International Journal of Security and Its Applications
Vol. 10, No. 8 (2016), pp.59-92
http://dx.doi.org/10.14257/ijsia.2016.10.8.07

Analysis of Various Cryptography Techniques: A Survey

Neha Tayal[1], Ritesh Bansal[2], Shailender Gupta[3] and Sangeeta Dhall[4]

[1,2,3,4] *YMCA University of Science and Technology, Faridabad*
[1]*nehatayal2292@gmail.com*, [2]*ritesh.bansal@hotmail.com*,
[3]*shailender81@gmail.com*, [4] *Sangeeta_dhall@yahoo.co.in*

Abstract

Maintaining the confidentiality of data during communication has always been a prime concern of many researchers. Several encryption mechanisms have been developed in order to protect the secret data from the access of unauthorized users. Encryption can be thought of as a set of instruction used for conversion of data from a readable state to nonsense form. An encryption scheme is said to be effective if it provides high security, low computational time and high brute force search time for hackers. This paper is an effort to compare all the text based encryption schemes mentioned in literature. These schemes are implemented in MATLAB-2010 and their efficacy is compared based on various performance metrics such as time complexity, Correlation, Key sensitivity analysis, Differential attack analysis and Entropy. These results can be fruitful for researchers working in this direction.

Keywords*: AES, Diffusion, Differential attacks, DES, Encryption/decryption, Logarithmic map, Statistical attacks*

1. Introduction

Cryptography is the usage of ciphers and coded language to protect secret data from the access of unauthorized users. But with the development in cryptography the cryptanalysis (breaking of codes and ciphers) has also developed in parallel with same pace. So with the advancement in technology, there is an evident need of more secure cryptographic schemes to protect the secret data. Therefore, researchers have to continuously work towards improving and finding new encryption algorithms for securing data. Cryptographic techniques can be broadly classified in two categories: Asymmetric key and Symmetric Key Cryptography (see Figure 1).

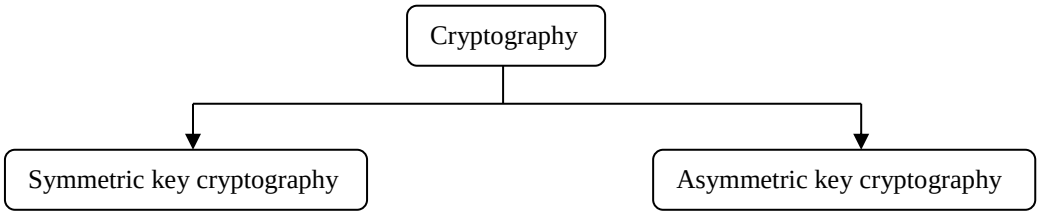

Figure 1. Classification of Cryptography

In asymmetric key cryptography, public key of the destination is used to encrypt the data which is to be transmitted, and the receiver uses its private key to decrypt data. While in case of symmetric key cryptography same key is used for encryption and decryption. Schemes base on later are considered to be fast as compare to one based on former while providing less security in comparison. These schemes employs a process called Encryption, aim of which is to transform secret data into a form which is redundant or appear to be noise to the unauthorized user. Only authorized parties should be able to read this encoded messages or information. The reverse of this process, is called Decryption. It

ISSN: 1738-9976 IJSIA

is logical reverse of encryption, where encoded message is converted back into its original form, which is understandable. Schemes from literature, are based on both Symmetric key [1-5,8-11,13-14,18-23,25-28] & Asymmetric key [6-7,12,15-16,17,24] and implemented in MATLAB. Following are the key areas for checking the efficacy of encryption scheme:

- **Large key space**: large key space helps in resisting brute force attacks.
- **Sensitive dependence on initial conditions**: Small changes in the initial values of keys should result in high degree change of enciphered output.
- **Less Time Complexity/High Speed**: Time taken by scheme for encryption should be very less.
- **Security**: There should be minimum effect of statistical, differential, brute force and several other attacks.
- **Diffusion** – Small change in input data should bring drastic changes in enciphered output
- **Confusion** -No or minimum Correlation between input data and encipher output.

In this paper, various cryptography schemes are compared by taking these performance measures into account. The rest of the paper is organized as: Section 2 provides the comparison between various implemented cryptography techniques. Section 3 gives the simulation set up parameters used to compare the said schemes. In Section 4, results are compared of all the mechanisms followed by conclusion and references. The next section gives the details of the all schemes.

2. Techniques Used

This section gives the complete detail of all the cryptographic schemes present in the literature.

2.1. Visual Cryptography

This scheme was first suggested by Adi Shamir in 1979[1]. It generates 2 shares (see Figure 2) where first share is made up of random numbers which are generated by using chaotic function and second share by using data and first share. Both the shares are necessary to generate the original secret data where a share can be defined as a component of data which contains partial information about it and hence it seems like a meaningless information.

Encryption Algorithm

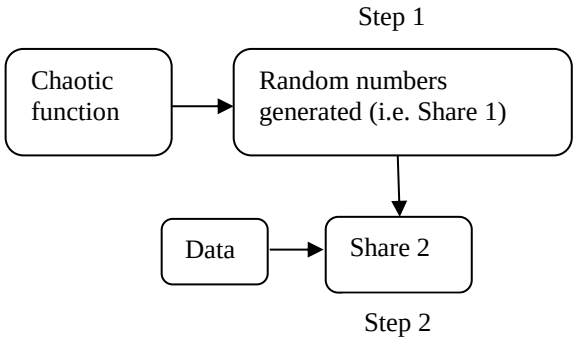

Figure 2. Visual Cryptography Encryption Scheme

1. Logistic chaotic map is used to generate the number numbers of the same length as of message which is called share1.

```
l=length(data);
bc1=.5;
u=3.6;
for i=2 to l
   bc1(i)=u*bc1(i-1)*(1-bc1(i-1));  /* logistic chaotic map */
end
share1=mod((floor(bc1*10^4)),2);
```

2. Then share2 is generated by using data bits and result obtained from step 1 as if data bit is 1,then compliment of bit of share1 is written into share2 else bit of share 1 is copied as it is into share2.

```
for(i=1 to l)
   if(data(i)==0)
     share2(i)=share1(i);
   else
        share2(i)=not(share1(i));
  end
end
```

Decryption Algorithm

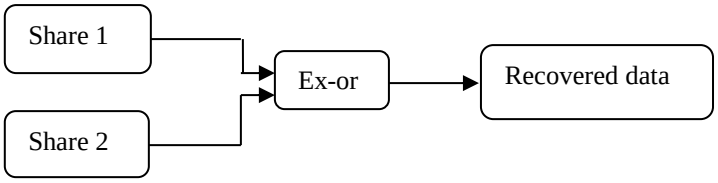

Figure 3. Visual Cryptography Decryption Scheme

Obtain both the shares and ex-or them to recover the original secret (see Figure 3).

```
for(i=1 to l)
       recovered_data(i)=xor(share1(i),share2(i));
end
```

Advantages:

1. Easy to implement.
2. Small time complexity.

Disadvantages:

1. Possibility of detection of message without applying decryption algorithm.
2. Low security as it is susceptible to attacks such as phishing attack.

2.2. Hierarchical Cryptography

This technique was proposed by Pallavi Vijay Chavan, Dr. Mohammad Atique and Dr. Latesh Malik in 2014[20]. It is the extension of visual cryptography. As the name suggests, it follows the hierarchical relationship between shares. First, two shares are

generated from secret data and these two generated shares are used to further generate two shares each (see Figure 4).

Encryption Algorithm

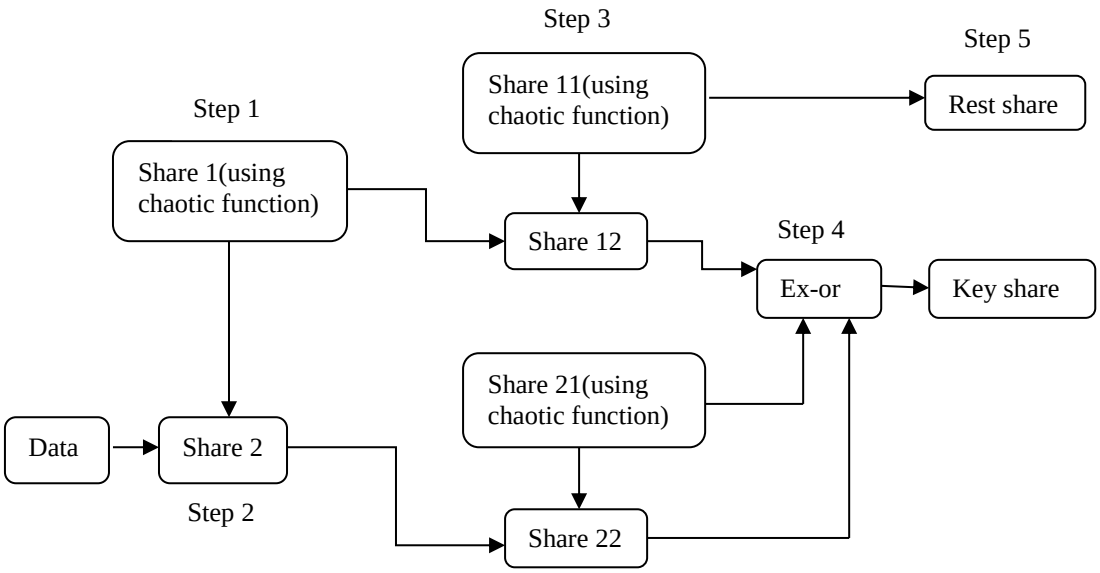

Figure 4. Hierarchical Cryptography Encryption Scheme

1. Share1 is made up of random numbers which are generated by using logistic chaotic function.

```
l=length(data);
bc1=.5;
u=3.6;
for i=2 to l
        bc1(i)=u*bc1(i-1)*(1-bc1(i-1));  /* logistic chaotic map */
end
share1=mod((floor(bc1*10^4)),2);
```

2. Then share2 is generated by using data bits and result obtained from step 1 as if data bit is 1,then compliment of bit of share1 is written into share2 else bit of share 1 is copied as it is into share2.

```
 for(i=1 to l)
        if(data(i)==0)
          share2(i)=share1(i);
        else
              share2(i)=not(share1(i));
       end
end
```

3. Similarly share11 and share12 are obtained from share1 and share21 and share22 are obtained from share2 by repeating the above process.

4. Then share12, share21 and share22 are ex-ored to obtain the key share.

5. Remaining share *i.e.*, share11 is known as rest share.

Decryption Algorithm

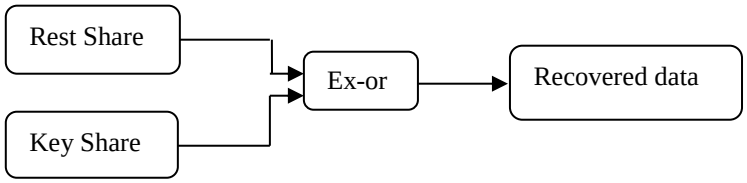

Figure 5. Hierarchical Cryptography Decryption Scheme

Obtain key share and rest share and ex-or them to originate the actual information (see Figure 5).

```
for(i=1 to l)
        recovered_data(i)=xor(share1(i),share2(i));
end
```

Advantages:

1. Encryption is performed at different levels due to hierarchy.
2. All generated shares give no information by visual inspection.
3. No graying effect as it increases the number of white pixels.

Disadvantages:

As the number of shares increases, this scheme becomes complex.

2.3. RC4

This scheme was designed by Ronald Rivest in 1987. In cryptography, RC4 (Rivest Cipher 4 also known as ARC4 or ARCFOUR meaning Alleged RC4) is a stream cipher [3,25]. In this scheme, variable length key ranging from 1 to 256 is used to generate the pseudorandom stream cipher (see Figure 6).

Encryption Algorithm

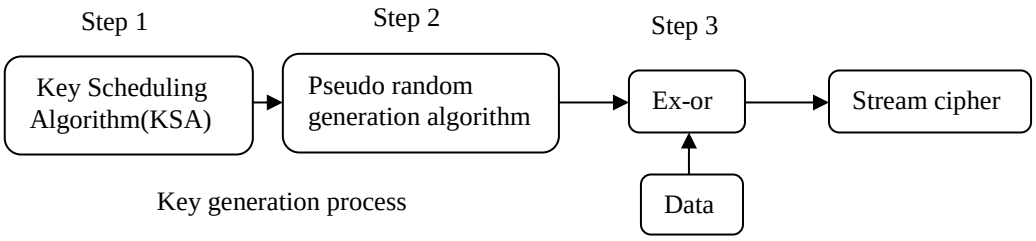

Figure 6. RC4 Encryption Scheme

1. Firstly, keys are generated using key scheduling algorithm (KSA) which is used to initialize the permutation in the array ‘s’ as:

```
for i=0 to 255
```

```
  key(i+1)=i;                  /* key initialization */
  s(i+1)=i;                    /* S is a state vector */
end
for(i=1:1:length(key))
   T(i)=key((mod(i-1,length(key)))+1);    /* T is a temporary vector */
end

j=0;
for(i=1 to 256)
   j=mod((j+s(i)+T(i)),256);
   swap s(i) and s(j)                     /* state vector initialization  using
permutation */
end
```

2. Next step is to generate the pseudo random keystream by using the state vector obtained in step 1. It uses pseudo random generation algorithm for this.

```
i=0;
j=0;
l=1;
while(l<=length(k))                /* k is keystream cipher */
    i=mod(i,256);
    j=(mod((j+s(i+1)),256));
   swap s(i) and s(j)
   t=mod((s(i+1)+s(j+1)),256);
   k(l)=s(t+1);
   l=l+1;
end
```

3. Now, perform bit-wise exclusive-or operation of data stream with keystream to generate the ciphertext.

```
data1=de2bi(data,8);            /* data1 is in binary form */
k1=de2bi(k(i1),8);              /* k1 is in binary form */
for(i=1 to length(data1))
   ciphertext(i)=xor(data1(i),k1(i));
end
```

Decryption Algorithm

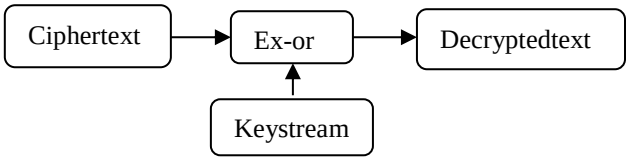

Figure 7. RC4 Decryption Scheme

Obtain ciphertext and keystream and perform ex-or operation to obtain the decryptedtext *i.e.,* plaintext (see Figure 7).

```
for(i=1 to length(ciphertext))
            decryptedtext(i)=xor(ciphertext(i),k1(i));
  end
```


Advantages:

1. As it's a stream cipher, so makes processing faster.
2. A particular RC4 key can be used only once.
3. Stream ciphers are immune to noise.
4. Applicable on data whose length is either unknown or continuous.

Disadvantages:

1. Encryption/decryption time is directly related to key length and data size i.e. as the size increases, speed goes down.
2. One in every 256 keys would be a weak key which are identified by crypanalysis that is able to find circumstances under which one of more generated bytes are strongly correlated with a few bytes of the key.
3. Stream ciphers provide no integrity protection or authentication.

2.4. Elliptic Curve Cryptography (ECC)

The use of elliptic curves was suggested by Neal Koblitz and Victor S. Miller in 1985 and this scheme was entered as an algorithm on 2005 to 2006. It is a public key cryptography in which two keys *i.e.,* one public and one private keys are used. Public key is used to encrypt the data and private key is used to decrypt the data which increases the security level. Here random numbers are used to generate the keys[7, 24]. This scheme is so called, because by representing data bits on elliptic curve, points are obtained which are used in generation of ciphers (see Figure 8).

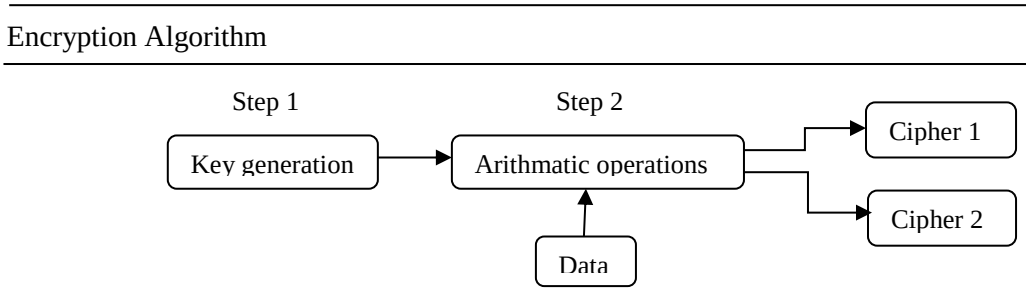

Figure 8. ECC Encryption Scheme

1. Within the range n, choose a number d which is called private key. Then generate public key Q by using the following equation

$$Q = d * P \quad (1)$$

where P is the point on the elliptic curve $y = x^2 + ax + \mathrm{b}$ (2)

/ n, a, b, d, x randomly generated no, with n, a & b having range 1 to 50, while d and x having 1 to n-1 and 0 to 15 respectively */*

```
y=x^2+a*x+b;
P=[x,y];
Q=d*P;
```

2. Represent message 'm' on the curve. Let m has a point M on the curve and generate two ciphers by using the below process:

```
for i=1 to length(message)
   x=m(i);                    /* m is in binary form */
   y=x^2+a*x+b;
```

```
  M=[x,y];
  k=randi([1,n-1]);
  Cipher1=k*P;
  Cipher2=M+k*Q;
end
```

Decryption Algorithm

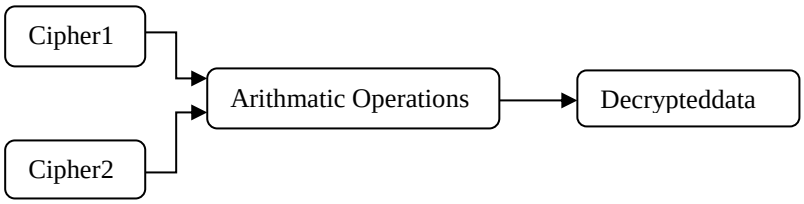

Figure 9. ECC Decryption Scheme

Obtain both the ciphers and perform the following operation to get the decrypteddata (see Figure 9).

```
for i=1 to length(decrypteddata)
decrypteddata(i)=Cipher2(i)-d*Cipher1(i);
end
```

Advantages:

1. More secure due to public key usage.
2. It uses smaller keys to provide high speed and security.
3. Due to small key size, it reduces the storage and transmission requirements.

Disadvantages:

Complicated and tricky to implement securely, particularly the standard curves as randon numbers are used to select the curve.

2.5. Vignere Table

This scheme was proposed by Chris Christensen in 2006. It is a symmetric key cryptography in which same key is used to encrypt and decrypt the message. It works on alphabetic data [4,21]. Table 1 shows the vignere table which is used in encryption (see Figure 10)/decryption (see Figure 11) process and is generated by a circular left shift in previous row.

Table 1. Vignere Table

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F

H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Encryption Algorithm

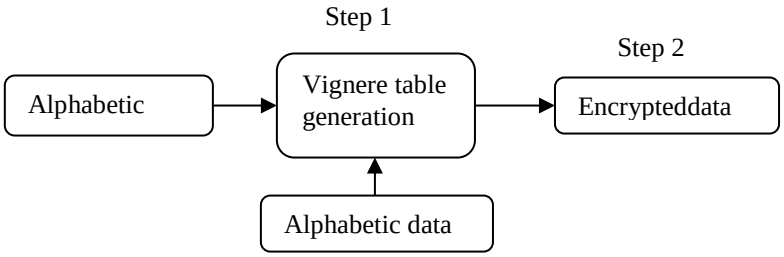

Figure 10. Vignere Encryption Scheme

1. Generate vignere table as follows:

```
p=1;
for(i=65 to 90)
   a(p)=char(i);
   p=p+1;
end
vignere_table(1,:)=a;
for(i=2 to 26)
    vignere_table(i,:)=shift left vignere_table(i-1,:) by 1;
end
```

2. Treat rows of vignere table as containing key and columns as containing data. Now, take the entry of corresponding row and column as the encrypteddata.

```
 for(i=1:1:length(data))
    encrypteddata(i)=vignere_table(key(i),data(i));
end
```

Decryption Algorithm

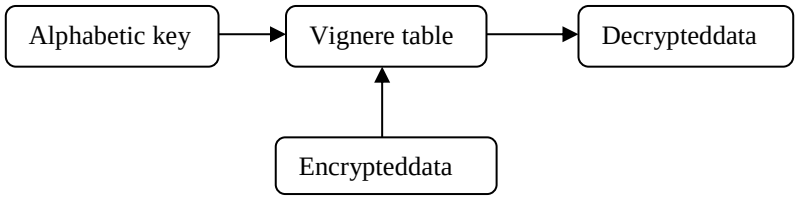

Figure 11. Vignere Decryption Scheme

Obtain the encrypteddata and search for each encrypted value in corresponding row containing corresponding key. The column number of that match is the decrypteddata.

```
for(i=1 to length(encrypteddata))
   k=key(i);
     for(j=1 to 26)
      if(vignere_table(k,j)==f(i))
         decrypteddata(i)=vignere_table(1,j);
         break;
       end
end
```

Advantages:

1. Easy to implement and understand.
2. Polyalphabetic substitution cipher which uses two or more cipher alphabets.

Disadvantages:

1. Primary weakness is repeating key *i.e.*, cryptanalyst can treat the cipher as Caesar cipher which can be broken.
2. It works only on alphabets.
3. It is a symmetric key cryptography, so less secure.

2.6. Diffie Hellman

This scheme was first published by Whitfield Diffie and Martin Hellman in 1976. It is a non-authenticated key agreement protocol and is used to provide forward secrecy in transport layer. In 2002, it was named as Diffie-Hellman-Merkle key exchange in recognition of Ralph Merkle's contribution to the invention of public key cryptography. This scheme uses random values to generate the key[6,15] which uses key exchange method for the same (see Figure 12).

Encryption Algorithm

1. Choose four random numbers p & q (known to both A & B), x (known only to A) and y (known only to B), all of which will be used in key generation by key exchange method as:

R1= p^x mod q; */* R1 generated by A */*
R2= p^y mod q; */* R2 generated by A */*
Key1= $(R2)^x$ mod q; */* key1 generated by A */*
Key2= $(R1)^y$ mod q; */* key2 generated by B */*
Where key1=key2

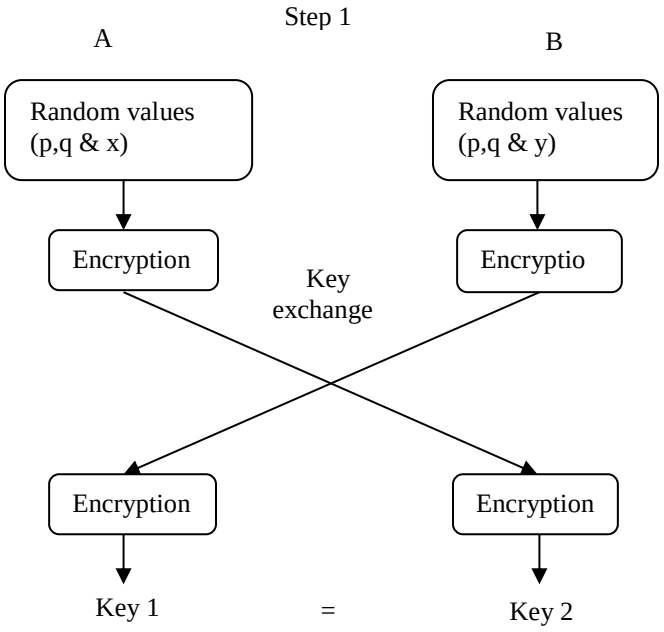

Figure 12. Key Exchange Process

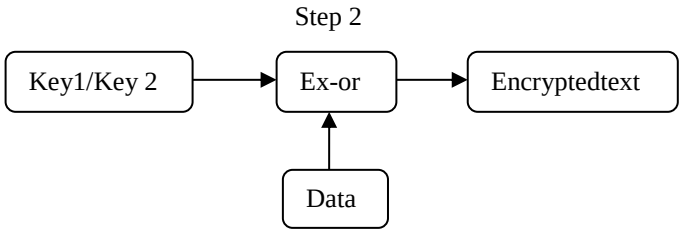

Figure 13. Diffie-Hellman Encryption Scheme

2. Now convert key and data in binary form and then perform exclusive-or operation to generate the ciphertext (see Figure 13) as:

```
e=1;
for i=1:length(data1)                    /* data1 is the data in binary form */
    ciphertext(i)=xor(data1(i),key(e);    /* key is the key1 in binary form */
    e=e+1;
    if e>length(key)
      e=1;
```

```
    end
end
```

Decryption Algorithm

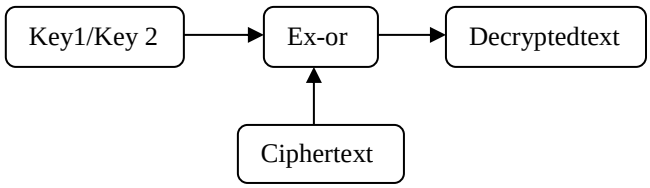

Figure 14. Diffie-Hellman Decryption Scheme

Obtain the ciphertext and perform the exclusive-or operation to get decryptedtext (see Figure 14) as:

```
e=1;
for i=1 to length(ciphertext)
  decryptedtext(i)=xor(ciphertext(i),key(e);      /* key is the key1 in binary form */
  e=e+1;
  if e>length(key)
    e=1;
  end
end
```

Advantages:

1. As key is generated by key exchange method, so increases security.
2. Random values are used in key generation process.

Disadvantages:

1. If large random numbers are used, speed becomes slow because of complex calculations.

2. Susceptible to Logjam attack which allows a man in the middle attacker to read and modify any data passed over the connection.

2.7. RSA

RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977. It is a public key cryptography in which two different keys are used, one *i.e.*, public key for encryption (see Figure 15) and another *i.e.*, private key for decryption [12,16] (see Figure 16).

Encryption Algorithm

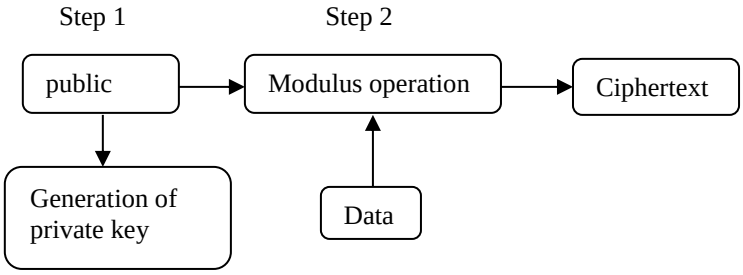

Figure 15. RSA Encryption Scheme

1. Two random prime numbers p & q are chosen and then calculate the following:

   *n=p*q;*
   a=(p-1)(q-1);*

2. Choose another random prime number e which is a public key to find the private key *i.e.*, d and then generate the ciphertext as:

   *d*e=1 mod a;*
   ciphertext=$(data)^{e}$ *mod n;*

Decryption Algorithm

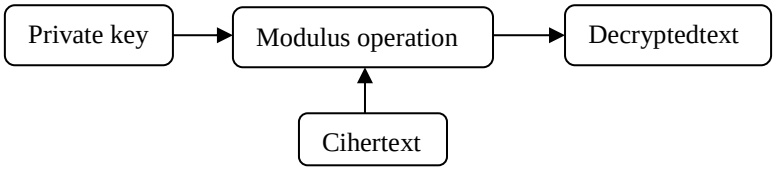

Figure 16. RSA Decryption Scheme

Obtain the ciphertext and then perform the following operation to get the decryptedtext as:

decryptedtext=$(ciphertext)^{d}$ *mod n;*

Advantages:

1. Since it is a public key cryptography, so more secure compared to symmetric key cryptography techniques.
2. Random numbers used in the process make this algorithm more complex.

Disadvantages:

If large random numbers are used, speed becomes slower than Diffie Hellman key exchange scheme due to weighty calculations.

2.8. DES

Data Encryption Standard (DES) was first developed by IBM in early 1970s and was published as an official Federal Information Processing Standard (FIPS) for the United States in 1977. It is a symmetric key cryptography which processes 64 bits block at a time

and key size is also 64 bit in which 56 bits are actually used by the algorithm and rest 8 bits are parity bits[10,27]. Data is processed for 16 rounds (see Figure 17). The structure used is Feistel structure (see Figure 18).

Encryption Algorithm

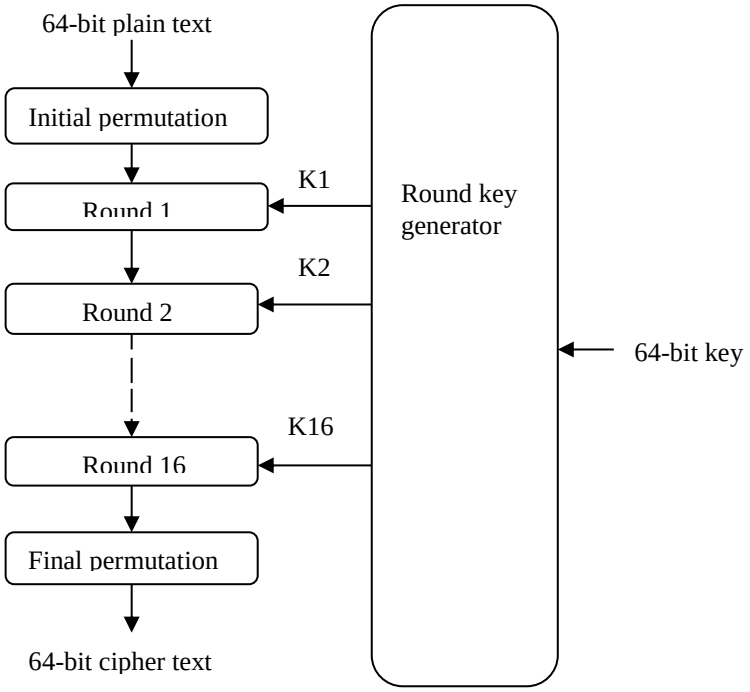

Figure 17. DES Encryption Scheme

1. 64-bit key is used to generate 16 48-bit keys one for each round first by compression permutation to 56-bits and then dividing the obtained key into left and right halves *i.e.,* 32-bit C0 & D0 and then create 16 blocks C_n and D_n where each pair of C_n and D_n is formed from the previous pair $C_{(n-1)}$ and $D_{(n-1)}$ by circular left shift by 1 (for rounds 1, 2, 9 and 16) and by 2 (for rest rounds) operation.

```
    p=1;c=2;
for(i=1 to 8)
   for(j=1 to 7)
     k_56(p)=K2(pc1(i,j));/* pc1 is 8x7 matrix used for compression permutation */
      p=p+1;
   end
end
C0=left half of k_56;
D0=right half of k_56;
for i=1 to 16
   C(i)=circular left shift C(i-1);
   D(i)= circular left shift D(i-1);
end
```


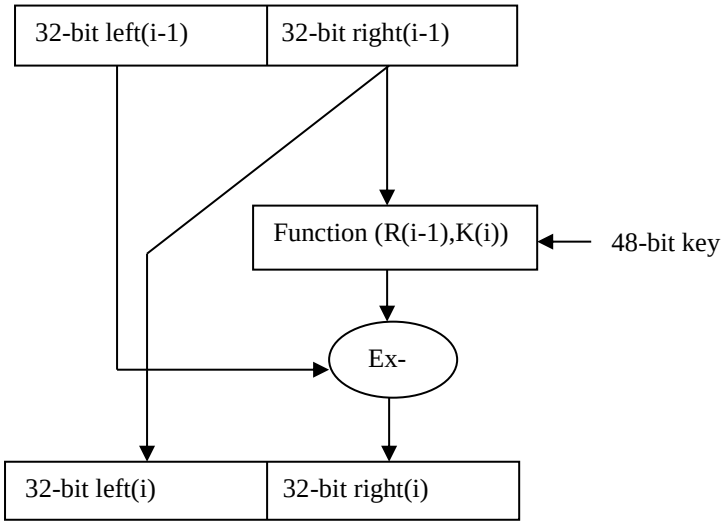

Figure 18. Encryption Round

2. Then concatenate pairs C_n D_n and apply **compression permutation** to generate 48-bit sixteen keys one for each round as:

```
for k=1 to 16
    p=1;
   for i=1 to 8
      for j=1 to 6
         k_48(k,p)=key(k,(pc2(i,j))); /* concatenated pairs are in key and pc2 is 8x6 matrix used for compression permutation */
         p=p+1;
      end
   end
end
```

3. Now, apply initial **permutation** on 64-bit data and divide into left and right halves L0 & R0.

```
p=1;
for i=1 to 8
   for j=1 to 8
      data1(p)=data(ip(i,j)); /* ip is 8x8 initial permutation matrix */
      p=p+1;
   end
end
L0= left half of data1;
R0=right half of data1;
```

4. Repeat steps 5 to 7 for 16 rounds in which result of previous round of right part becomes the left part for current round.

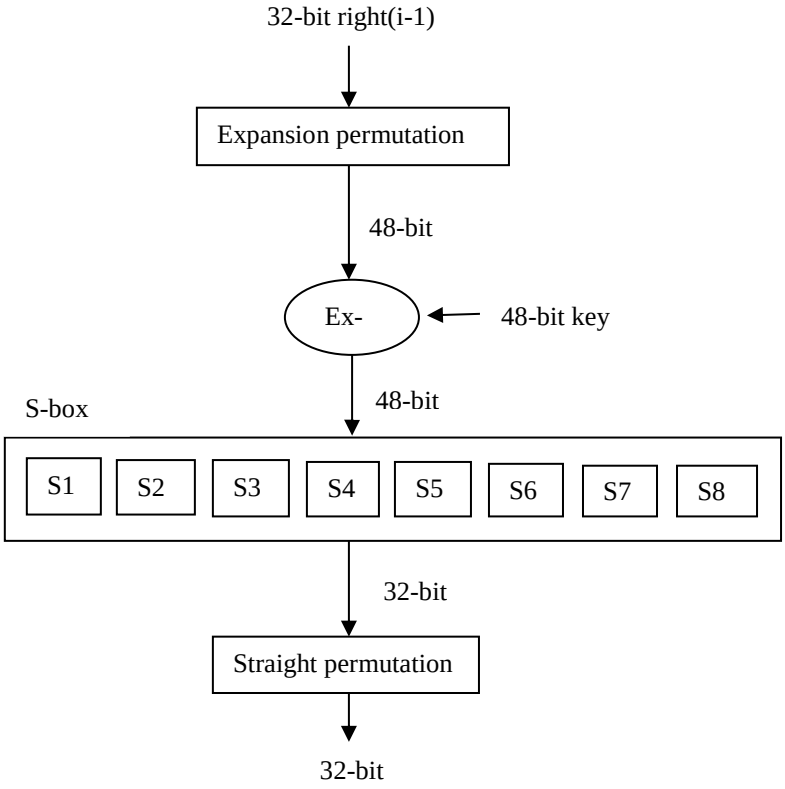

Figure 19. Function Process

5. R0 is then passed into function. In function (see Figure 19), first **expansion permutation** is applied to make R0 of 48-bit. Then it is **xor-ed** with 48-bit key corresponding to a round.

```
p=1;
for i=1 to 8
    for j=1 to 6
        R0_48(p)=R0(pc3(i,j)); /* pc3 is 8x6 matrix used for compression permutation */
        p=p+1;
    end
end
for i=1 to 48
   R_xor(i)=xor(R0_48(i),k_48(1,i));
end
```

6. This result is now passed through s-box which convert the data again into 32-bit by dividing 48-bit data into eight 6-bit blocks. Then is concatenates first and last bit of each block which defines the row number and middle four bits defines the column number of an entry in the s-corresponding **s-box** for that block which is then converted into binary form of 4-bit as:

```
h=1;
for i=1 to 48
    row(p)=R0_xor(i);
    row(p+1)=R0_xor(i+5);
    col(p)=R0_xor(i+1);
    col(p+1)=R0_xor(i+2);
```



```
        col(p+2)=R0_xor(i+3);
        col(p+3)=R0_xor(i+4);
        row1=bin2dec(row);          /* row1 defines the row number of s-box */
        col1=bin2dec(col);          /* col1 defines the column number of s-box */
        for j=1 to 4
           for k=1 to 16
              if((j-1)==row1 && (k-1)==col1)
                s_data(h to h+3)=dec2bin(s_box(j,k),4);   /* s_data contains 32-bit s-box
result */
                h=h+4;
              end
            end
         end
      end
```

7. Then apply **straight permutation** on above result and then ex-or it with the left 32-bit obtained in step 3 to get encrypted value for that right part of actual data

8. Now. interchange the position of right and left halves and concatenate as R16 L16 where R16 and L16 are the results after 16th round and then apply **final permutation** to obtain ciphertext.

Decryption algorithm

1. Obtain the ciphertext.

2. Apply the same process as in encryption from step 4 to 8 to decrypt the ciphertext. The only difference is that now the key is used in reverse order *i.e.*, key used in 16th round key for encryption will now be used in 1st round of decryption and so on (see Figure 20).

Advantages:

1. Completeness *i.e.*, each bit of cipher text depends upon multiple bits of plaintext.

2. Complex structure with 16 rounds make it more secure as complex operations are performed in it.

Disadvantages:

1. Because of Feistel structure, system becomes complex.

2. Due to usage of symmetric key, this algorithm is less secure.

3. Since it operates on blocks, so processing speed is less.

4. Block ciphers are more susceptible to noise in transmission.

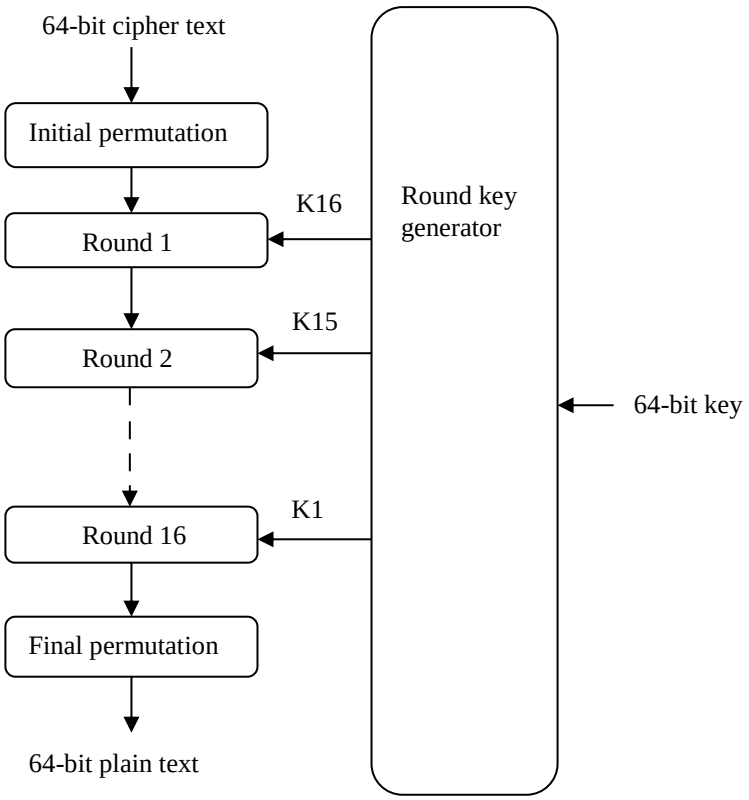

Figure 20. DES Decryption Round

2.9. IDEA

International Data Encryption Standard (IDEA) is a block cipher designed by Xuejia Lai and James L. Massey in 1991[2]. It is a symmetric key cryptography which operates on 64-bit data and the key length is 128 bits.

Key Scheduling for encryption

It divides 128-bit key into eight 16-bit blocks, the first 6 blocks are used for first round, while the remaining 2 are to be used for the second round. Then entire 128 bit key is given a rotation for 25 steps to the left and again divided into eight blocks. The first 4 sub-keys are used in 2nd round and the remaining in next round. This process continues till the end of the algorithm [26].

Below code shows the generation of key for the first round. Similarly, for all the 8 rounds, keys are generated.

```
subkeyrot(1,:)=initialkey;
 for i=2 to 13
    subkeyrot(i,:) = circshift(subkeyrot(i-1,:),[0 -25]);
 end
 for i=1:6
     subkey(i,:,1)=subkeyrot(1,16*i-15:16*i);   /* keys for round 1 */
 end
```


Encryption Algorithm

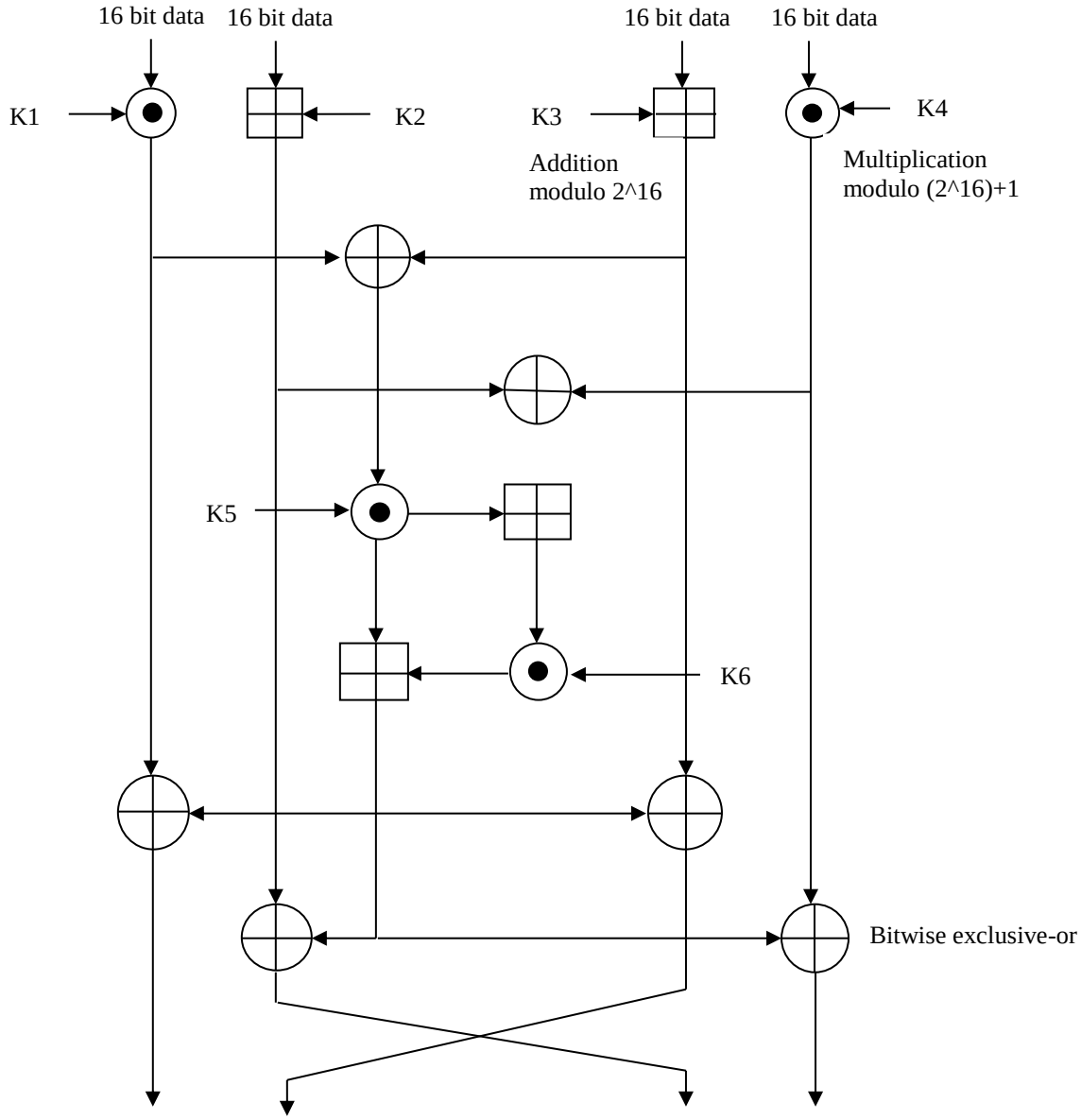

Figure 21. An Encryption Round

1. The 64-bit data is divided into four 16-bit blocks X1, X2, X3 and X4 (see Figure 21).

```
for i=1 to 4
   X(i,:)=data(1,16*i-15 to 16*i);
end
```

2. Repeat step 3 for 8 rounds. Here multiplication is modulo 2^16+1 and addition is modulo 2^16.

3. Perform the following operations for each round:

*X1=mod(X1*subkey(1,round),2^16+1);*

```
X2=mod(X2+subkey(2,round),2^16);
X3=mod(X3+subkey(3,round),2^16);
X4=mod(X4*subkey(4,round),2^16+1);
g1=bitxor(X1,X3);
g2=bitxor(X2,X4);
m1=mod(subkey(5,round)*g1,2^16+1);
a1=g2+m1;
m2=mod(subkey(6,round)*a1,2^16+1);
a2=m1+m2;
Y1=bitxor ( X1, m2);
Y2=bitxor ( X3, m2);
Y3=bitxor ( X2, a2);
Y4=bitxor ( X4, a2);
Swap Y2 and Y3;       /* not performed in last round */
[X1 X2 X3 X4]=[Y1 Y2 Y3 Y4];
```

4. After 8 rounds, perform the below steps:

```
Y1=mod(X1*subkey(1,9),2^16+1);
Y2=mod(X3+subkey(2,9),2^16);
Y3=mod(X2+subkey(3,9),2^16);
Y4=mod(X4*subkey(4,9),2^16+1);
```

5. Concatenate Y1, Y2, Y3 and Y4 after converting the data into binary form to generate the ciphertext.

ciphertext=[Y1 Y2 Y3 Y4];

Key scheduling for decryption

Keys used in encryption process are further processed to generate the decryption key as:

```
for round=1 to 8
        decryption_key (1,round)=inverse_modulo_multiplication(subkey(1,10-round));
        decryption_key (2,round)=inverse_modulo_addition(subkey(3,10-round));
        decryption_key (3,round)=2^16-subkey(2,10-round);
        decryption_key (4,round)=inverse_modulo_multiplication(subkey(4,10-round));
        decryption_key (5,round)=subkey(5,9-round);
        decryption_key (6,round)=subkey(6,9-round)
end
decryption_key  (1,9)=inverse_modulo_multiplication(subkey(1,10-round));      /* keys
required after 8th round */
decryption_key (2,9)=inverse_modulo_addition(subkey(3,10-round));
decryption_key (3,9)=2^16-subkey(2,10-round);
decryption_key (4,9)=inverse_modulo_multiplication(subkey(4,10-round));


function      [result]      =      inverse_modulo_multiplication(subkey);              /*
inverse_modulo_multiplication function definition */
    [g,c,d]=gcd(2^16+1,subkey);
    if d<0
       d=2^16+1+d;
    elseif d==0
       d=2^16;
```



```
    else
       d=d;
    end
    result=d;
end
```

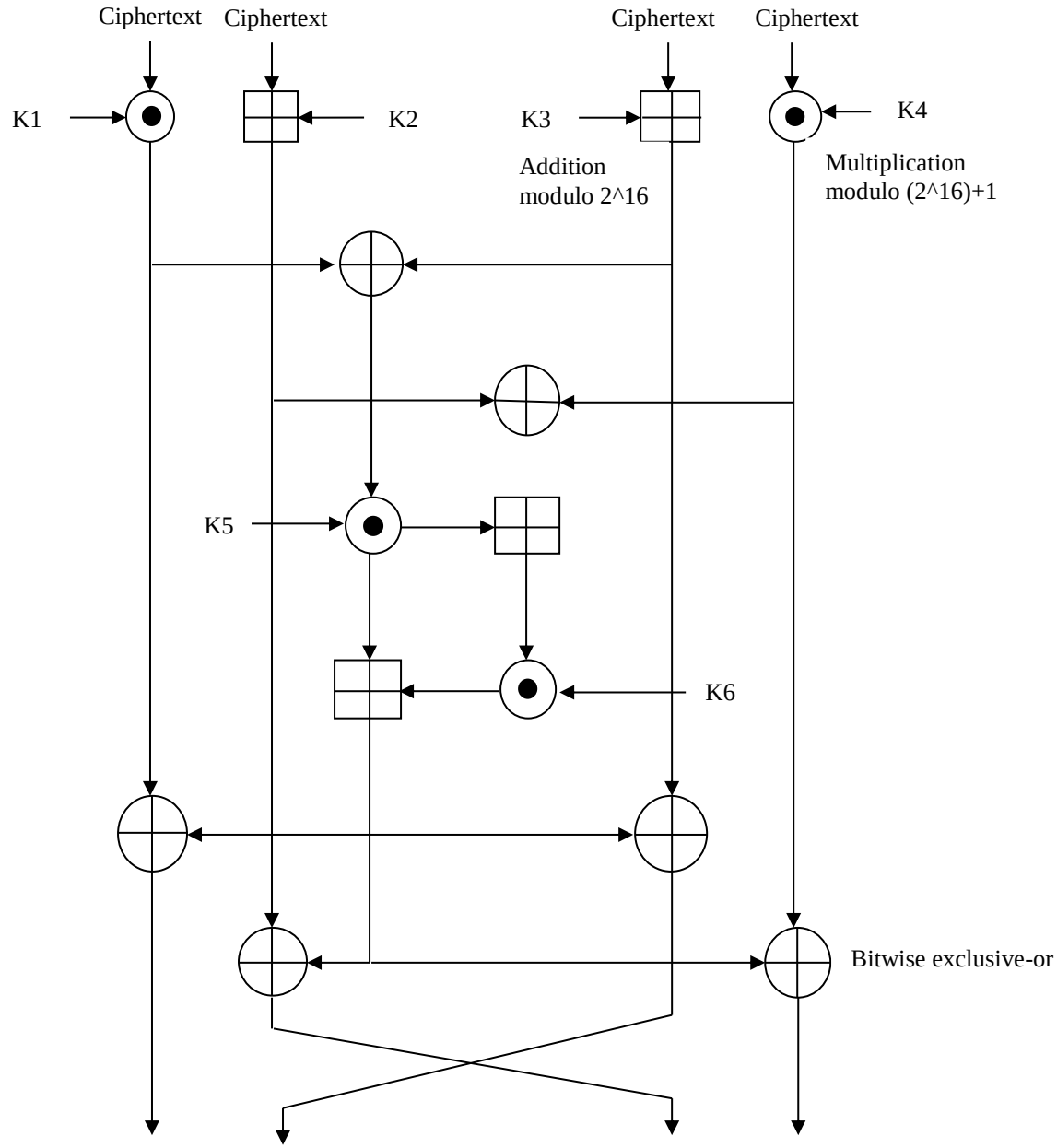

Figure 22. A Decryption Round

```
function [result] = inverse_modulo_addition(subkey)        /* inverse_modulo_addition
function definition */
    if subkey==0
       result=2^16;
    else
       result=2^16-subkey;
```

end
end

Decryption Algorithm

1. Obtain the ciphertext.

2. Apply the same process as for encryption to decrypt the ciphertext (see Figure 22).

Advantages:
1. Easy to implement.
2. Small time complexity.

Disadvantages:
1. The very simple key schedule makes IDEA subject to a class of weak keys.
2. Low security.

2.10. AES

Advanced Encryption Standard (AES), also known as Rijndael, is based on Rijndael cipher developed by two Belgian cryptographers John Daemen and Vincent Rijmen and published in 1998. AES is a block cipher and uses symmetric key to encrypt and decrypt the data. It encrypts a minimum of 16 bytes of data and key size can be 128/192/256 bits depending on number of rounds [5,28].

Key generation

In AES, since the key size is much smaller than the size of sub keys, so key expansion is provided by using the functions: rot word, sub word, rcon, ek and k. All of these are described as below:

Rot Word (4 bytes): This performs a circular left shift on 4 bytes.
rotword=circular shift arr by 1; / where arr is an array of 4 bytes */*

Sub Word (4 bytes): This step applies the S-box substitution in which first nibble of a byte tells the row number and second nibble tells the column number of S-box. The intersection of these two determines the value which is to be substituted. Here S-box is a 16x16 matrix containing hexadecimal values.

Rcon ((Round/(key size/4))-1): This function returns a 4 byte value based on the following:
Rcon (0) = 01000000, Rcon (1) = 02000000, Rcon (2) = 04000000, Rcon (3) = 08000000
Rcon (4) = 10000000, Rcon (5) = 20000000, Rcon (6) = 40000000, Rcon (7) = 80000000
Rcon (8) = 1B000000, Rcon (9) = 36000000, Rcon (10) = 6C000000, Rcon (11) = D8000000
Rcon (12) = AB000000, Rcon (13) = 4D000000, Rcon (14) = 9A000000

EK (offset): This function returns 4 bytes of the expanded key after the specified offset.
f=1;
for i=(offset+1) to (offset+4)
ek(f)=key(i);
f=f+1;
end

K (offset): This function returns 4 bytes of the expanded key after the specified offset.

```
f=1;
for i=(offset+1) to (offset+4)
   k(f)=key(i);
   f=f+1;
end
```

Encryption Algorithm

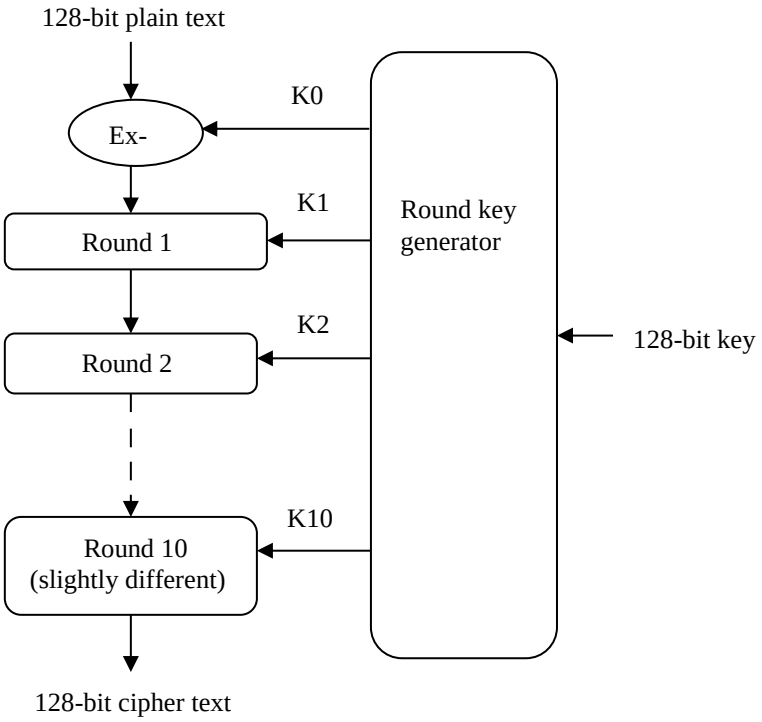

Figure 23. AES Encrvption Scheme

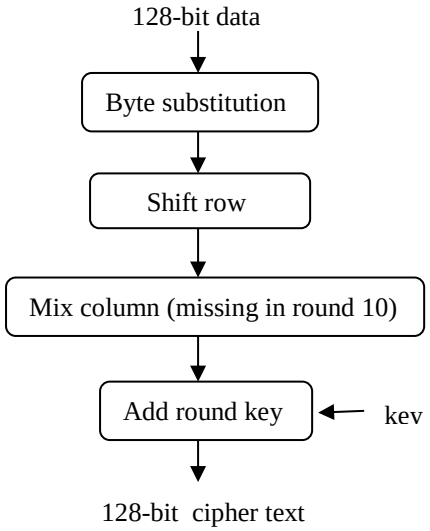

Figure 24. AES Encryption Round

Perform bitwise exclusive–or operation of 128-bit data and key K0 (see Figure 23). Then this result is processed in 10 rounds which contains the blocks described below (see Figure 24):

2.10.1. Byte Substitution: In this block, each value of state *i.e.*, current condition is replaced with the corresponding S-box value. This block is same as SubWord function used in key generation.

```
function [f]=byte_substitution(data)
b=char(data);
for m=1 to 16
    for j=1 to 16
      if((b(m,1)==dec2hex(j-1)))
         break;
       end
   end
     for(c=1:1:16)
         if(b(m,2)==dec2hex(c-1))
            break;
         end
     end
   f(m,:)= char(s_box(j,c));   /* s_box is 16x16 byte substitution matrix */
 end
end
```

2.10.2. Shift Row: It first arranges the state vector in matrix form and then performs a circular left shift for each row *i.e.*, it shifts Ith row by (I-1) times.

```
function [shiftrow1]=shiftrow(f)
shiftrow1=reshape(f,4,4);   /* arrange state vector in a matrix columnwise */
for i=1 to 4
   shiftrow1(i,:)=circular shift of shiftrow1(i,:) by (i-1) times;
end
end
```

2.10.3. Mix Column: Here matrix multiplication is performed one column at a time. Each value in the column is multiplied against every value of the matrix. The results of these multiplications are XORed together to produce only 4 result bytes for the next state. Thus creates 16 bytes at the end of multiplication. It uses Galois field multiplication in which hexadecimal values are not multiplied, instead they are replaced by values obtained from the two tables *i.e.*, E table and L table. The substitution from L table is same as byte substitution. Then resulted bytes are added together. If this value is greater than FF, then FF is subtracted from it and the result obtained is now substituted by value obtained from E table.

```
function [mix]=mix_column(sub1)
mul=[2 3 1 1;1 2 3 1;1 1 2 3;3 1 1 2];  /* standard matrix used in multiplication */
p=1;
for j=1 to 4
   for i=1 to 4
      for k2=1 to 4
         m1=hex2dec(l_tab(dec2hex(mul(i,k2),2)));  /* l_tab is L table used in substitution
*/
```



```
        m2=hex2dec(l_tab(dec2hex(sub1(k2,j),2)));
        m3=(m1+m2);
        if(m3>255)
           m3=m3-255;
        end
        m3=dec2hex(m3,2);
       x(k2)=hex2dec(e_tab(m3));   /* e_tab is E table used in substitution */
     end
        d=(bitxor(bitxor(bitxor(x(1),x(2)),x(3)),x(4)));
           mix(p)=d;                    /* 16 bytes state vector */
              p=p+1;
end
end
```

2.10.4. Add Round Key: In this block, 16 bytes state vector and the corresponding key are XORed to get the resultant state vector.

```
function [addround1]=add_round_key(state,key)
for i=1 to 16
   addround1(i)=(bitxor(state(i),key(i)));
end
end
```

The above blocks are processed for 10 rounds and in 10^{th} round, mix column block is not executed. The result of last round is the ciphertext.

Decyption Algorithm

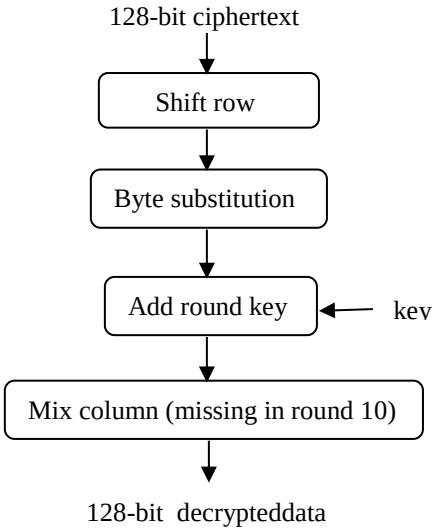

Figure 25. AES Decryption Round

Decryption process for AES is same as encryption (see Figure 25). The only difference is that order of function blocks has been interchanged.

1. Obtain the ciphertext and XOR it with the initial key K0.
2. Repeat steps 3 to 6 for 10 rounds.

3. **Shift row:** It first arranges the state vector in matrix form and then performs a circular right shift for each row *i.e.,* it shifts Ith row by (I-1) times.

4. **Byte substitution:** In this block, each value of state *i.e.,* current condition is replaced with the corresponding S-box value. Here the S-box used is different from that was used in encryption process.

5. **Add round key:** In this block, 16 bytes state vector and the corresponding key are XORed to get the resultant state vector.

6. **Mix column:** This block is same as for encryption. The only difference is that the standard matrix *mul* used for multiplication is different which is:

mul=['0E' '0B' '0D' '09';'09' '0E' '0B' '0D';'0D' '09' '0E' '0B';'0B' '0D' '09' '0E'];

This block is not present in last round.

Advantages:

1. Secure due to complex structure.
2. As key size is large, so possibility of brute force attack is less.

Disadvantages:

1. System complexity because of complex operations.
2. Large time complexity.

Next section provides the experimental setup and security parameters to evaluate the performance of all the implemented schemes.

3. Experiments and Security Analysis

3.1. Experimental Setup

The experiments are carried out on a personal computer. Table 2 provides the specifications and initial values and parameters.

Table 2. Specification Table

Processor	1.7GHz Intel Core-i3
Memory	4 GB
Operating system	Windows 8.1
Simulation tool	MATLAB 7.14.0.739, 64 bit (win 64)
Version	2010
Text used for encryption	“ABCDEFABCDEFABCD”
Hierarchical visual cryptography	Number of share in which data is divided=4

3.2. Performance Analysis Tests and Parameters

3.2.1. Visual Assessment: Visual assessment is the first test. It needs to be to performed on the encrypted output. If attacker is able to deduce some meaningful information by visual assessing the encrypted result, then the scheme is said to be failed at the first step itself. So for a scheme to be successful, the attacker should not be able to deduce any meaningful information out of encrypted result.

3.2.2. Key-space Analysis: Encryption schemes should be highly sensitive towards a very small change in key, used during encryption. Using a large key space ensures resistance of technique towards brute force attacks.

3.2.3. Statistical Analysis: This analysis is done to analyze the confusion properties of an encrypted image. Evaluation of correlation coefficients shows the relation among the encrypted data and original data. Correlation analysis can specify which technique has better confusion properties and can resist statistical attacks.

3.2.3.1. Correlation Analysis: This analysis calculates the correlation among the encrypted data and original data. A good encryption technique should result into an encrypted data with zero correlation ideally. In order to calculate the correlation between plain-text and cipher-text, the following formulae is used.

$$r\alpha\beta = \frac{cov(\alpha,\beta)}{\sqrt{D(\alpha)}\sqrt{D(\beta)}} \quad (3)$$

$$cov(\alpha,\beta) = \frac{1}{N}\sum_{i=1}^{N}(\alpha i - E(\alpha))(\beta i - E(\beta))) \quad (4)$$

$$D(\alpha) = \frac{1}{N}\sum_{i=1}^{N}(\alpha i - E(\alpha))^2 \quad (5)$$

$$D(\beta) = \frac{1}{N}\sum_{i=1}^{N}(\beta i - E(\beta))^2 \quad (6)$$

where $E(\alpha)$= mean of α and $E(\beta)$= mean of β, α and β denote two values for which correlation needs to be calculated, and N is the total number of elements obtained from the data.

3.2.4. Key Sensitivity Analysis: Key sensitivity analysis is performed with the aim to check the sensitivity of encryption scheme towards change in initial conditions. It means that a slight change in encryption key should produce an entirely different cipher image.

3.2.5. Information Entropy Analysis: Information entropy is the measure of amount of randomness. The entropy H(S) of a message m can be calculated as:

$$H(S) = \sum_{i=0}^{n-1} P(S_i) log_2 \frac{1}{P(S_i)} \quad (7)$$

where P(Si) signifies the probability symbol Si, log is of base 2. If there are 256 possible outcomes of the message S with equal probability, then H(S) = 8, which is an ideal value for this case. The value of entropy close to value 8 signifies that encrypted output is highly random in nature.

3.2.6. Computational Speed Analysis: An encryption mechanism is required to be fast and should take very less encryption time. It usually depends upon a number of factors like, the type of processor used, programming language used and type of operating system used.

4. Results

Various analysis has been done to measure the performance of each cryptography technique. The following results depict the performance measurement are:

4.1. Visual Assessment

Table 3. Visual Assessment

Cryptography Technique	Original Secret	Encrypted Data	Decrypted Data
RSA	ABCDEFABCD EFABCD	¶Ô¨ÌÍ¦¶Ô¨ÌÍ¦¶Ô¨Ì	ABCDEFABCD EFABCD
Diffie-Hellman	ABCDEFABCD EFABCD	ÁÂÃÄÅÆÁÂÃÄÅÆÁÂÃÄ	ABCDEFABCD EFABCD
Vignere Table	ABCDEFABCD EFABCD	NHJGKXNHJGKXNHJG	ABCDEFABCD EFABCD
Visual Cryptography	ABCDEFABCD EFABCD	ÇùD*([{ÒBu (share 1) 5J½ÀLÐil-=F1 (share 2)	ABCDEFABCD EFABCD
Hierarchical Cryptography	ABCDEFABCD EFABCD	5J½ÀLÐil-=F1 (key share) ÇùD*([{ÒBu (rest share)	ABCDEFABCD EFABCD
RC4	ABCDEFABCD EFABCD	eBãXÕéØ³Ã$Pn¹ug	ABCDEFABCD EFABCD
Elliptic Cryptography	ABCDEFABCD EFABCD	i´--<<-xáÃÃixÿ¥ðÃá--ðá<¥¥Z-áxix´´Z--x<<áÃ-´áZ-ð<xÿKZZ-ÒZÒá-á--xððáKðÒ-áZÒxxxZKÿx-á´Ò (cipher 1) ÓZ=xx[ðÒñZxµ[ðÒð´Z<ñxxZ´Zyð´´<µZ[Ó<ð<´ððñ´ðZ (cipher 2)	ABCDEFABCD EFABCD
DES	ABCDEFABCD EFABCD	++sC|rÃP^«□{A	ABCDEFABCD EFABCD
AES	ABCDEFABCD EFABCD	÷üjðK□hTMØ	ABCDEFABCD EFABCD
IDEA	ABCDEFABCD EFABCD	ôî+±0m{WyQ%	ABCDEFABCD EFABCD

After analyzing the Table 3, it is found that the encrypted data is either in unreadable form or depicts no information about the actual data. Hence, by visual inspection, no relationship between the original secret and encrypted value is found which proves that all the defined schemes are successful in this test.

4.2. Key-space Analysis

Table 4. Key-space Analysis

Cryptography Technique	Key size	Key space
RSA	Not fixed	-
Diffie-Hellman	Not fixed	-
Vignere Table	Not fixed	-
Visual Cryptography	Same as message length(say l)	2^l
Hierarchical Cryptography	Same as message length(say l)	2^l
RC4	1 to 256(say p)	2^p
Elliptic Cryptography	Not fixed but smaller key size	-
DES	64	2^{64}
AES	128	2^{128}
IDEA	128	2^{128}

Table 4 depicts the chances for brute force attacks *i.e.*, breaking of algorithm by trial and error method to get the key by using automated software to generate a large number of consecutive guesses for all the described techniques. Larger the key space, low the possibility of this attack. So, for a good cryptography scheme, key size should be large to get a large key space.

4.3. Correlation Analysis

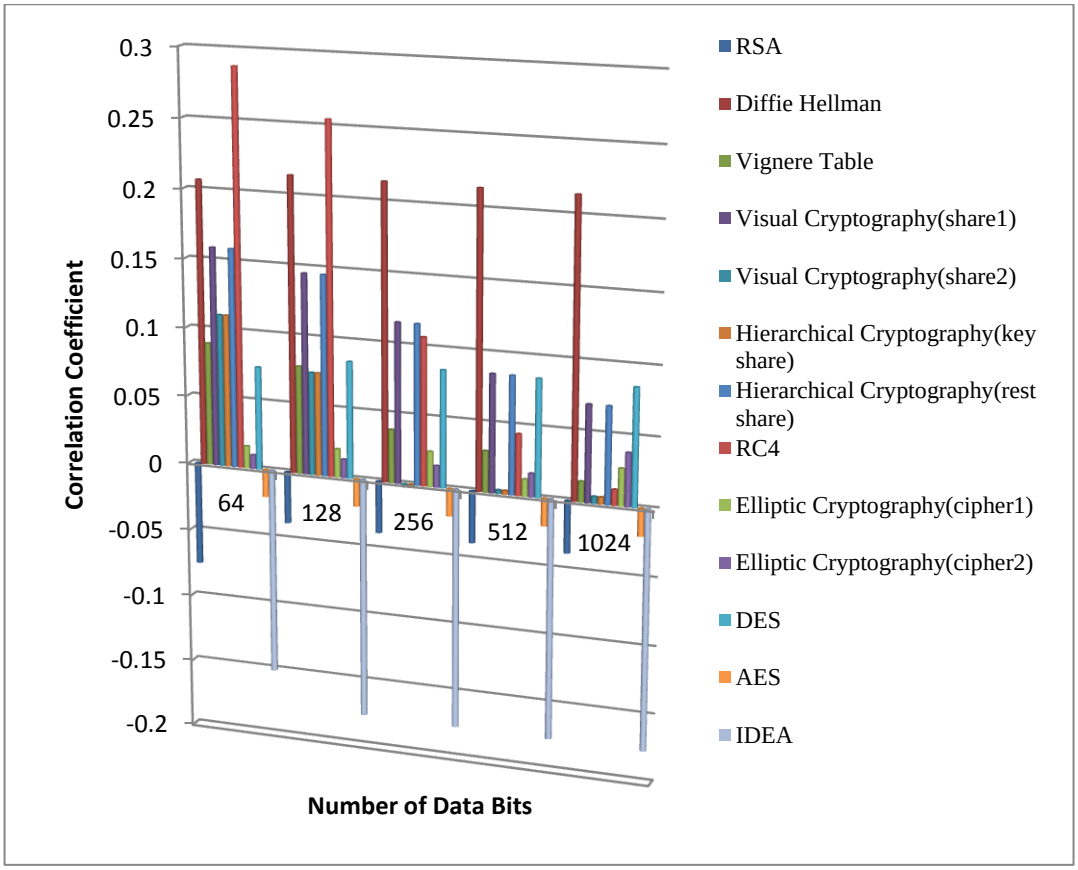

Figure 26. Effect of Data Size on Correlation Coefficient

Figure 26 depicts that correlation coefficient, which is a measure of similarity between input data and encrypted data, goes down for some schemes and remain constant for some other schemes. This analysis does not provide good results for schemes RSA, Diffie Hellman, DES and AES. IDEA has least value of correlation coefficient among all schemes, but it becomes stable after a small increment in data size. Correlation coefficient is going down more for hierarchical cryptography at every step of grown data bit which proves this technique better as compared to other techniques.

4.4. Key Sensitivity Analysis

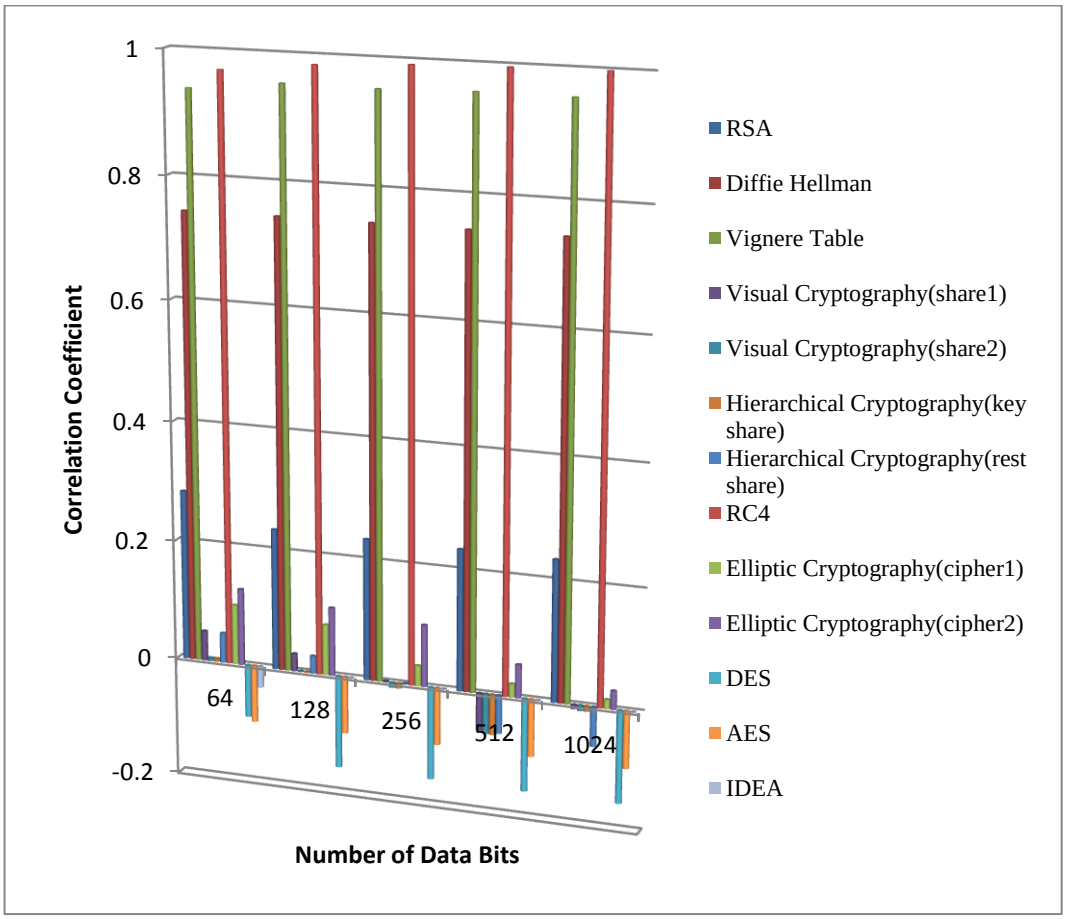

Figure 27. Effect of Data Size on Correlation Coefficient with a Small Change in Key

Figure 27 depicts that key sensitivity, which is a measure of similarity between two encrypted data obtained by applying a small change in key on the same input data, does not provide good results for traditional schemes such as RSA, Diffie Hellman, Vignere table cipher, AES, DES and IDEA. While the schemes RC4, visual cryptography, hierarchical cryptography and elliptic curve cryptography are found to be effective in this test. As there is a rapid decrease in the value of correlation coefficient in visual and hierarchical cryptography techniques, these schemes prove to be more key sensitive.

4.5. Information Entropy Analysis

Figure 28 shows that with an increase in data size, entropy which is a measure of randomness in encrypted data, traditional schemes such as RSA, visual cryptography, hierarchical cryptography, RC4 and elliptic curve cryptography comes very close in

providing good results, but value of entropy for visual cryptography and hierarchical cryptography comes very near to ideal value 8, thereby showing the output with most randomness. While it does not change for the schemes DES, Diffie Hellman, AES and IDEA. For vignere cipher, it first increases and then becomes stable. Higher the entropy, higher the randomness *i.e.*, unpredictability.

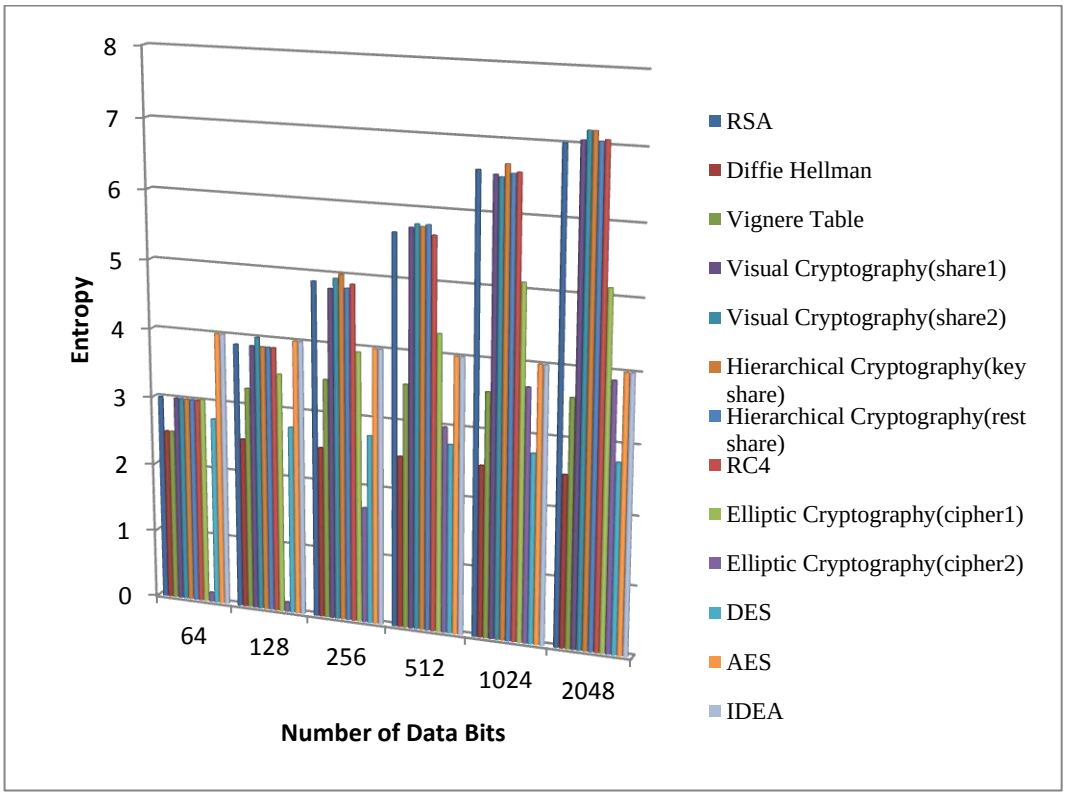

Figure 28. Effect of Data Size on Entropy

4.6. Computational Speed Analysis

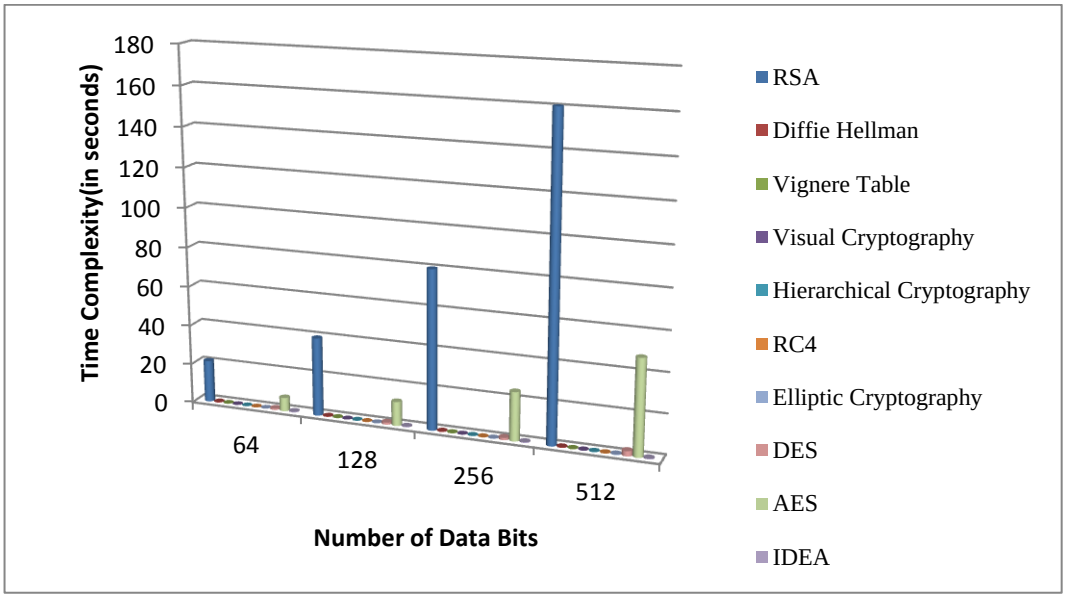

Figure 29. Effect of Data Size on Time Complexity

Figure 29 clearly shows that the time complexity, which is a measure of computational speed, is highest for RSA algorithm and higher for AES as compared to all other techniques which do not provide good results with a rise in data size, while all traditional schemes take almost no time *i.e.,* very small time to process. Higher the time complexity, lower the processing speed. So, for a good scheme, time complexity should be low.

5. Conclusion

This paper is an effort to provide a detailed comparison survey of various text based encryption mechanisms. Ciphers ranging from traditional ciphers to more recent ones are compared in detail. After implementation of all the said schemes, a detailed comparison was drawn on basis of various performance tests such as visual assessment, key space analysis, correlation analysis, key sensitivity, information entropy analysis and computational speed analysis and individual quality attribute is attached to the scheme on basis of performance. Table 5 clearly depicts that hierarchical cryptography scheme emerges out to be the best amongst all on the basis of overall score.

Table 5. Overall Comparison

Techniques	Entropy	Time complexity	Correlation coefficient	Key sensitivity
RSA	High	Highest	Lower and constant	Insensitive
Diffie Hellman	Low	Low	Low and constant	Insensitive
Vignere table	Low	Lower	Low and variable	Insensitive
Visual cryptography	Higher	Lower	Low and variable	Highly sensitive
Hierarchical cryptography	Higher	Lower	Low and variable	Highly sensitive
RC4	Higher	Low	Medium and variable	Sensitive
Elliptic curve cryptography	High	High	Low and variable	Sensitive
DES	Low	High	Low and constant	Insensitive
AES	Medium	Higher	Low and constant	Insensitive
IDEA	Medium	High	Lowest and constant	Insensitive

References

[1] A. Shamir, "How to Share a Secret", in proceedings of Communications of Association for Computing Machinery (ACM), vol. 22, Issue 11, **(1979)**, pp. 612-613.
[2] X. Lai and J. L. Massey, "A Proposal for a New Block Encryption Standard", in proceedings of Springer-Verlag, **(1991)**, pp. 389-404.
[3] A. Mousa and A. Hamad, "Evaluation of the RC4 Algorithm for Data Encryption", in proceedings of International Journal of Computer Science & Applications, vol. 3, Issue 2, **(2006)**, pp. 44-56.
[4] C. Christensen, "Cryptography of the Vigenere Cipher", in proceedings of Computer Sciences Corporation, **(2006)**, pp. 1-18.
[5] M. N. Islam, Md. M. H. Mia, Md. F. Islam and M. A. Matin, "An Efficient Approach for Increasing Security to Symmetric Data Encryption", in proceedings of International Journal of Computer Science and Network Security, vol. 8, Issue 4, **(2008)**, pp. 16-20.

[6] M. J. Kakish, "Security Improvments to the Diffie-Hellman Schemes", in proceedings of International Journal of Research and Reviews in Applied Sciences, vol. 8, Issue 1, **(2011)**.
[7] M. Amara and A. Siad, "Elliptic Curve Cryptography and its Applications", in proceedings of 7th International Workshop on Systems, Signal Processing and their Applications (WOSSPA), **(2011)**, pp. 247-250.
[8] L. Stosic and M. Bogdanovic, "RC4 Stream Cipher and possible attacks on WEP", in proceedings of International Journal of Advanced Computer Science and Applications, vol. 3, Issue 3, **(2012)**, pp. 111-114.
[9] Q-A. Kester, "A Cryptosystem based on Vignere Cipher with Varying key", in proceedings of International Journal of Advanced Research in Computer Engineering & Technology, vol. 1, Issue 10, **(2012)**, pp. 108-113.
[10] K. R. Shah and B. Gambhava, "New Approach of Data Encryption Standard Algorithm", in proceedings of International Journal of Soft Computing and Engineering, vol. 2, Issue 1, **(2012)**, pp. 322-325.
[11] R. Pahal and V. Kumar, "Efficient Implementation of AES", in proceedings of International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, Issue 7, **(2013)**, pp. 290-295.
[12] N. Y. Goshwe, "Data Encryption and Decryption Using RSA Algorithm in a Network Environment", in proceedings of International Journal of Computer Science and Network Security, vol. 13, Issue 7, **(2013)**, pp. 9-13.
[13] H. P. Singh, S. Verma and S. Mishra, "Secure-International Data Encryption Standard", in proceedings of International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 2, Issue 2, **(2013)**, pp. 780-792.
[14] Ms. S. Patil and Prof. A. C. Lomte, "Impact of Implementation of Cryptographic IDEA Algorithms for Secure Data Communication", in proceedings of International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, Issue 11, **(2013)**, pp. 775-777.
[15] B. S. Preeti, "Review Paper on Security in Diffie-Hellamn Algorithm", in proceedings of International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, Issue 3, **(2014)**, pp. 264-266.
[16] Saranya, Vinothini and Vasumathi, "A Study on RSA Algorithm for Cryptography", in proceedings of International Journal of Computer Science and Information Technologies, vol. 5, **(2014)**, pp. 5708-5709.
[17] A. Tuteja and A. Shrivastava, "Faster Decryption and More Secure RSA Cryptosystem", in proceedings of International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, Issue 11, **(2014)**, pp. 919-923.
[18] S. Mandal, S. Das and A. Nath, "Data hiding and Retrival using Visual Cryptography", in proceedings of International Journal of Innovative Research in Advanced Engineering, vol. 1, Issue 1, **(2014)**, pp. 102-110.
[19] M. E. Hodeish and Dr. V. T. Humbe, "State-of-the-Art Visual Cryptography Schemes", in proceedings of International Journal of Electronics Communication and Computer Engineering, vol. 5, Issue 2, **(2014)**, pp 412-420.
[20] P. V. Chavan, Dr. M. Atique and Dr. L. Malik, "Design and Implementation of Hierarchical Visual Cryptography with Expansionless Shares", in proceedings of International Journal of Network Security & Its Applications, vol. 6, no. 1, **(2014)**.
[21] F. M. Sher Ali and F. H. Sarhan, "Enhancing Security of Vignere Cipher by Stream Cipher", in proceedings of International Journal of Computer Applications, vol. 100, Issue 1, **(2014)**, pp. 1-4.
[22] Ms. B. Shrivas and Prof. S. Yadav, "A Survey on Visual Cryptography Techniques and their Applications", in proceedings of International Journal of Computer Science and Information Technologies, vol. 6, Issue 2, **(2015)**, pp. 1076-1079.
[23] V. Alikkal, Dr. T. Senthil Prakash and A. Hussain, "Enhanced Hierarchical Design for Visual Cryptography-Overview", in proceedings of International Journal on Engineering Technology and Sciences, vol. 2, Issue 4, **(2015)**.
[24] Ms. B. Sharma and Mrs. V. Madaan, "Enhancing Security of MANETs by Implementing Elliptical Curve based Threshold Cryptography", in proceedings of International Journal of Engineering and Computer Science, vol. 4, Issue 7, **(2015)**, pp. 13346-13350.
[25] P. Jindal and B. Singh, "A Survey on RC4 Stream Cipher", in proceedings of I. J. Computer Network and Information Security, **(2015)**, pp. 37-45.
[26] https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0ahUKEwjz6ZvT9oDMAhWBxI4KHUBHAeQQFggrMAI&url=https%3A%2F%2Fusers.cs.jmu.edu%2Fabzugcx%2FPublic%2FStudent-Produced-Term-Projects%2FCryptology-2002-SPRING%2FIDEA-by-How-Shen-Chang-2004-FALL.doc&usg=AFQjCNGtmaiJs0-4c8rgonMf2RRKqU_PQA&sig2=x0kcELs9Mskt2r5a303CZA
[27] http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm
[28] http://www.quadibloc.com/crypto/co040401.htm

Authors

Neha Tayal, She is persuing M.Tech in the field of Electronics and communication at YMCA University of Science and Technology, Faridabad. She has passed her B.tech in the same field from G.S. Modern Vidya Niketan Institute of Engineering and Technology, Palwal..

Ritesh Bansal, He is a B. tech in Electronics Instrumentation & Control Engineering from YMCA University of Science & Technology, Faridabad, India. His research interest includes work Encryption, Digital Image processing, Steganography, Network Security.

Shailender Gupta, He is B.Tech (Electronics Engineering), M.Tech (Computer Engineering) and recieved his Ph. D in the area of ad-hoc mobile network security. His academic interests include network security, Signal Processing, automata theory and fuzzy logic. Currently working as Assistant Professor in Electronics Engineering department at YMCA University of Science and Technology, Faridabad, India.