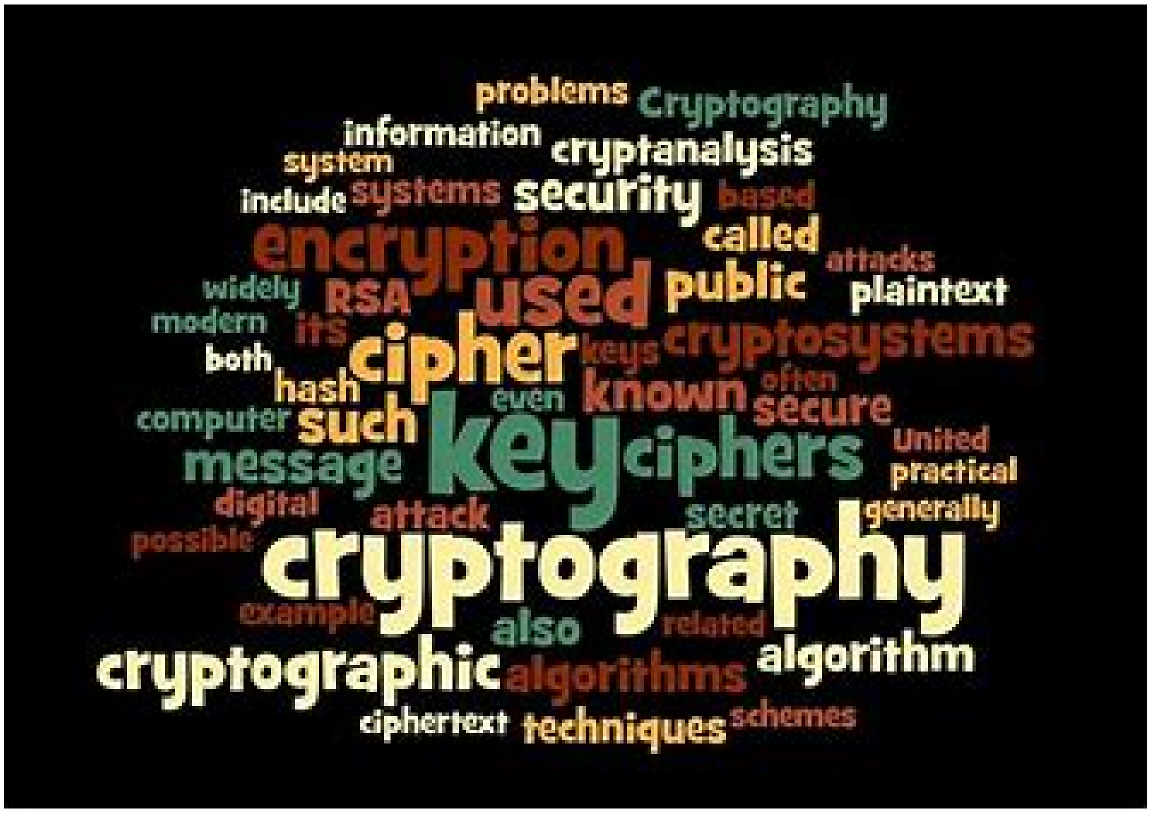
Understanding-Cryptography

Report

Mentee: Sree Vamshi Madhav Nenavath
Roll No: 23b1039



Contents

1 Introduction

Have you ever wondered why your social media accounts ask for two-factor authentication? Or how streaming services prevent unauthorized access to their content? How can you be confident that your online purchases are secure? The answer lies in cryptography, the science of securing information.

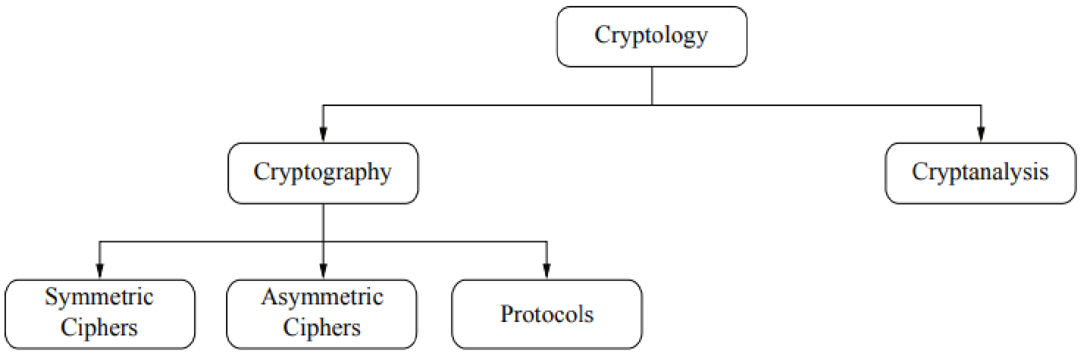
Cryptography, derived from the Greek words for "hidden writing," involves creating and deciphering codes to protect data. Its history is rich and varied, stretching back to ancient times when leaders used simple ciphers to safeguard their secrets to World War II where cryptography played a crucial role with machines like the Enigma, which the Allies eventually cracked, influencing the war's outcome.In today's digital age, cryptography is even more essential. It secures everything from your social media accounts to online streaming services.

This is a report on my learnings on Cryptography though **Understanding Cryptography by Prof. Christof Paar**

1.1 Cryptography

Cryptography is one of the two branches of **Cryptology** , whereas the other branch is **Cryptanalysis.**Cryptography is Further Divided as follows:

Cryptography is the science of secret writing with the goal of hiding the meaning of a message.
Cryptanalysis is the science and sometimes art of breaking cryptosystems.Cryptanalysis is of central importance for modern cryptosystems: without people who try to break our crypto methods, we will never know whether they are really secure or not.



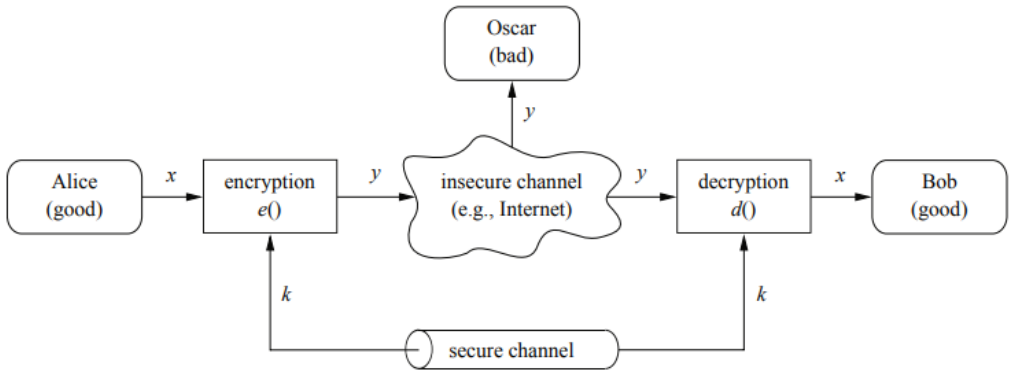
Symmetric Algorithms are what many people assume cryptography is about: two parties have an encryption and decryption method for which they share a secret key.

Asymmetric (or Public-Key) Algorithms In public-key cryptography, a user possesses a secret key as in symmetric cryptography but also a public key. Asymmetric algorithms can be used for applications such as digital signatures and key establishment, and also for classical data encryption.

Cryptographic Protocols Roughly speaking, crypto protocols deal with the application of cryptographic algorithms. Symmetric and asymmetric algorithms can be viewed as building blocks with which applications such as secure Internet communication can be realized.

1.2 Symmetric Cryptography.

Symmetric cryptographic schemes are also referred to as symmetric-key, secret-key,and single-key schemes or algorithms.Let us understand this by a simple example: Let's say Alice and Bob wants to discuss about a bussiness idea over an insecure channel(e.g: internet).The problem here is Oscar, the bad guy who has access to the channel can listen to the conversation between Alice and Bob.This type of hearing is called *eavesdropping.*We can prevent Oscar from eavesdropping by using symmetric cryptography.



x is *plaintext* or *cleartext*, y is called *ciphertext*, k is called the *key* and the set of all possible keys is the *keyspace.*
We can see that key is the only parameter Oscar is unaware of, if Oscar knows the key he can decrypt the message as the encryption and decryption algorithms are publicly known and tested. So, the key must be transmitted securely to both Alice and Bob.

1.3 Historical Ciphers

1.3.1 Substitution Cipher

As the name implies, we substitute each letter of the alphabet to a different letter.
Ex: $A \to k$; $B \to c$; by doing so ; the text **ABAB** is encrypted as **kckc** where the substitution pattern is only known by the sender and receiver.Here the substitution pattern acts as the **key**. And the **keyspace** is the no.of ways we can match the letters of the alphabet.

$$\text{key space} = 26.25.24....3.2.1 = 26!$$

The cipher looks fairly safe, but isn't. It does not hie the stastistical properties of the plaintext. One can easily break the cipher by doing a letter frequency analysis.A good cipher must hide the statistical propeties of the plaintext.

1.3.2 Shift Cipher(or Ceaser Cipher)

It is a special case of shift cipher and has an elegant mathematical description unlike substitution cipher.We simply shift every letter of the alphabet by a fixed number

of positions E.x: if A is mapped to b , then B to c and so on in a cyclic order.We encode the letters of the alphabet with numbers , A is 0 B is 1 and so on Z is 25. Both the plaintext letters and the ciphertext letters are now elements of the ring Z_{26}.Also, the key, i.e., the number of shift positions, is also in Z_{26}.
Encryption: $e_k(x) \equiv x + kmod26$
Decryption: $d_k(y) \equiv y - kmod26$
The shift cipher is not secure at all the key space is only 26. Brute force method can easily crack this.

1.3.3 Affine Cipher

This is an improvement of shift cipher ,we increase the size of the key space by adding a a new element **a** which also belongs to the ring Z_{26}, with the condition $gcd(a, 26) = 1$.The mathematical description of Affine cipher is :
Encryption: $e_k(x) \equiv a.x + kmod26$
Decryption: $d_k(y) \equiv a^{-1}.(y - k)mod26$

2 Stream Ciphers

Symmetric cryptography is divided into two blocks **Stream Ciphers** and **Block Ciphers**.Stream Ciphers encrypt bits individually whereas Block Ciphers encrypt chunks of bits at once.Stream Ciphers **encrypt bits individually** by adding a bit from **key stream** to the plaintext.Stream ciphers can be two types: synchronous and asynchronous. Encryption in synchronous strem cipher is dependent only on the key whereas in the asynchronous cipher it also depends on the ciphertext(a kind of feedback).

2.1 Encryption and Decryption with Stream Ciphers

How does stream ciphers actually encrypt individual bits? The answer is simple: each plaintext bit x_i is encrypted by adding a secret keybit s_i modulo 2.

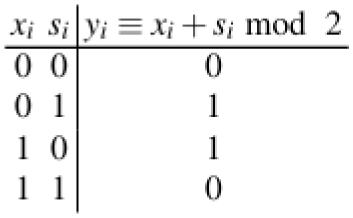
The plaintext, the ciphertext and the key stream consist of individual bits, i.e $x_i, y_i, s_i \in \{0, 1\}$
Encryption: $y_i = e_{s_i}(x) \equiv x_i + s_i mod2.$
Decryption: $x_i = d_{s_i}(y) \equiv y_i + s_i mod2.$

We can notice that both encryption and decryption are the same functions!!That is one of the property of modulo 2 addition.To prove that we must get x_i from the decryption function $x_i = y_i + s_i mod2$.

$$y_i = x_i + s_i mod2$$

$$d_{s_i}(y) = x_i + 2s_i mod2 = x_i mod2$$

where x_i can be 0 or 1 so $x_i mod2$ is x_i.From this fact we can relate the arithmetic modulo to the boolean algebra and logic gates like AND, NOR etc.By observing one can easily identify that **modulo 2 addition resembles the XOR operation**. One



x_i	s_i	$y_i \equiv x_i + s_i \bmod 2$
0	0	0
0	1	1
1	0	1
1	1	0

may wonder why do we use XOR , why not others, well it has a specific purpose.If

we look at the XOR table we observe that for any x_i ,y_i can be anything "0" or "1" depending on the value of s_i. No other operation has this type of property. If we consider an input of 10 bits , the probability of guessing the correct plaintext from ciphertext is very low $1/2^{10}$.
Everything is good until now but what about the key stream where do we get it from.

2.2 Key Stream

It can be observed that the security of a stream cipher is solely dependent on the key streamseries of s_i.It is obvious that a key stream should be random : like the outcomes of tossing of a coin.

2.2.1 Random Numbers and unbreakable Stream Cipher

. We undertood the importance of randomness in a keystream. let us have a look at Random Number Generators(RNGs).

True Random Number Generators(TRNGs) are based on physical processes like coin flipping, dice rolling, and radioactive decay. Their outputs are non-reproducible, making them suitable for cryptographic purposes such as generating session keys.

Pseudorandom Number Generators(PRNGs) compute sequences from an initial seed, making them deterministic and reproducible. Examples include the linear congruential generator and the ANSI C rand() function. PRNGs are used in simulations and testing due to their good statistical properties, even though they are not truly random.

$$s_0 = seed; s_{i+1} = f(s_i)$$

Cryptographically Secure Pseudorandom Number Generators (CSPRNGs) are a type of PRNG designed to be unpredictable. Given a sequence of n output bits, it is computationally infeasible to predict subsequent or preceding bits.This unpredictability is crucial for cryptographic applications but not necessary for other uses of PRNGs.

2.2.2 The One-Time Pad

*A cryptosystem is **unconditionally or information-theoretically secure** if it cannot be broken even with infinite computational resources.*

A stream cipher for which
1. the key stream s_0, s_1, s_2,...is generated by a true random number generator, and
2. the key stream is only known to the legitimate communicating parties, and
3. every key stream bit s_i is only used once is called a one-time pad.

The one-time pad is unconditionally secure. But are not suitable for practical usage because of large size and are not reusable. for ex: a 200MB file has a OTP of size 1.6GB.

2.2.3 Practical Stream Ciphers

OTPs are unconditionally secure but it has its drawbacks.What we now do is to use PRNGs to generate a key stream and give a **initial key** k as seed. for ex:

$$S_0 = seed; S_{i+1} \equiv AS_i + Bmodm$$

where m can be choosen(generally 100 bits).However if Oscar gets to know the first 3 'S' Values he can easily solve for A and B.That is why CSPRNGs are used , in CSPRNGs even if we know the $s_1 s_2 s_3 ... s_n$ we cannot calculate s_{n+1}

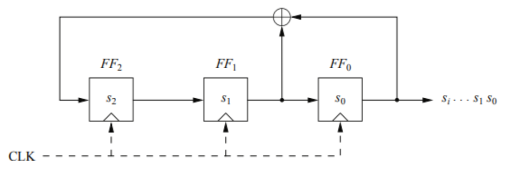
2.3 Shift-Register-Based Stream Ciphers.

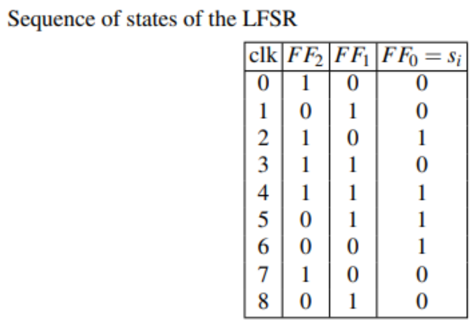
2.3.1 Linear Feedback Shift Registers (LFSRs)

We know that practical stream ciphers use a stream of key bits generated by a keystream generator, we can generate long pseudorandom sequences by using **Linear feedback shift registers(LFSRs)** which are hardware friendly.The sequence poduced b a single LFSR has some noticeable statistical properties and hence not so secure , but using a combination of LFSRs makes the cipher very secure.

An LFSR consists of storage elements(flip-flops) , a clock and a feedback path.The number of *flip-flops* denote the degree of a LFSR.The internal state bits are denoted by s_i and are shifted by one to the right with each clock tick.The initial set of states are feed as input(s_0, s_1, s_2) and other states are calculated using XOR gates and given input.
If we assume an initial state of ($s_2 = 1, s_1 = 0, s_0 = 0$), the table below shows the values of s_i aginst lock time. We can notice that after the 7th tick of the clock the values of s_i repeat .This means that the period of this LFSR is '7'.The output bit is computed as

$$s_{i+3} \equiv s_{i+1} + s_i mod2$$

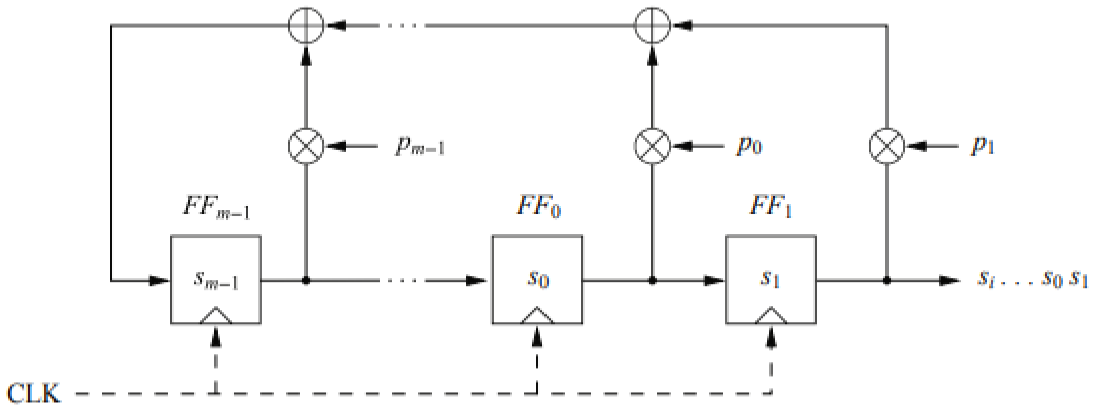



Sequence of states of the LFSR

clk	FF_2	FF_1	$FF_0 = s_i$
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	1	0	0
8	0	1	0

The general form of an LFSR of degree m is shown in Fig below .It shows m flip-flops and m possible feedback locations, all combined by the XOR operation. Whether a feedback path is active or not, is defined by the feedback coefficient p0, p1,..., pm-1. If $p_i = 0$ it is not active and if $p_i = 1$ it is active. So now , if we multiply the out of a flipflop with the p value we te result is either 0 or the s_i. as there are m flipflops they are feed with m input values $s_0 to s_{m-1}$. s_m is defined as

$$s_m \equiv s_{m-1}p_{m-1} + s_1p_1 + s_0p_0 mod 2$$



An LFSR can produce output sequences of various lengths depending on the input. **The maximum length of sequence produced by a LFSR of degree m is $2^m - 1$.**
An LFSR with a feedback coefficient vector (pm-1,..., p1, p0) is represented by the polynomial $P(x) = x^m + p_{m-1}x^{m-1} + ... + p_1x + p_0$.

3 The Data Encryption Standard (DES)

The *Data Encryption Standard* has been the most popular blockk cipher in the last 30 years.Nowadays DES is not considered very secure because of its samll key space(*56bits*), but still used in many legacy applications.The DES Algorithm proposed by IBM in early 1970's came into action in the late 1970's.

3.1 Confusion and Diffusion

These terms are proposed by the famous information theorist Claude Shannon.

Confusion is an encryption operation where the relationship between key and ciphertext is obscured. ***Diffusion*** is an encryption operation where the influence of one plaintext symbol is spread over many ciphertext symbols with the goal of hiding statistical properties of the plaintext. A simple diffusion element is the bit permutation.

The historical ciphers so far only had confusion property , the modern block ciphers are very good at diffusion property ,that is change in one plaintext bit should affect.

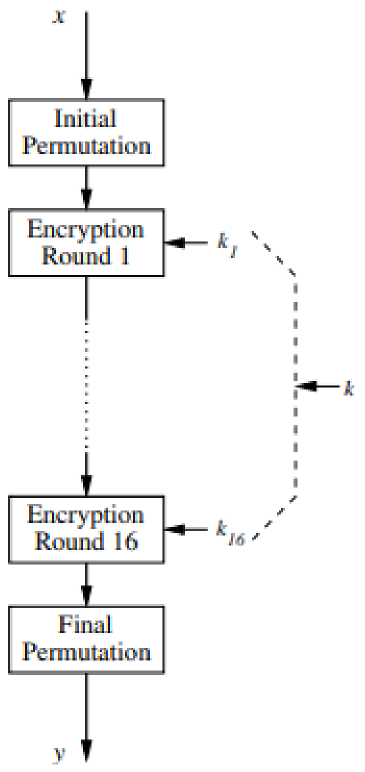
3.2 Overview of DES

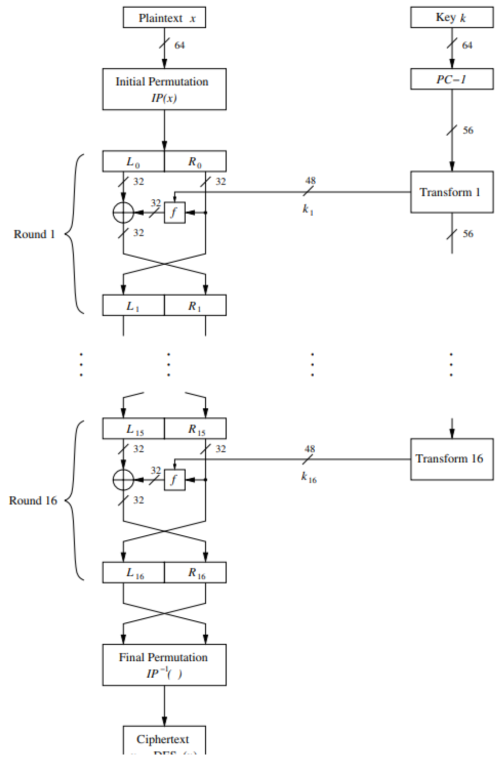
DES is a symmetric block cipher with block size of 64bits and key size of 56bits.since it is symmetric it uses the same key for decryption.For each block of plaintext encryption is done in 16 rounds with every round being identical.After the initial bitwise permutation IP of 64-bit the plain text x, is divided into 32 bits each $L_0 and R_0$.. The right half Ri is fed into the function f . The output of the f function is XORed (as usually denoted by the symbol $\oplus$)with the left 32-bit half Li. Finally, the right and left half are swapped. This process repeats in the next round and can be expressed as:

$$L_i = R_{i-1},$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$$

After 16 rounds L_{16} and R_{16} are again swapped and the step IP^{-1} is done. A round key k_i is derived from the initial key using a key schedule.If we keenly observe we notice that the fiestel structure first directly copies R0 without encryption but the $f - function$ takes it as Input and encrypts that means that the ciphertext of L0 is dependent on R0. This ensures that both **confusion and diffusion** are taking place.





3.3 Internal Structure of DES

3.3.1 The f-function

The f-function plays a very important role in the security of DES. in round i the inputs are $R_{i-1} and k_i$.The figure below describes what is happening in the f-fuction.the 32 bit input is expanded to 48 bits by mapping each 4 bit block of the input is mapped to 6 bits.This happens in **E-box**.

The 48 bit output is XORed with the 48 bit round key derived from main key.The resulting 48 bits are further divided into 8 6-bit blocks.These are fed in the **S-boxes** or **Substitution-Boxes**.These S-boxes map the 6 bit block to a 4 bit-block using a special algorithm.To know **more about S and E boxes click here**

3.3.2 Key Schedule

Now Let us discuss how do we get the k_i from the main key. The *Key Schedule* Derives 16 round keys or *subkeys*.The actual input size of key for DES is 56 bits even though the input is 64 bits it ignores every 8th bit.They are called parity bits, the

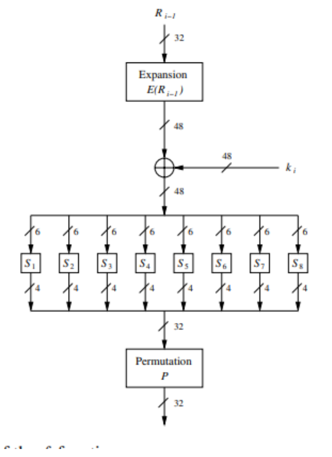


Figure 3: the f-function

pariy bits are stripped in the **PC-1(permuted choice -one)** step.DES is **not a 64-bit** cipher it **is a 56-bit cipher**.
The resulting 56-bit key is split into two halves C_0 and D_0, and the actual key schedule starts as The two 28-bit halves are cyclically shifted, i.e., rotated, left by one or two bit positions depending on the round i according to the following rules:

- In rounds $i = 1, 2, 9, 16$ the two halves are rotated left by one bit.
- In the other rounds ,the two halves are rotated left by two bits

The total number of rotation positions is 4.1+12.2 = 28 this implies $C_0 = C_{16} and D_0 = D_{16}$ which is necessary for decryption.To derive the 48-bit round keys k_i, the two halves are permuted bitwise again with PC -2 which will ignore 8 bits from C_i and D_i.

3.4 Decryption

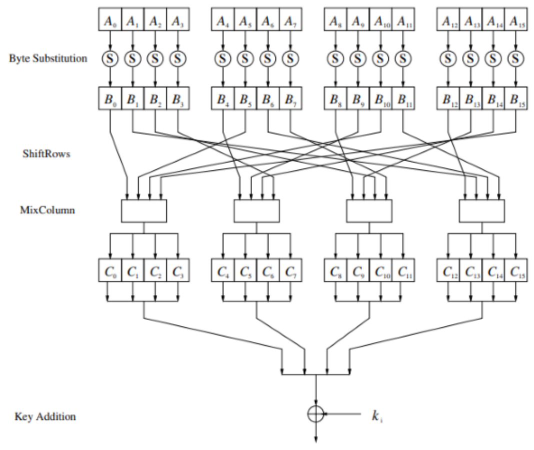
So far we have learned about encryption Using DES , Now we'll look at decryption.One amazing thing about DES is that the encryption and decryption are the same.This is one of the elegant properties of fiestel network.The decryption function reverses the DES encryption in a round-by-round manner. That means that decryption round 1 reverses encryption round 16, decryption round 2 reverses encryption round 15, and so on.

4 The Advanced Encryption Standard (AES)

DES was secure initially but because of its small key space, It was broken by brute force attack in a few days. 3DES which an improvement of DES was secure but it had some issues.So there was a need for a better Cipher that is AES.
AES is a block cipher with 128 bits input size and can accept key lengths of 128,192,256 bits.It consists of layers **Key Addition Layer** which XORs the current state and subkey. **Byte Substitution Layer** acts as a *confusion layer* where each element of the state is nonlinearly transferred based on a special table.**Diffusion Layer** which further consists of wo layers : *Shift Rows(at byte level) and Mix columns(at a block of 4 bytes).*

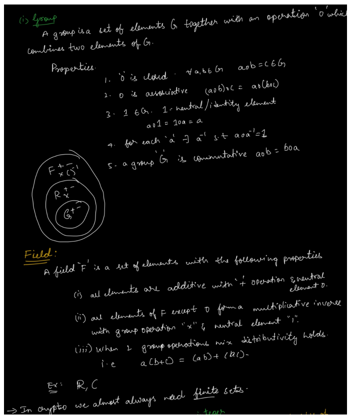
4.1 Internal Structure of AES

The fig illustrates one of the rounds invloved in encryption of the AES.The 16-byte input is fed in byte by byte as $A_0, A_1, A_2, ..., A_15$.The 16 byte output $B_0, B_1, B_2, .., B_15$ is permuted bype wise in the **shift rows** layer each block of 4 bytes is connected to all the 4 blocks in the next layer as to maximise diffusion, i.e; change in one B will affect all the others.And finally the 128bit subkey is XORed with the intermediate output.



The Advanced Encryption Standard (AES) operates on a 128-bit data block arranged as a four-by-four byte matrix. For example, a state A consisting of 16 bytes A0,A1,. . . ,A15 is arranged as follows:

4.2 Math Required



(i) Group

A group is a set of elements G together with an operation '$\circ$' which combines two elements of G.

Properties.

1. '$\circ$' is closed. $\forall a,b \in G \quad a \circ b = c \in G$
2. $\circ$ is associative $(a \circ b) \circ c = a \circ (b \circ c)$
3. $1 \in G$. 1 = neutral/identity element $a \circ 1 = 1 \circ a = a$
4. for each 'a' $\exists\, a^{-1}$ s.t $a \circ a^{-1} = 1$
5. a group 'G' is commutative $a \circ b = b \circ a$

$F\ {}^{+\,-}_{\times\,()^{-1}}$

$R\ {}^{+\,-}_{\times}$

$G\ {}^{+\,-}$

Field:

A field 'F' is a set of elements with the following properties

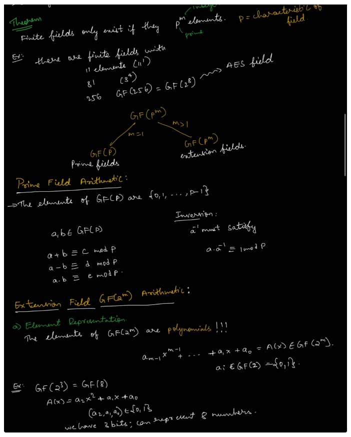
(i) all elements are additive with '$+$' operation & neutral element 0.

(ii) all elements of F except 0 form a multiplicative inverse with group operation "$\times$" & neutral element "1".

(iii) when 2 group operations mix distributivity holds. i.e $a(b+c) = (ab) + (ac)$.

Ex: $\mathbb{R}, \mathbb{C}$

→ In crypto we almost always need finite sets.



Theorem

Finite fields only exist if they p^m elements. p = characteristic of field.

Ex: there are finite fields with
11 elements (11^1)
81 (3^4)
256 $GF(256) = GF(2^8) \rightsquigarrow$ AES field

$$\begin{tikzcd} & GF(p^m) \arrow[dl, dash, "m=1"'] \arrow[dr, dash, "m>1"] & \\ GF(p) & & GF(p^m) \end{tikzcd}$$

Prime fields — extension fields.

Prime Field Arithmetic:

→ The elements of $GF(p)$ are $\{0, 1, \ldots, p-1\}$

$a, b \in GF(p)$

$a + b \equiv c \bmod p$
$a - b \equiv d \bmod p$
$a \cdot b \equiv e \bmod p$.

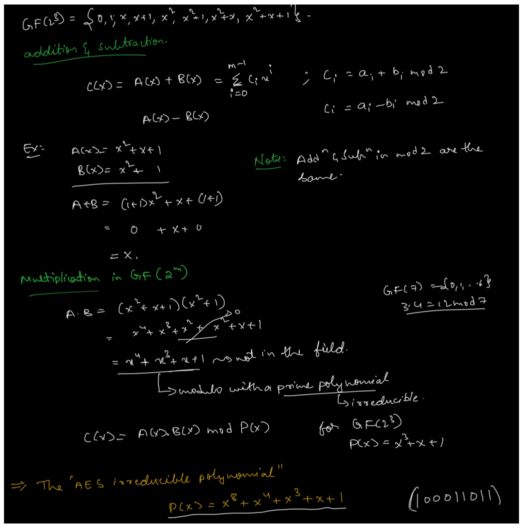
Inversion:
a^{-1} must satisfy
$a \cdot a^{-1} \equiv 1 \bmod p$

Extension Field $GF(2^m)$ Arithmetic:

a) Element Representation.
The elements of $GF(2^m)$ are polynomials !!!

$a_{m-1}x^{m-1} + \cdots + a_1 x + a_0 = A(x) \in GF(2^m)$.
$a_i \in GF(2) = \{0, 1\}$.

Ex: $GF(2^3) = GF(8)$
$A(x) = a_2 x^2 + a_1 x + a_0$
$(a_2, a_1, a_0) \in \{0, 1\}$
we have 3 bits; can represent 8 numbers.



$GF(2^3) = \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$.

addition & subtraction.

$C(x) = A(x) + B(x) = \sum_{i=0}^{m-1} c_i x^i \quad ; \quad c_i = a_i + b_i \bmod 2$

$A(x) - B(x) \qquad c_i = a_i - b_i \bmod 2$

Ex: $A(x) = x^2 + x + 1$
$B(x) = x^2 + 1$

$A+B = (1+1)x^2 + x + (1+1)$
$= 0 + x + 0$
$= x$.

Note: Add^n & Sub^n in mod 2 are the same.

Multiplication in $GF(2^m)$

$GF(7) = \{0, 1, \ldots 6\}$
$3 \cdot 4 = 12 \bmod 7$

$A \cdot B = (x^2 + x + 1)(x^2 + 1)$
$= x^4 + x^3 + \cancel{x^2 + x^2} + x + 1$
$= x^4 + x^3 + x + 1 \rightsquigarrow$ not in the field.
↳ modulo with a prime polynomial
↳ irreducible.

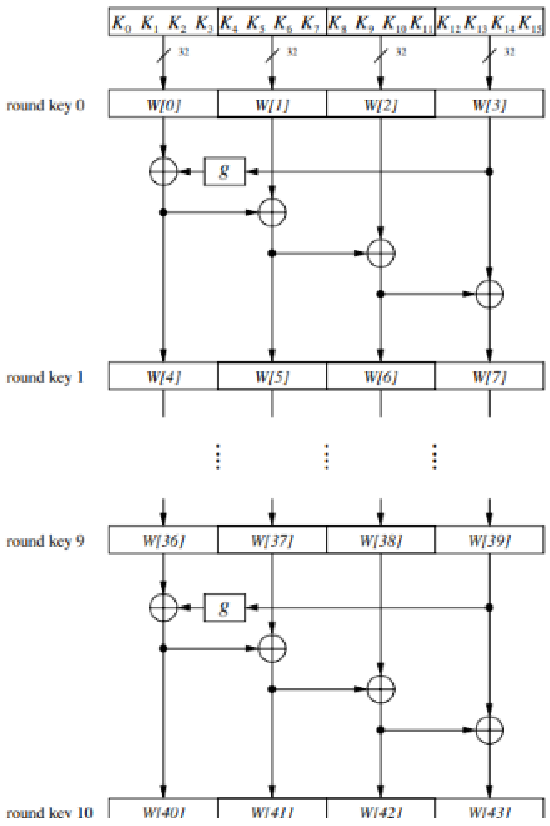
$C(x) = A(x) \cdot B(x) \bmod P(x)$ for $GF(2^3)$ $P(x) = x^3 + x + 1$

⇒ The "AES irreducible polynomial"
$P(x) = x^8 + x^4 + x^3 + x + 1$ (100011011)

4.3 Key Schedule

The key schedule takes the original input key (of length 128, 192 or 256 bit) and derives the subkeys used in AES.The number of subkeys is equal to the number of rounds plus one, due to the key needed for key whitening in the first key addition layer.Thus, for the key length of 128 bits, the number of rounds is $n_r = 10$, and there are 11 subkeys, each of 128 bits. The AES with a 192-bit key requires 13 subkeys of length 128 bits, and AES with a 256-bit key has 15 subkeys. The AES subkeys are computed recursively, i.e., in order to derive subkey k_i, subkey k_{i-1} must be known, etc.

4.3.1 Key Schedule for 128-Bit Key



The ll subkeys are stored in a key expansion array with the elements W[0],...,W[43].The first element is the main key itself. The other elements are computed as follows.

$$W[4i] = W[4(i-1)] + g(W[4i-1]).$$

$$W[4i+j] = W[4i+j-1] + W[4(i-j)+j]$$

To know more about this refer

4.4 Decryption

Because AES is not based on a Feistel network, all layers must actually be inverted, i.e., the Byte Substitution layer becomes the Inv Byte Substitution layer,the ShiftRows layer becomes the Inv ShiftRows layer, and the MixColumn layer becomes Inv MixColumn layer. However, as we will see, it turns out that the inverse layer operations are fairly similar to the layer operations used for encryption. In addition the order of the subkeys is reversed, i.e., we need a reversed key schedule.

5 Public Key Cryptography

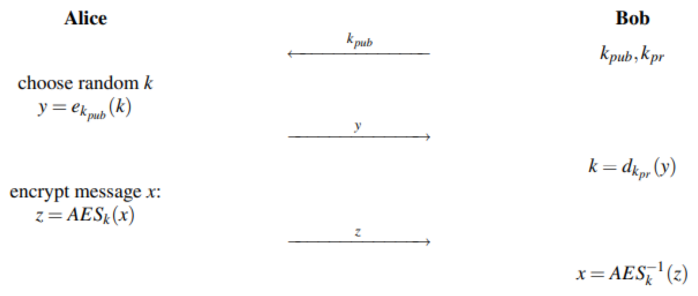
The term Public Key Cryptography is also referred as ASYMMETRIC CRYPTOGRAPHY.Symmetric cryptography was being used for more than 4000 years whereas asymmetric cryptography was proposed in around late 1970's.But why did we need a different algorithm other than symmetric cryptography, let's see:

The *same key* is used for both encryption and decryption. The encryption and decryption functions are very similar.We'll now look at a simple analogy . Consider a locker which has two keys one with Alice and the other with Bob.If Alice wants to send a message to Bob , Alice opens the locker encrypts the message and later Bob uses a similar key to decrypt or open the message.There are some shortcomings

Key Distribution The key must be established between Alice and Bob using a secure channel, Which is a problem. **Number of keys** with the number of users increasing creating and storing the key pairs is a hustle. **No Protection Against Cheating** For instance, in e-commerce applications it is often important to prove that Alice actually sent a certain message, say, an online order for a flat screen TV. If we only use symmetric cryptography and Alice changes her mind later, she can always claim that Bob, the vendor, has falsely generated the electronic purchase order. Preventing this is called *nonrepudiation* and can be achieved with asymmetric cryptography.

Principles of Asymmetric Cryptography

In order to overcome the drawbacks of symmetric ciphers, Scientists proposed an idea: It is not necessary that the *key* used for encryption is secret.The important pat is the receiver,Bob can **only decrypt using a personal key**. Thus, Bob's key k consists of two parts k_{pub} for encryption and a matching secret k_{pr} for decryption.



The discussed protocol allows for secure message encryption without a secret channel for key establishment by encrypting a symmetric key (e.g., AES key) using a public-key algorithm. Once Bob decrypts the symmetric key, both parties can use it for faster encryption and decryption of messages using symmetric ciphers. This approach is illustrated in Figure above and is more efficient than using an asymmetric algorithm for the entire payload. Asymmetric cryptography, based on one-way functions, is essential for security applications, with practical schemes introduced in subsequent chapters.

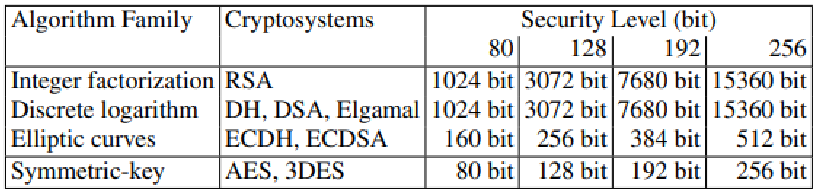
5.1 Practical Aspects

5.1.1 Important Public Key Algorithms

As there are different types of symmetric ciphers there are also different types of asymmetric ciphers.Asymmetric ciphers majorly rely on mathematics, which is opposite of symmetric ciphers where more of electrical components like S-boxes are used.Several Popular Public Key Cryptographic algorithms include **Integer Factorisation Schemes**, **Discrete Logarithm Schemes** and **Elliptic Curve(EC) Schemes**

5.1.2 Key Lengths and Security Levels

An algorithm is said to have a "security level of n bit" if the best known attack takes 2^n steps to crack it.Having a key length of 'n' bits does not imply security level of n bits.The following table shows the key lengths vs security level of asymmetric ciphers.



Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

5.2 Number Theory for Public-Key Algorithms

Euclidean Algorithm

This is about finding the *greatest common divisor(gcd)* of two numbers $r_0 and r_1$. For small numbers $r_0 and r_1$ gcd can be found directly by factorisation , gcd is the product of all common primes.But what about larger numbers which are almost impossible to factorise . This is where Euclidean Algorithm come into action.We **assume that** $r_0 > r_1$

$$gcd(r_0, r_1) = gcd(r_0 - r_1, r1) = gcd(r_0 - 2r_1, r_1) = ... = gcd(r_0 - m.r_1, r_1)$$

$$gcd(r_0, r_1) = gcd(r_0 mod r_1, r_1)$$

Thus gcd for very large numbers can also be computed using this algorithm.

Extended Euclidean Algorithm

An extension of the algorithm allows us to compute modular inverses, which is of major importance in public-key cryptography. In addition to computing the gcd, the *extended Euclidean algorithm* (EEA) computes a linear combination of the form:

$$gcd(r_0, r_1) = s.r_0 + t.r_1$$

where s,t are integer coefficients.Now how do we compute s and t.we reprsent the current remainder r_i linearly in the form:

$$r_i = s_i.r0 + t_i.r_i$$

On doing so at the last iteration we end up at

$$r_l = gcd(r_0, r_1) = s_l.r_0 + t_l.r_1 = s.r_0 + t.r_1$$

If we carefully observe we use the result of the $i - 1$ th iteration to link $r_i to r_0 and r_1$. The main goal of EEA is finding inverse. We know that inverse exists only if $gcd(r_0, r_1)$ is 1. From this gcd(r_0, r_1) = 1. that implies

$$s.r_0 + t.r_1 = 1$$

by applying mod r_0 on both sides, The LHS becomes t.r1 which is the remainder.

$$t.r_1 = 1 mod r_0$$

$$t = r_1^{-1} modr r_0$$

t is the inverse of r_1!!!

5.3 Theorems

Euler's Phi function

The number of integers in Z_m which are co-primes with m is called the phi function $\phi(m)$.

Theorem

let m have the following canonical factorization m=$p_1^{e_1}.p_2^{e_2}....p_n^{e_n}$ then $\phi(m) = \Pi(p_i^{e_i} - p_i^{e_{i-1}})$.

Fermat's Little Theorem

Let a be an integer and p be a prime then $a^p \equiv a(modp)$

Euler's Theorem

Let a and m be integers with gcd(a,m) =1, then $a^{\phi(m)} = 1(modm)$.

6 The RSA Cryptosystem

The RSA crypto scheme, also referred to as Rivest-Shamir-Adleman algorithm is currently the most widely used asymmetric cipher.RSA is comparitively slower than Symmetric ciphers.This acts like a **one-way function**. It is solely dependent on arithmetic. Multiplying two primes is very easy whereas factorizing large numbers is very difficult.Euler's phi function is very important in RSA.

6.1 Key Generation

Unlike symmetric ciphers , Public Key Algorihms require the computation of pair (K_{pub}, K_{pr}).

- Choose large primes p,q.
- $n = p.q$
- $\phi(n) = (p-1).(q-1)$
- Choose $K_{pub} = e \in 1, 2, ..., \phi(n) - 1$ such that $gcd(e, \phi(n)) = 1$.
- Compute $K_{pr} = d, s.td.e \equiv 1 mod \phi(n)$

6.2 Encryption and Decryption

Public Key (n,e) = k_{pub}; Private Key (d) = $K_p r$

Encryption $y = e_{k_{pub}}(x) \equiv x^e mod n$
Decryption $x = d_{k_{pr}}(y) \equiv y^d mod n$

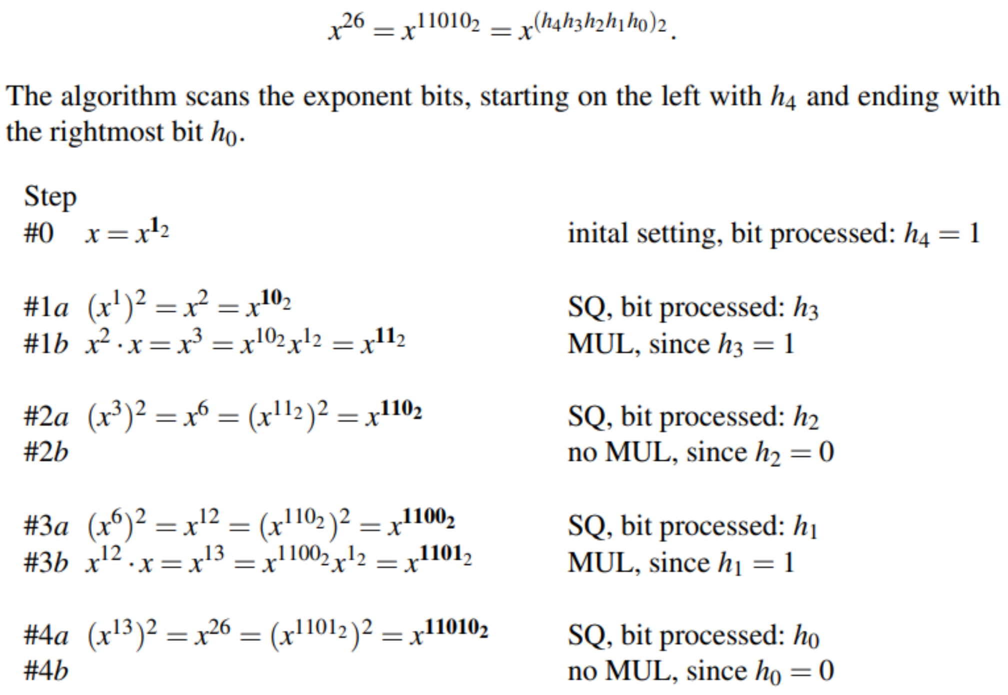
Real world RSA Parameters are very large $p, q \geq 2^{512}$ and $n \geq 2^{1024}$.

6.2.1 Fast Exponentiation

We learned that encryption and decryption in RSA requires heavy computational resources. So there was a need for efficient exponentiation of numbers. for ex: to calculate x^4 we can do $x.x = x^2$
$x^2.x = x^3$
$x^3.x = x^4$
or simply after calculating x^2 we can square it to find x^4.

The second method is way more efficient for large exponents. For exponents of the form 2^n it is very efficient to keep on squaring. But what if the exponent is not a power of 2 or not an even number at all.We can use the **square-and-multiply algorithm** or **binary method.**

The algorithm is based on scanning the bit of the exponent from the left (the most significant bit) to the right (the least significant bit). In every iteration, i.e., for every exponent bit, the current result is squared. If and only if the currently scanned exponent bit has the value 1, a multiplication of the current result by x is executed following the squaring.
Sounds confusing right? Let us understand it with the help of an example , consider x^{26}.



$$x^{26} = x^{11010_2} = x^{(h_4h_3h_2h_1h_0)_2}.$$

The algorithm scans the exponent bits, starting on the left with h_4 and ending with the rightmost bit h_0.

Step		
#0	$x = x^{\mathbf{1}_2}$	inital setting, bit processed: $h_4 = 1$
#1a	$(x^1)^2 = x^2 = x^{\mathbf{10}_2}$	SQ, bit processed: h_3
#1b	$x^2 \cdot x = x^3 = x^{10_2}x^{1_2} = x^{\mathbf{11}_2}$	MUL, since $h_3 = 1$
#2a	$(x^3)^2 = x^6 = (x^{11_2})^2 = x^{\mathbf{110_2}}$	SQ, bit processed: h_2
#2b		no MUL, since $h_2 = 0$
#3a	$(x^6)^2 = x^{12} = (x^{110_2})^2 = x^{\mathbf{1100_2}}$	SQ, bit processed: h_1
#3b	$x^{12} \cdot x = x^{13} = x^{1100_2}x^{1_2} = x^{\mathbf{1101}_2}$	MUL, since $h_1 = 1$
#4a	$(x^{13})^2 = x^{26} = (x^{1101_2})^2 = x^{\mathbf{11010_2}}$	SQ, bit processed: h_0
#4b		no MUL, since $h_0 = 0$

6.3 Finding Large Primes

We have to find sufficiently large prime number p, q which is not an easy task.the general approach is to generate integers randomly and check for primality. The

RNG(random number generator) should be unpredictable because the attacker should not be able to notice any pattern after observing many testcases.There are some points to notice while doing this

- How many random integers do we have to test before we have a prime? (If the likelihood of a prime is too small, it might take too long.)
- How fast can we check whether a random integer is prime? (Again, if the test is too slow, the approach is impractical

How Frequent are Primes?

As we notice the prime number sequence , the frequency becomes less as we go through large numbers. But we only check odd numbers which almost doubles the chance . According to the prime number theorem the probability of a large number p being prime is

$$P(p) \approx \frac{2}{ln(p)}$$

Primality Tests

Practical primality tests are **probabilistic algorithms** i.e , they are not true always. If the output says the number is composite , it is true but if the output says the number is prime It **may be true**.There are various well-known tests for primality are Fermat-Primality test , Miller-Rabin Primality Test etc.

6.4 RSA in Practice: Padding

The RSA we have defined so far has several weaknesses ; It is **deterministic** i.e; for a particular key, a particular plaintext is always mapped to a particular ciphertext. **Small public exponents** e and small plaintexts x might be subject to attacks if no padding or weak padding is used. and there are several other weaknesses as well.A possible solution of all these problems is the use of **padding**, which embeds a random structure e into the plaintext before encryption and avoids the above mentioned problems. To know more about Padding and how is it done visit https://en.wikipedia.org/wiki/RSA_(cryptosystem)