Volume 8, No. 3, March – April 2017

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

Study of Modern Cryptographic Algorithms

Shraddha Kulkarni
Computer Engineering – Third Year, LDRP-Institute of Research and Technology
KSV University, Gandhinagar, Gujarat

Abstract: In the present times, when use of Internet and it's services has become so widespread, Personal privacy has become of the utmost priority of every person. One of the ways to safeguard their personal Information is the use of cryptography. This paper discusses the modern cryptographic Algorithms that have been in action in the past few decades. Different symmetric and asymmetric algorithms ensure that data is completely secure, and if a third party intercepted the message, it becomes difficult for the person to decipher the particular data due to multilevel encryption levels.

Keywords: Modern Cryptographic Algorithms, AES, DES, 3DES, RSA, Diffie-Hellman, DSA, SEAL,RC4.

I. INTRODUCTION

Human being from ages had two inherent needs -to communicate and share information and to communicate selectively. These two needs gave rise to the art of coding the messages in such a way that only the intended people could have access to the information. Unauthorized people could not extract any information, even if the scrambled messages fell in their hand. The art and science of concealing the messages to introduce secrecy in information security is recognized as cryptography. The word 'cryptography' was coined by combining two Greek words, 'Krypto' meaning hidden and 'graphene' meaning writing.

There are two basic techniques for encrypting information: symmetric encryption (also called secret key encryption) and asymmetric encryption (also called public key encryption).

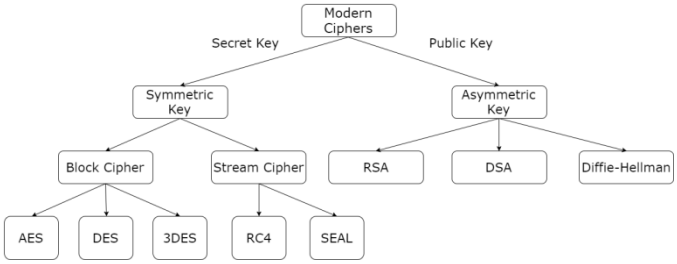

Fig. 1. A Hierarchy of Modern Ciphers

Symmetric encryption is the oldest and best-known technique. A secret key, which can be a number, a word, or just a string of random letters, is applied to the text of a message to change the content in a particular way. This might be as simple as shifting each letter by a number of places in the alphabet. As long as both sender and recipient know the secret key, they can encrypt and decrypt all messages that use this key.[1]

The problem with secret keys is exchanging them over the Internet or a large network while preventing them from falling into the wrong hands. Anyone who knows the secret key can decrypt the message. One answer is asymmetric encryption, in which there are two related keys--a key pair. A public key is made freely available to anyone who might want to send you a message. A second, private key is kept secret, so that only you know it. Any message (text, binary files, or documents) that are encrypted by using the public key can only be decrypted by applying the same algorithm, but by using the matching private key. Any message that is encrypted by using the private key can only be decrypted by using the matching public key.

This means that you do not have to worry about passing public keys over the Internet (the keys are supposed to be public). A problem with asymmetric encryption, however, is that it is slower than symmetric encryption. It requires far more processing power to both encrypt and decrypt the content of the message.[1]

II. SYMMETRIC KEY CRYPTOGRAPHIC ALGORITHMS

A. AES Algorithm

1) Description :

- AES is based on a design principle known as a substitution-permutation network, combination of both substitution and permutation, and is fast in both software and hardware. AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification per se is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits.

- AES operates on a 4 × 4 column-major order matrix of bytes, termed the state, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field.[2]

- The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext. The number of cycles of repetition are as follows:

10 cycles of repetition for 128-bit keys.
12 cycles of repetition for 192-bit keys.
14 cycles of repetition for 256-bit keys.

- Each round consists of several processing steps, each containing four similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.[2]

2) Outline of the Algorithm :

- Key Expansions—round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.

- Initial Round.

- AddRoundKey—each byte of the state is combined with a block of the round key using bitwise xor.

- Rounds :
  - ➢ SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
  - ➢ ShiftRows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
  - ➢ MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
  - ➢ AddRoundKey

- Final Round (no MixColumns) :
  - ➢ SubBytes
  - ➢ ShiftRows
  - ➢ AddRoundKey.[2]

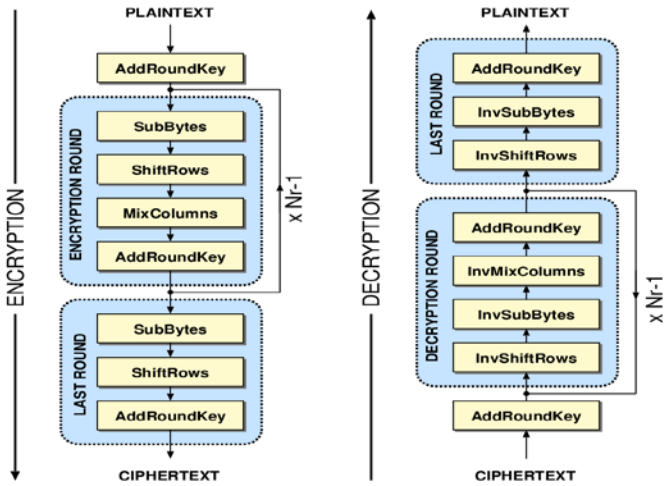

Fig. 2. Description of AES Encryption and Decryption cycle

3) Description of the Steps :

- *Step 1 : The SubBytes step*
  In the SubBytes step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table, S; bij = S(aij).

  In the SubBytes step, each byte in the state matrix is replaced with a SubByte using an 8-bit substitution box, the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF(28), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. While performing the decryption, Inverse SubBytes step is used, which requires first taking the affine transformation and then finding the multiplicative inverse (just reversing the steps used in SubBytes step).[2]

- *Step 2 : The ShiftRows step*
  In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row.

  The ShiftRows step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For blocks of sizes 128 bits and 192 bits, the shifting pattern is the same.

  Row n is shifted left circular by n-1 bytes. In this way, each column of the output state of the ShiftRows step is composed of bytes from each column of the input state. (Rijndael variants with a larger block size have slightly different offsets). For a 256-bit block, the first row is unchanged and the shifting for the second, third and fourth row is 1 byte, 3 bytes and 4 bytes respectively—this change only applies for the Rijndael cipher when used with a 256-bit block, as AES does not use 256-bit blocks. The importance of this step is to avoid the columns being linearly independent, in which case, AES degenerates into four independent block ciphers.[2]

- *Step 3 : The MixColumns step*
  In the MixColumns step, each column of the state is multiplied with a fixed polynomial.

  In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides diffusion in the cipher.

During this operation, each column is transformed using a fixed matrix (matrix multiplied by column gives new value of column in the state):

Matrix multiplication is composed of multiplication and addition of the entries. Entries are 8 bit bytes treated as coefficients of polynomial of order. Addition is simply XOR. Multiplication is modulo irreducible polynomial. If processed bit by bit then after shifting a conditional XOR with 0x1B should be performed if the shifted value is larger than 0xFF (overflow must be corrected by subtraction of generating polynomial).[2]

- *Step 4 : The AddRoundKey step*
  In the AddRoundKey step, each byte of the state is combined with a byte of the round subkey using the XOR operation. In the AddRoundKey step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.[2]

4) Applications :

- IEEE 802.11i, an amendment to the original IEEE 802.11 standard specifying security mechanisms for wireless networks, uses AES-128 in CCM mode (CCMP).
- The ITU-T G.hn standard, which provides a way to create a high-speed (up to 1 Gigabit/s) Local area network using existing home wiring (power lines, phone lines and coaxial cables), uses AES-128 for encryption.
- DataLocker Uses AES 256-bit CBC and XTS mode hardware encryption.
- Get Backup Pro uses AES-128 and AES-256
- GPG, GPL-licensed, includes AES, AES-192, and AES-256 as options.
- IPsec
- IronKey Uses AES 128-bit and 256-bit CBC-mode hardware encryption
- KeePass Password Safe
- LastPass
- Linux kernel's Crypto API, now exposed to userspace
- Pidgin (software), has a plugin that allows for AES Encryption
- PyEyeCrypt Free open-source text encryption tool/GUI with user-selectable AES encryption methods and PBKDF2 iterations.
- SocialDocs file encryption uses AES256 to provide a free-online file encryption tool
- TextSecure
- XFire uses AES-128, AES-192 and AES 256 to encrypt usernames and passwords
- Certain games and engines, such as the Rockstar Advanced Game Engine used in Grand Theft Auto IV, use AES to encrypt game assets in order to deter hacking in multiplayer.
- Intel and AMD processors include the AES instruction set.
- On IBM zSeries mainframes, AES is implemented as the KM series of assembler opcodes when various Message Security Assist facilities are installed.
- SPARC S3 core processors include the AES instruction set, which is used with SPARC T4 and SPARC T5 systems.[3]

B. DES Algorithm

1) Description :

- Originally designed by researchers at IBM in the early 1970s, DES was adopted by the U.S. government as an official Federal Information Processing Standard (FIPS) in 1977 for the encryption of commercial and sensitive yet unclassified government computer data. [4]

- It was the first encryption algorithm approved by the U.S. government for public disclosure. This ensured that DES was quickly adopted by industries such as financial services, where the need for strong encryption is high. The simplicity of DES also saw it used in a wide variety of embedded systems, smart cards, SIM cards and network devices requiring encryption like modems, set-top boxes and routers.[4]

- The Data Encryption Standard is a block cipher, meaning a cryptographic key and algorithm are applied to a block of data simultaneously rather than one bit at a time. To encrypt a plaintext message, DES groups it into 64-bit blocks. Each block is enciphered using the secret key into a 64-bit ciphertext by means of permutation and substitution. The process involves 16 rounds and can run in four different modes, encrypting blocks individually or making each cipher block dependent on all the previous blocks.

- Decryption is simply the inverse of encryption, following the same steps but reversing the order in which the keys are applied. For any cipher, the most basic method of attack is brute force, which involves trying each key until you find the right one. The length of the key determines the number of possible keys -- and hence the feasibility -- of this type of attack. DES uses a 64-bit key, but eight of those bits are used for parity checks, effectively limiting the key to 56-bits. Hence, it would take a maximum of 2^56, or 72,057,594,037,927,936, attempts to find the correct key.[4]

2) Outline of Algorithm :

- Since DES is based on the Feistel Cipher, all that is required to specify DES is:
  - Round Function
  - Key Schedule
  - Any additional processing - Initial and Final Permutation.

- Input:
  T: 64 bits of clear text
  k1, k2, ..., k16: 16 round keys
  IP: Initial permutation
  FP: Final permutation
  f(): Round function

- Output:
  C: 64 bits of ciphertext

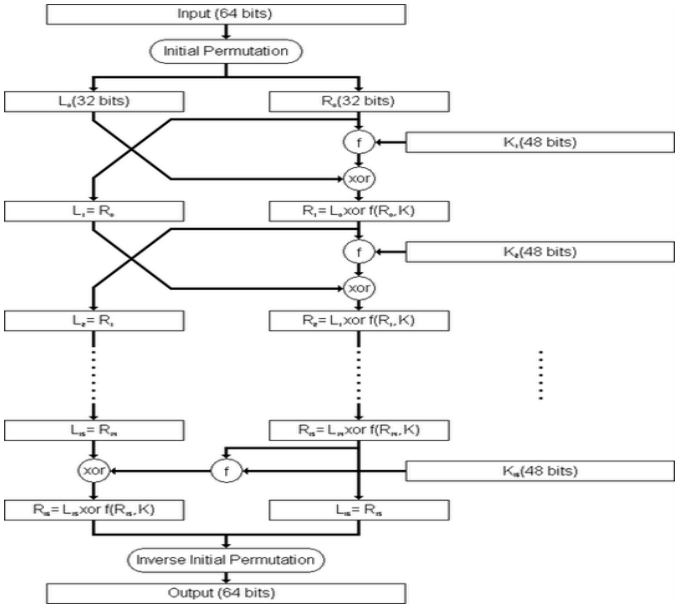

Fig. 3. Flowchart for DES Algorithm

- Overview :
  - ➢ T' = IP(T), applying initial permutation
  - ➢ (L0, R0) = T', dividing T' into two 32-bit parts
    (L1, R1) = (R0, L0 ^ f(R0, k1))
    (L2, R2) = (R1, L1 ^ f(R1, k2))
  - ➢ C' = (R16, L16), swapping the two parts
  - ➢ C = FP(C'), applying final permutation
    where ^ is the XOR operation.[6]

3) Description of Functions :

- The round function f(R,k) is defined as:

  Input:
  R: 32-bit input data
  k: 48-bit round key
  E: Expansion permutation
  P: Round permutation
  s(): S boxes function

  Output:
  R' = f(R,k): 32-bit output data

  Algorithm:
  X = E(R), applying expansion permutation and returning 48-bit data
  X' = X ^ k, XOR with the round key
  X'' = s(X'), applying S boxes function and returning 32-bit data
  R' = P(X''), applying the round permutation[6]

- The S boxes function s(X) is defined as:
  Input:
  X: 48-bit input data
  S1, S2, ..., S8: 8 S boxes - 4 x 16 tables
  Output:
  X' = s(X): 32-bit output data

  Algorithm:
  (X1, X2, ..., X8) = X, dividing X into 8 6-bit parts
  X' = (S1(X1), S2(X2), ..., S8(X8))
  where Si(Xi) is the value at row r and column c of S box i with
  r = 2*b1 + b6
  c = 8*b2 + 4*b3 + 2*b3 + b4
  b1, b2, b3, b4, b5, b6 are the 6 bits of the Xi .[6]

4) Applications :

- ➢ Today, DES is considered to be too insecure for a number of transactions primarily due to the 56-bit key size being too small for modern technology. In January of 1999, the Electronic Frontier Foundation and distributed.net were able to collaborate to break a DES key in less than 24 hours (22 hours and 15 minutes). Additionally, NIST (National Institute of Standards and Technology) has withdrawn DES as a standard. The DES algorithm can also be referred to as the Data Encryption Algorithm (DEA).[7]

C. 3-DES Algorithm

1. Description :

- The speed of exhaustive key searches against DES after 1990 began to cause discomfort amongst users of DES. However, users did not want to replace DES as it takes an enormous amount of time and money to change encryption algorithms that are widely adopted and embedded in large security architectures.[8]
- The pragmatic approach was not to abandon the DES completely, but to change the manner in which DES is used. This led to the modified schemes of Triple DES (sometimes known as 3DES).
- Incidentally, there are two variants of Triple DES known as 3-key Triple DES (3TDES) and 2-key Triple DES (2TDES).
- Before using 3TDES, user first generate and distribute a 3TDES key K, which consists of three different DES keys K_1, K_2 and K_3. This means that the actual 3TDES key has length 3×56 = 168 bits. The encryption scheme is illustrated as follows −

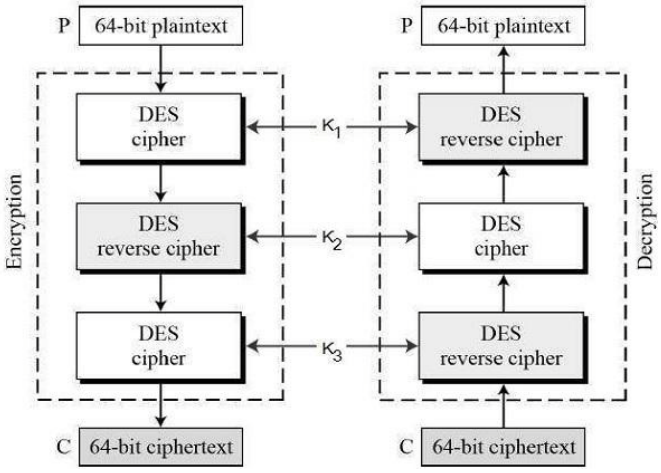

Fig. 4. Flowchart for 3-DES Algorithm

- The encryption-decryption process is as follows –
  - ➢ Encrypt the plaintext blocks using single DES with key K_1.
  - ➢ Now decrypt the output of step 1 using single DES with key K_2.
  - ➢ Finally, encrypt the output of step 2 using single DES with key K_3.
  - ➢ The output of step 3 is the ciphertext.
- Decryption of a ciphertext is a reverse process. User first decrypt using K_3, then encrypt with K_2, and finally decrypt with K_1.[8]
- Due to this design of Triple DES as an encrypt–decrypt–encrypt process, it is possible to use a 3TDES (hardware) implementation for single DES by setting K_1, K_2, and K_3 to be the same value. This provides backwards compatibility with DES.
- Second variant of Triple DES (2TDES) is identical to 3TDES except that K_3 is replaced by K_1. In other words, user encrypt plaintext blocks with key K_1, then decrypt with key K_2, and finally encrypt with K_1 again. Therefore, 2TDES has a key length of 112 bits.
- Triple DES systems are significantly more secure than single DES, but these are clearly a much slower process than encryption using single DES.[8]

2) Applications :

- ➢ With regard to the use of single DES, exhaustion of the DES (i.e., breaking a DES encrypted ciphertext by trying all possible keys) has become increasingly more feasible with technology advances. Following a recent hardware based DES key exhaustion attack, NIST can no longer support the use of single DES for many applications. Therefore, Government agencies with legacy 5 single DES systems are encouraged to transition to Triple DES. Agencies are advised to implement Triple DES when building new systems.
- ➢ The electronic payment industry uses Triple DES and continues to develop and promulgate standards based upon it (e.g. EMA).
- ➢ Microsoft OneNote, Microsoft Outlook 2007 and Microsoft System Center Configuration Manager 2012 use Triple DES to password protect user content and system data.[9]

D. RC4 Algorithm :

1) Description :

- RC4 generates a pseudorandom stream of bits (a keystream). As with any stream cipher, these can be used for encryption by combining it with the plaintext using bit-wise exclusive-or; decryption is performed the same way (since exclusive-or with given data is an involution). (This is similar to the Vernam cipher except that generated pseudorandom bits, rather than a prepared stream, are used.) To generate the keystream, the cipher makes use of a secret internal state which consists of two parts:
  - ➢ A permutation of all 256 possible bytes (denoted "S" below).
  - ➢ Two 8-bit index-pointers (denoted "i" and "j").
- The permutation is initialized with a variable length key, typically between 40 and 2048 bits, using the key-scheduling algorithm (KSA). Once this has been completed, the stream of bits is generated using the pseudo-random generation algorithm (PRGA).[10]

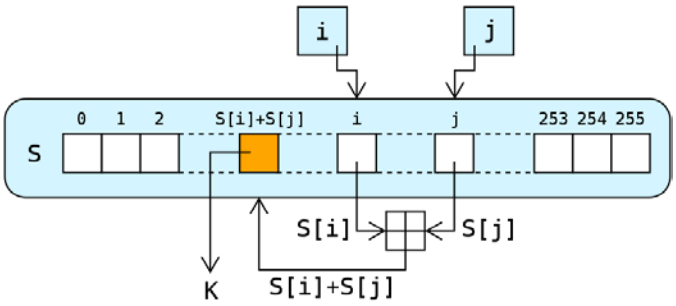

Fig. 5. Encryption of plainText to cipher

2) Description of Functions :

A. Key-scheduling algorithm (KSA)

The key-scheduling algorithm is used to initialize the permutation in the array "S". "keylength" is defined as the number of bytes in the key and can be in the range 1 = keylength = 256, typically between 5 and 16, corresponding to a key length of 40 – 128 bits. First, the array "S" is initialized to the identity permutation. S is then processed for 256 iterations in a similar way to the main PRGA, but also mixes in bytes of the key at the same time.[10]

```
for i from 0 to 255
        S[i] := i
Endfor
j := 0
for i from 0 to 255
j := (j + S[i] + key[i mod keylength]) mod 256
        swap values of S[i] and S[j]
        Endfor
```

B. Pseudo-random generation algorithm (PRGA)

The lookup stage of RC4. The output byte is selected by looking up the values of S[i] and S[j], adding them together modulo 256, and then using the sum as an index into S; S(S[i] + S[j]) is used as a byte of the key stream, K.

For as many iterations as are needed, the PRGA modifies the state and outputs a byte of the keystream. In each iteration, the PRGA increments i, looks up the ith element of S, S[i], and adds that to j, exchanges the values of S[i] and S[j], and then uses the sum S[i] + S[j] (modulo 256) as an index to fetch a third element of S, (the keystream value K below) which is bitwise exclusive OR'ed (XOR'ed) with the next byte of the message to produce the next byte of either ciphertext or plaintext. Each element of S is swapped with another element at least once every 256 iterations.[10]

```
i := 0
j := 0
while GeneratingOutput:
        i := (i + 1) mod 256
        j := (j + S[i]) mod 256
        swap values of S[i] and S[j]
        K := S[(S[i] + S[j]) mod 256]
        output K
endwhile
```

3) Applications :

➢ RC4 is a fast stream cipher invented in 1987 by Ron Rivest. Despite its age, RC4 has become an extremely popular ciphersuite for SSL/TLS connections — to the point where it accounts for something absurd like half of all TLS traffic. There are essentially two reasons for this:

- RC4 does not require padding or IVs, which means it's immune to recent TLS attacks like BEAST and Lucky13. Many admins have recommended it as the solution to these threats.
- RC4 is pretty fast. Faster encryption means less computation and therefore lower hardware requirements for big service providers like Google.

➢ According to AlFardan, Bernstein, Paterson, Poettering and Schuldt (a team from Royal Holloway, Eindhoven and UIC) the RC4 ciphersuite used in SSL/TLS is broken. If you choose to use it — as do a number of major sites, including Google — then it may be possible for a dedicated attacker to recover your authentication cookies. The current attack is just on the edge of feasibility, and could probably be improved for specific applications.

E. SEAL Algorithm

- In cryptography, SEAL (Software-Optimized Encryption Algorithm) is a very fast stream cipher optimised for machines with a 32-bit word size and plenty of RAM. SEAL is actually a pseudorandom function family in that it can easily generate arbitrary portions of the keystream without having to start from the beginning. This makes it particularly well suited for applications like encrypting hard drives.
- The first version was published by Phillip Rogaway and Don Coppersmith in 1994. The current version, published in 1997, is 3.0. SEAL, covered by two patents in the United States, both of which are assigned to IBM.[11]

1) Applications :

➢ On a modern 32- bit processor SEAL can encrypt messages at a rate of about clock cycles per byte of text In comparison the DES algorithm is more than times as expensive. Even a Cyclic Redundancy Code CRC is more costly.

➢ It should be emphasized that using SEAL in the expected way does nothing to provide for data authenticity Many applications which require data privacy also require data authenticity. Such applications should accompany SEAL encrypted data by a message authentication code (MAC) Techniques for fast MAC generation are an active area of research. [12]

➢ SEAL is endian neutral and yet an endian convention is needed to interoperably encrypt using SEAL. One possibility is to allow encryption with either endian convention but to include information in SEAL encrypted ciphertext which unambiguously indicates the endian convention employed. It is easy to modify SEAL to get a cipher optimized for bit architectures The tables would be twice as wide and Initialize would be slightly changed. SEAL has the unusual attribute that doubling the word size and making natural changes in the ciphers definition would nearly double the ciphers speed It is unclear whether security would be impacted by the longer word length.[12]

III. ASYMMETRIC KEY CRYPTOGRAPHIC ALGORITHMS

A. RSA Algorithm

- RSA is designed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1978. It is one of the best known public key cryptosystems for key exchange or digital signatures or encryption of blocks of data. RSA uses a variable size encryption block and a variable size key.

- It is an asymmetric (public key) cryptosystem based on number theory, which is a block cipher system. It uses two prime numbers to generate the public and private keys. These two different keys are used for encryption and decryption purpose. Sender encrypts the message using Receiver public key and when the message gets transmit to receiver, then receiver can decrypt it using his own private key.[13]

- RSA have many flaws in its design therefore not preferred for the commercial use. When the small values of p & q are selected for the designing of key then the encryption process becomes too weak and one can be able to decrypt the data by using random probability theory and side channel attacks.

- On the other hand if large p & q lengths are selected then it consumes more time and the performance gets degraded in comparison with DES. Further, the algorithm also requires of similar lengths for p & q, practically this is very tough conditions to satisfy. Padding techniques are required in such cases increases the system's overheads by taking more processing time.[13]

- The basic principle behind RSA is the observation that it is practical to find three very large positive integers e, d and n such that with modular exponentiation for all *m*:

$$(m^e)^d \equiv m \bmod n$$

and that even knowing e and n or even m it can be extremely difficult to find d.

- Additionally, for some operations it is convenient that the order of the two exponentiations can be changed and that this relation also implies:

$$(m^d)^e \equiv m \bmod n$$

- RSA operations can be decomposed in three broad steps;
  - ➢ Key generation
  - ➢ Encryption
  - ➢ Decryption [13]

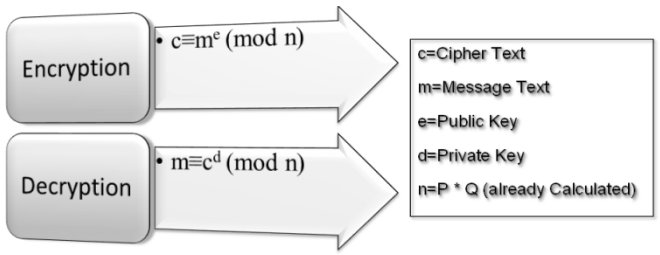

Fig. 6. RSA Algorithm

1) *Description of Functions* :

- Key Generation Procedure
  - ➢ Choose two distinct large random prime numbers p & q.
  - ➢ Compute n= p × q.
  - ➢ Calculate: phi (n) = (p-1) (q-1).
  - ➢ Choose an integer e such that 1<e<phi(n)
  - ➢ Compute d to satisfy the congruence relation d × e = 1
    mod phi (n); d is kept as private key exponent.
  - ➢ The public key is (n, e) and the private key is (n, d).
    Keep all the values d, p, q and phi secret.

- Encryption
  Plaintext: P < n
  Ciphertext: $C = P^e \bmod n$.

- Decryption
  Ciphertext: C
  Plaintext: $P = C^d \bmod n$.[13]

2) *Applications :*

- ➢ SSH can use either "RSA" (Rivest-Shamir-Adleman) or "DSA" ("Digital Signature Algorithm") keys. Both of these were considered state-of-the-art algorithms when SSH was invented, but DSA has come to be seen as less secure in recent years. RSA is the only recommended choice for new keys, so this guide uses "RSA key" and "SSH key" interchangeably.[15]
- ➢ Key-based authentication uses two keys, one "public" key that anyone is allowed to see, and another "private" key that only the owner is allowed to see. To securely communicate using key-based authentication, one needs to create a key pair, securely store the private key on the computer one wants to log in from, and store the public key on the computer one wants to log in to.[15]
- ➢ In practice, the RSA system is often used together with a secret-key cryptosystem, such as DES, to encrypt a message by means of an RSA digital envelope.[14]

B. Diffie-Hellman Key Exchange Algorithm

1) *Explanation :*

- Diffie-Hellman is a way of generating a shared secret between two people in such a way that the secret can't be seen by observing the communication. That's an important distinction: You're not sharing information during the key exchange, you're creating a key together.

- This is particularly useful because you can use this technique to create an encryption key with someone, and then start encrypting your traffic with that key. And even if the traffic is recorded and later analyzed, there's absolutely no way to figure out what the key was, even though the exchanges that created it may have been visible. This is where perfect forward secrecy comes from. Nobody analyzing the traffic at a later date can break in because the key was never saved, never transmitted, and never made visible anywhere.

- The way it works is reasonably simple. A lot of the math is the same as you see in public key crypto in that a trapdoor function is used. And while the discrete logarithm problem is traditionally used (the xy mod p business), the general process can be modified to use elliptic curve cryptography as well.

- But even though it uses the same underlying principles as public key cryptography, this is not asymmetric cryptography because nothing is ever encrypted or decrypted during the exchange. It is,

however, an essential building-block, and was in fact the base upon which asymmetric crypto was later built.

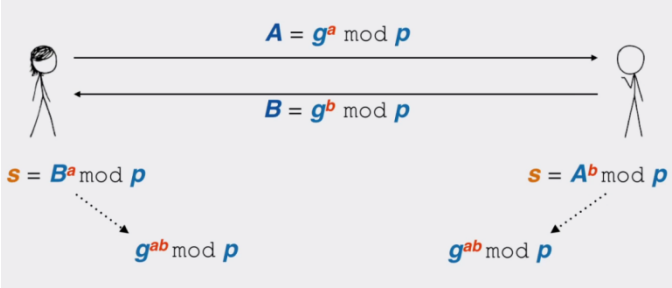

Fig. 7. Basic process involved in Diffie-Hellman Key Exchange Algorithm

- The basic idea works works like this:
  - ➢ I come up with two prime numbers g and p and tell you what they are.
  - ➢ You then pick a secret number (a), but you don't tell anyone. Instead you compute g^a mod p and send that result back to me. (We'll call that A since it came from a).
  - ➢ I do the same thing, but we'll call my secret number b and the computed number B. So I compute g^b mod p and send you the result (called "B")
  - ➢ Now, you take the number I sent you and do the exact same operation with it. So that's B^a mod p.
  - ➢ I do the same operation with the result you sent me, so: A^b mod p.

- The "magic" here is that the answer I get at step 5 is the same number you got at step 4. Now it's not really magic, it's just math, and it comes down to a fancy property of modulo exponents. Specifically:

$$(g^a \bmod p)^b \bmod p = g^{ab} \bmod p$$
$$(g^b \bmod p)^a \bmod p = g^{ba} \bmod p$$

  Which, if you examine closer, means that you'll get the same answer no matter which order you do the exponentiation in. So I do it in one order, and you do it in the other. I never know what secret number you used to get to the result and you never know what number I used, but we still arrive at the same result.

- That result, that number we both stumbled upon in step 4 and 5, is our shared secret key. We can use that as our password for AES or Blowfish, or any other algorithm that uses shared secrets. And we can be certain that nobody else, nobody but us, knows that key that we created together.

2) Applications :

➢ Secure Sockets Layer (SSL) is a cryptographic protocol developed by Netscape in 1995. SSL V3.0 has since become the predominant method and de-facto standard for securing information flow between web users and web servers.Users may not know they are using SSL, but they probably will notice the padlock icon and the “https:” in the URL that indicates encryption is taking place. SSL most commonly runs on TCP using port 443. SSL/TLS is composed of two layers: the lower layer called the Record Protocol rides on TCP and manages the symmetric (private) cryptography so the communication is private and reliable. The upper layer is called the Handshake Protocol and is responsible for authentication of the parties and negotiation of encryption methods and keys. It is in this layer that DH can be used. DH is considered the strongest alternative of the available options for the key exchange according to Wagner and Schneier .[16]

➢ Secure Shell (SSH) is a both a protocol and a program used to encrypt traffic between two computers. This is most commonly done as a secure replacement for tools like telnet, ftp and the Berkeley “r” commands (rlogin, rsh, etc.).The two parties to the connection (e.g., client and server) begin their conversation by negotiating parameters (e.g., preferred encryption and compression algorithms, and certain random numbers). Then a shared secret is computed using DH in a manner similar to SSL/TLS. A hashing function on the shared secret is used to derive an encryption key for the negotiated symmetric encryption algorithm (e.g., 3DES). From this point the two sides encrypt all traffic between them with the symmetric encryption algorithm.[16]

➢ Internet Protocol Security (IPSec) is a protocol being developed by the IETF to incorporate secure communications into the IP network layer itself. Like the previous protocols, IPSec uses DH and asymmetric cryptography to establish identities, preferred algorithms, and a shared secret. Then, the algorithm uses a symmetric cipher to encrypt the bulk data as it is transferred.[16]

C. DSA Algorithm

- The Digital Signature Algorithm (DSA) was introduced in 1994 by the U.S. Department of Commerce and National Institute of Standards and Technology. It uses the same Diffie-Hellman domain parameters (p, q, g) and private/public key pair (a, A = g^a mod p) for a signing party A.

- In the original standard, the prime factor q was specified to be 160 bits long and the message representative to be signed was the 160-bit SHA-1 hash of the message M.

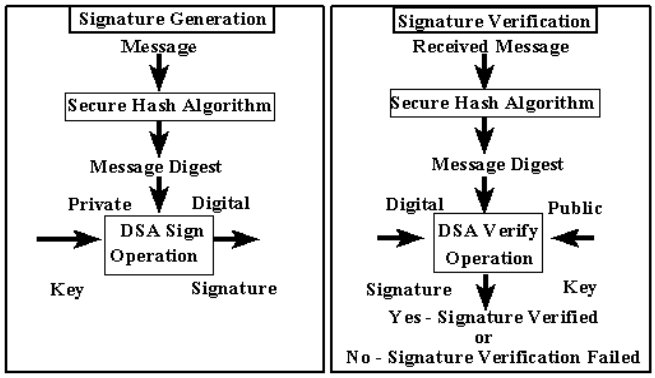

Fig. 8. DSA Algorithm Flowchart[18]

1) Description of Functions :

- DSA Signature Generation

INPUT:
Domain parameters (p, q, g); signer's private key a; message-to-be-signed, M, with message digest h = Hash(M).

OUTPUT: Signature (r, s).

- ➢ Choose a random k in the range [1, q − 1].
- ➢ Compute $X = g^k \bmod p$ and r = X mod q. If r = 0 (unlikely) then go to step 1.
- ➢ Compute $k^{-1} \bmod q$.
- ➢ Compute h = Hash(M).
- ➢ Compute $s = k^{-1} (h + ar) \bmod q$. If s = 0 (unlikely) then go to step 1.
- ➢ Return (r, s).[19]

- DSA Signature Verification

INPUT:
Domain parameters (p, q, g); signer's public key A; signed-message, M, with message digest h = Hash(M); signature (r, s).

OUTPUT: "Accept" or "Reject".

- ➢ Verify that r and s are in the range [1, q − 1]. If not then return "Reject" and stop.
- ➢ Compute $w = s^{-1} \bmod q$.
- ➢ Compute h = Hash(M).
- ➢ Compute u1 = hw mod q and u2 = rw mod q.
- ➢ Compute $X = g^{u1} * A^{u2} \bmod p$ and v = X mod q.
- ➢ If v = r then return "Accept" otherwise return "Reject".[19]

2) Applications :

- ➢ DSA is covered by U.S. Patent 5,231,668, filed July 26, 1991 and attributed to David W. Kravitz, a former NSA employee. This patent was given to "The United States of America as represented by the Secretary of Commerce, Washington, D.C.", and NIST has made this patent available worldwide royalty-free. Claus P. Schnorr claims that his U.S. Patent 4,995,082 (expired) covered DSA; this claim is disputed. DSA is a variant of the ElGamal Signature Scheme.[17]
- ➢ With DSA, the entropy, secrecy, and uniqueness of the random signature value k are critical. It is so critical that violating any one of those three requirements can reveal the entire private key to an attacker. Using the same value twice (even while keeping *k* secret), using a predictable value, or leaking even a few bits of *k* in each of several signatures, is enough to break DSA.

IV. CONCLUSIONS

Cryptography has been one of the favourite topics for researchers over the years. As the time passes, mathematicians and scholars keep coming up with new advancements in this field. Many breakthrough researches have already been made in this field, which has completely changed the course of history. This paper was a study of how the algorithms have evolved over the years, overcoming the issues of the previous versions. It gave a complete overview of modern cryptographic Algorithms which included symmetric and Asymmetric techniques. It has widely discussed the shortcomings of the algorithms and the areas where it is being used in the present times.

REFERENCES

[1] Description of Symmetric and Asymmetric Encryption https://support.microsoft.com/en-us/help/246071/description-of-symmetric-and-asymmetric-encryption
[2] Advanced Encryption Standard https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
[3] AES implementations https://en.wikipedia.org/wiki/AES_implementations
[4] Margaret Rouse Data Encryption Standard (DES) http://searchsecurity.techtarget.com/definition/Data-Encryption-Standard
[5] Data Encryption Standard https://www.tutorialspoint.com/cryptography/data_encryption_standard.htm
[6] Dr. Herong Yang DES (Data Encryption Standard) Cipher Algorithm http://www.herongyang.com/Cryptography/DES-Algorithm-Data-Encryption-Standard.html
[7] DES (Data Encryption Standard) http://www.tech-faq.com/des.html
[8] Triple DES https://www.tutorialspoint.com/cryptography/triple_des.htm
[9] Triple DES https://en.wikipedia.org/wiki/Triple_DES
[10] RC4 https://en.wikipedia.org/wiki/RC4
[11] SEAL (cipher) https://en.wikipedia.org/wiki/SEAL_(cipher)
[12] Phillip Rogaway, Don Coppersmith A Software Optimized Encryption Algorithm http://web.cs.ucdavis.edu/~rogaway/papers/seal.pdf
[13] RSA (cryptosystem) https://en.wikipedia.org/wiki/RSA_(cryptosystem)

[14] 3.1.7 HOW IS THE RSA ALGORITHM USED FOR PRIVACY IN PRACTICE? https://india.emc.com/emc-plus/rsa-labs/standards-initiatives/rsa-algorithm-used-for-privacy-in-practice.htm
[15] SSH/OpenSSH/Keys https://help.ubuntu.com/community/SSH/OpenSSH/Keys
[16] A Review of the Diffie-Hellman Algorithm and its Use in Secure Internet Protocols https://www.sans.org/reading-room/whitepapers/vpns/review-diffie-hellman-algorithm-secure-internet-protocols-751
[17] Digital Signature Algorithm https://en.wikipedia.org/wiki/Digital_Signature_Algorithm
[18] DIGITAL SIGNATURE STANDARD (DSS) http://www.umich.edu/~x509/ssleay/fip186/fip186.htm
[19] Public key cryptography using discrete logarithms. Part 4: Digital Signature Algorithm (DSA) http://www.di-mgt.com.au/public-key-crypto-discrete-logs-4-dsa.html