. Figure 6.6 gives the comparison of hybrid approaches of cryptographic algorithms for mean time processing in cloud storage as follows.

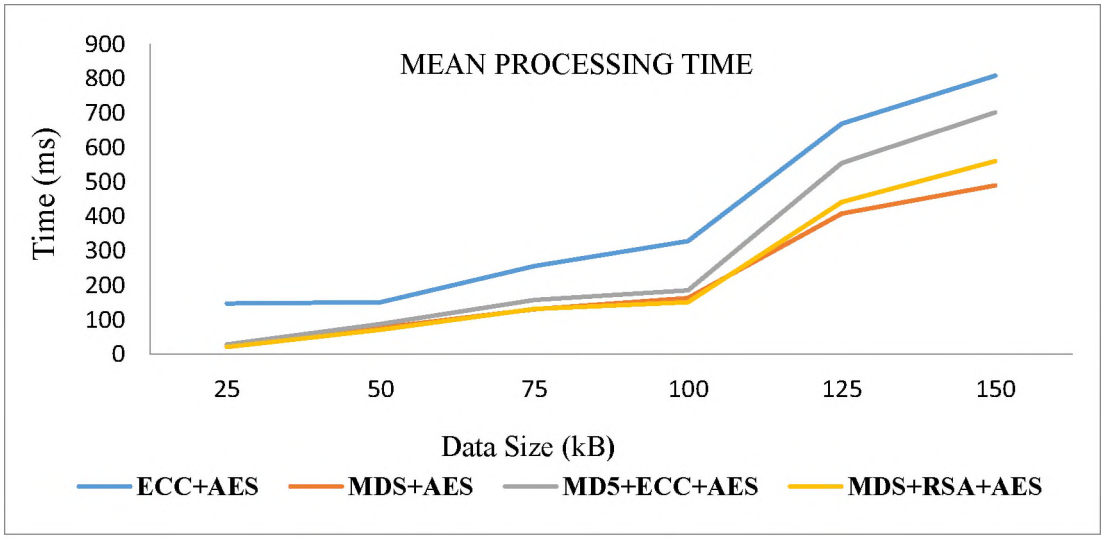


Figure 6.6 Mean Time Processing over Cloud

In cloud environment, as the size of input increases the time taken to encrypt the data will increase and with the increase in time, the mean processing time is decreased. Figure 6.7 gives the comparison of hybrid approaches of cryptographic algorithms for speed up ratio in cloud storage as follows.

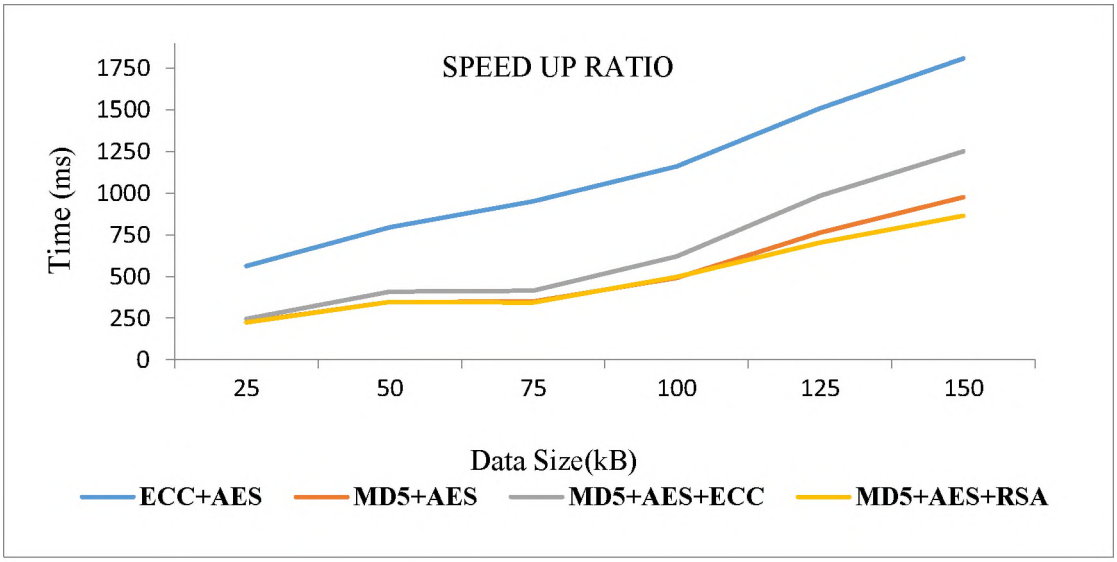


Figure 6.7 Speed Up Ratio over Cloud

In cloud environment, as the size of input increases, the mean processing time also increases which tends the speed up ratio to decrease. Figure 6.8 gives the comparison of hybrid approaches of cryptographic algorithms for throughput over cloud storage as follows.

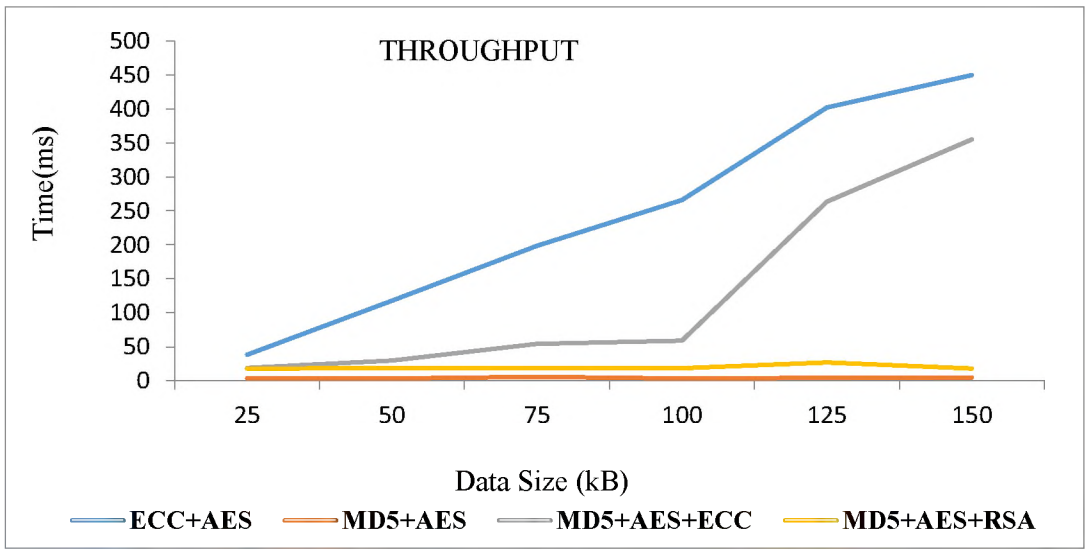


Figure 6.8 Throughput over Cloud

In cloud environment, as the size of input the execution time increases the throughput remains constant. Figure 6.9 gives the comparison of hybrid approaches of cryptographic algorithms for turnaround time over cloud storage as follows.

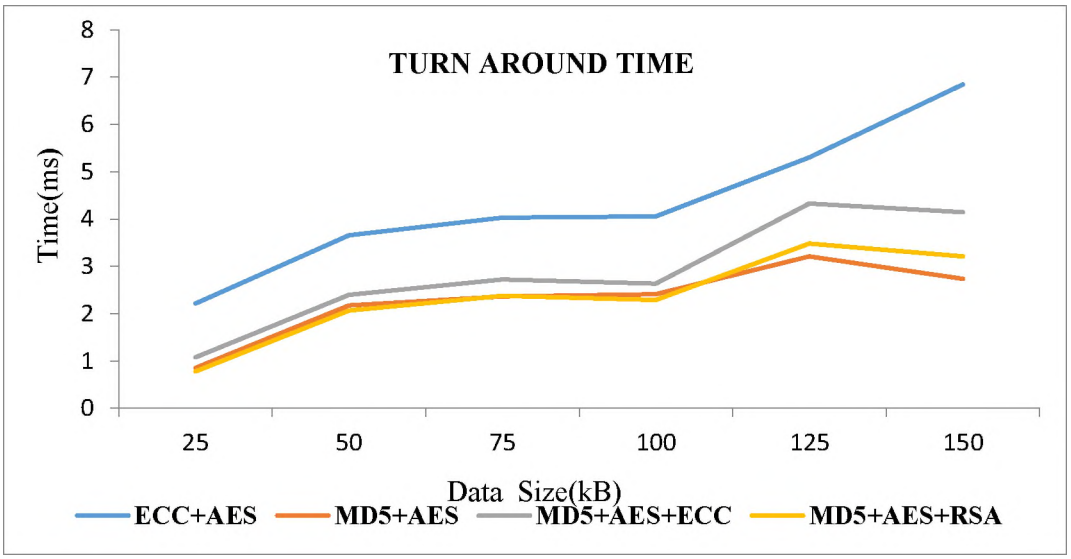


Figure 6.9 Turnaround Time over Cloud

In cloud environment, as the size of input increases, total time to upload and encryption is also increases, which tends to decrease in turnaround time. The performance comparison of the parameters over local environment is shown in table 6.3.

Table 6.3: Performance Comparison of Hybrid Approach in Local Device

Algorithms	Size (kB)	Mean Processing Time (ms)	Throughput (kB/ms)	Turn Around Time (ms)
ECC+AES	25	615.9	370	31.15
	50	942.6	522.5	22.67
	75	1204.4	728.2	18.61
	100	1485.5	834.3	16.72
	125	2176.0	1006.8	17.7
	150	2614.1	1150.9	17.15
MD5+AES	25	**246.1**	**3.1**	**16.36**
	50	**420.4**	**6.2**	**12.23**
	75	**476.7**	**9.0**	**8.91**
	100	**651.9**	**12.2**	**8.38**
	125	**1170**	**15.1**	**9.66**
	150	**1464.2**	**17.8**	**9.48**
MD5+ ECC + AES	25	271.5	57.33	17.37
	50	493.9	154.7	13.7
	75	568.5	191.4	10.13
	100	802.9	312.6	9.89
	125	1537.1	763.0	12.59
	150	1950.4	1016.8	12.73
MD5+RSA + AES	25	243.3	36.8	16.24
	50	416.5	35.7	12.15
	75	472.5	34.2	8.85
	100	645.1	34.0	8.31
	125	1144.2	26.0	9.45
	150	1423.5	24.8	9.21

Table 6.3 provides the comparison of parameters between MD5, ECC, AES and RSA of hybrid approaches. Figure 6.10 gives the comparison of hybrid approaches of cryptographic algorithms for mean time processing over local device as follows.

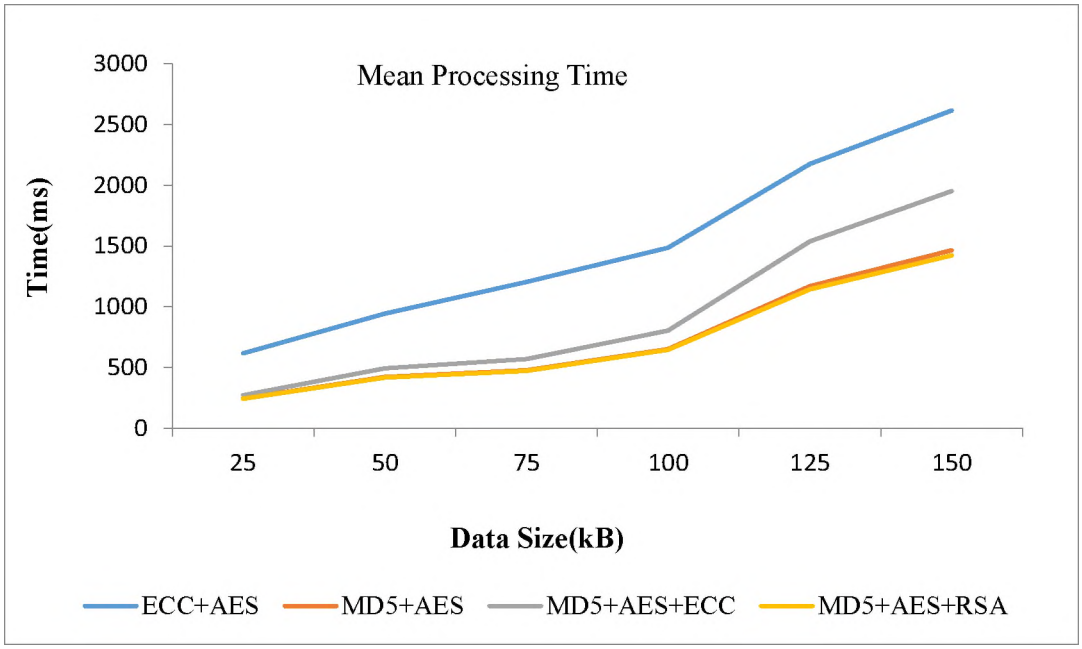


Figure 6.10 Mean Processing Time over Local Device

In local environment, as the size of input increases the time taken to encrypt the data will increase and with the increase in time, the mean processing time is decreased. Figure 6.11 gives the comparison of hybrid approaches of cryptographic algorithms for throughput over local device as follows.

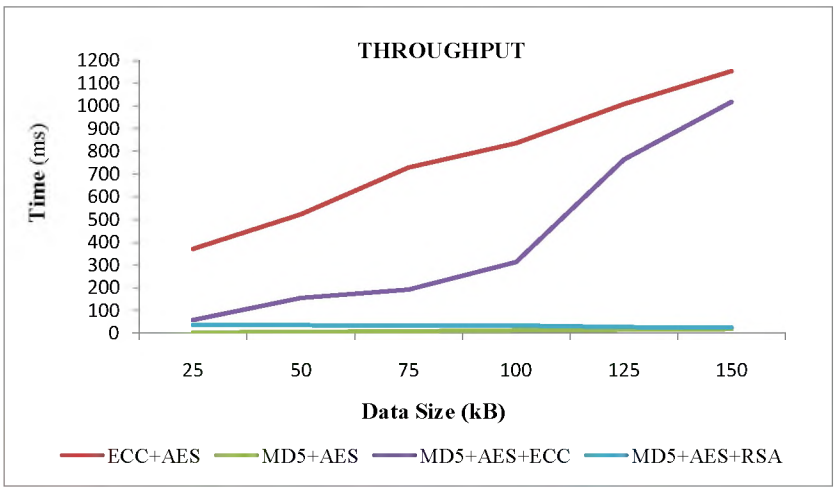


Figure 6.11 Throughput over Local Device

In local environment, as the size of input and execution time increases the throughput decreases. Figure 6.12 gives the comparison of hybrid approaches of cryptographic algorithms for turnaround time over local device as follows.

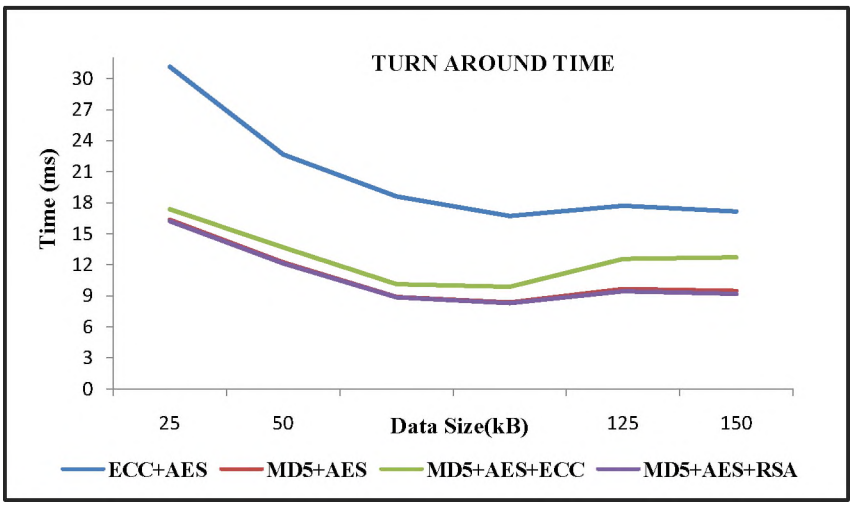


Figure 6.12 Turnaround Time over Local Device

In local environment, as the size of input increases, total time to upload and encryption is also increases, which tends to decrease in turnaround time. From the experimental result, the proposed MSAES approach using MD5 with Advanced Encryption Standard Algorithm has better efficiency in terms of mean processing time, throughput and turnaround time over cloud storage.

6.7 Chapter Summary

In this chapter, the proposed method MSAES for protecting the confidentiality and integrity of uploaded mobile device data over cloud storage is discussed. In secure data transfer to cloud, the hybrid cryptographic approach MSAES shows high efficiency when compared with other algorithms based on the mean processing time, throughput, speed up ratio and turnaround time. The proposed hybrid cryptographic approaches ensure the security of the outsourced mobile device data. For efficient retrieval of outsourced encrypted data, a secure ranked fuzzy multi-keyword searching technique is proposed in Contribution 4 and is explained in chapter 7.