Evolution of Crytography and the Math behind it

Sai Krishna

Contents

1 Cryptography

Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries.[1]

The minimal information unit in cryptography is called an **alphabet**.

The usual cast for the Cryptography play:

- Good guys: *Alice*, *Bob*, *Carol*
- Bad guys: *Eve*, *Mallory*
  Eve can only read messages shared between the good guys
  Mallory can read and change the messages shared between the good guys

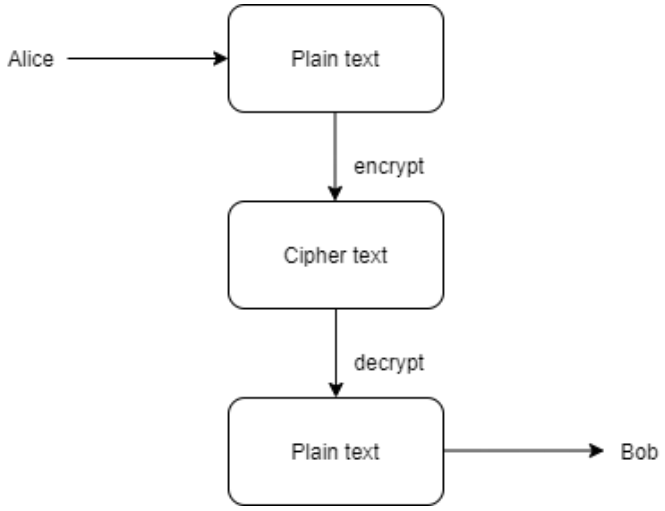


Figure 1: Big picture of Cryptography

The encrypted message is called **cipher text**.[2]

Possible threats in the above case:

1. Adversary reading our message
2. Adversary changing our message
3. Adversary replying to the message in place of Bob.

Main goals of information security:

1. **Confidentiality:** Can't read our messages

[1] In cryptography, an adversary is a malicious entity whose aim is to prevent the users of the cryptosystem from achieving their goal.

[2] A cipher is a pair of algorithms that create the encryption and the reversing decryption

2. **Integrity:** Can't change our messages

3. **Authenticity:** Can't forge our messages

To ensure that only the person to whom the message was sent must be able to decrypt it, and not the bad guys. It should be ensured that the good guy has some extra information which the bad guy does not have. The minimum amount of information that the good guy must have which the bad guys dont is called **Message KEY.**

cryptanalysis: The study of how to crack encryption algorithms without the key.

1.1 Kerckhoffs' design principles

- System should not have complex rules
- Algorithms should not be secret
- Security should rely on simple keys

1.1.1 Kerckhoffs' principle of cryptography

Stated by **Auguste Kerckhoffs** in 19^{th} century.

> A cryptosystem should be secure even if everything
> about the system, except the key, is public knowledge.

This was further reformulated by **Claude Shannon** and is called **Shannon's maxim**

> One ought to design systems under the assumption that
> the enemy will immediately gain full familiarity with them.

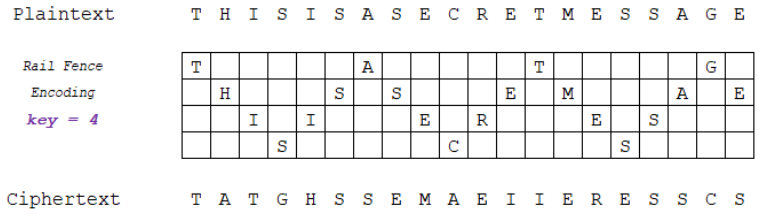
1.2 Classic cryptography

Classical cryptography relied on secret algorithms, that is for the message to be protected the entire algorithm used to encode it should be protected (goes against Kerckhoffs' principle). This is because these codes could be cracked using brute force techniques easily once the algorithm is known, and this is called **security through obscurity**. The main classical cipher types are:

1.2.1 Transposition ciphers

Rearrange the letters of the word
Ex: hello $\rightarrow$ ehlol

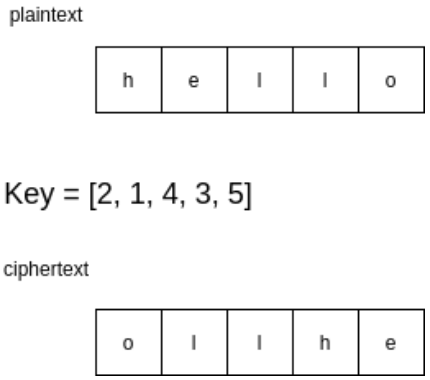
- **Rail Fence Cipher** The plaintext alphabets are listed diagonally over a number of rows, and then retrieve alphabets row by row.



  Key is the number of rows used for diagonally writing the message.

- **Permutation cipher** Plaintext alphabets are listed row by row and retrieved column by column.
  Key determines the column order and is a permutation of a set of size n. Length of the key determines the number of columns to be used. Length of the message determines the number of rows used.

  NOTE: If the length of the message is not divisible by length of the key. Then there will be some empty elements in the matrix. A particular character can be predetermined to fill these empty spaces.



  With a keylength = n, the size of the keyspace is n!

The frequency distribution of the plain text and cipher text will be the same in case of transposition cipher, hence these kind of ciphers are susceptible to frequency analysis attacks.

The other attacks to which these cipher are vulnerable are known plain-text attack and chosen plain-text attack.

1.2.2 Substitution ciphers

Systematically replace the letters or group of letters with other letters or group of letters.
Ex: hello → gdkkn
(replace letter by its previous letter)

- **Caesar cipher** was one of the famous Substitution. This used a shift of x
  Ex: x = 3 then a → d, b → e
  NOTE: If x = 26, ciphertext = plaintext

  Size of key space is equal to the size of the plain text alphabet.

- **Mary Queen of Scots** used a symbol as a replacement for each alphabet. In addition 5 symbols were used which map to NULL. And some other symbols which maps to an entire word. This was used to increase the security, and confuse the adversaries.

The above two techniques come under **Monoalphabetic ciphers** i.e. each letter maps to only one possible letter/symbol in the cipher text. To further increase the complexity of the code **Polyalphabetic ciphers** are used. *Vigenere cipher* and the *Enigma machine* are the best known examples for polyalphabetic ciphers.

- **Alberti Cipher**
  Created in 1467 by **Leon Battista Alberti** this was one of the first polyalphabetic cipher.

  The process of encrypting into the Alberti cipher is simplified by Alberti's discs.

  - To encrypt, the disk is set in one position, the initial shift (which can be zero) corresponds to the number of letters shifted at the beginning.
  - Each letter of the plain text found on the outer ring, is replaced by the corresponding one (the one aligned) in the inner ring.
  - By default, every 4 characters (4 = period), the disk is rotated clockwise of 1 letter (1 = periodic increment), this changes the substituting alphabet.

Figure 2: Alberti Cipher Disk

- **Trimethius Cipher**
  The *trimethius cipher* was published by **Johannes Trimethius** in his book *Polygraphia*, which is credited with being the first published work on cryptology.

  This cipher uses the tabula recta to define a polyalphabetic cipher.*tabula recta* and is a square table of alphabets, each row of which is made by shifting the previous one to the left. The table has 26 rows and each row creates a different Caesar cipher.
  This cipher lacks a key and thus violates Kerckhoffs' principle of cryptology

  Ex:

  Message: helloworld
  Cipher text: hfnosbuytm

  The encryption uses successive shifts based on the index of the letter in the plain text. In the above example h is shifted by 0 places to get h, e is shifted by 1 place to get f, and similarly proceeding at the end d is shifted

by 9 places to get m.

	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

Figure 3: tabula recta

An important extension to Trimethius' method was developed by **Giovan Battista Bellaso** and is now called Vigenere cipher.

- **Vigenere Cipher**

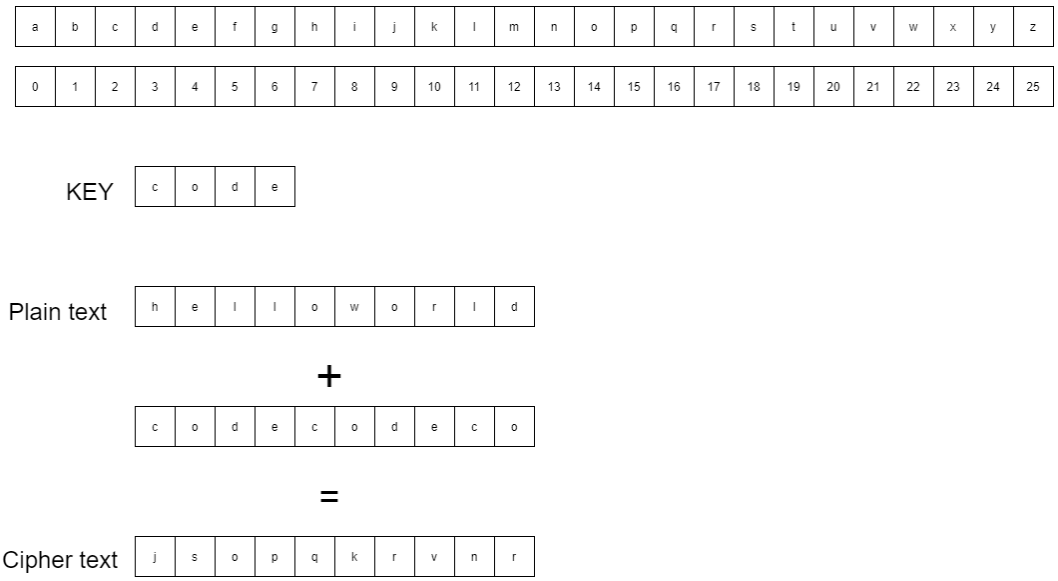
$$\textbf{Encryption: } C_i = (p_i + k_{i \bmod m}) \bmod 26$$

c - cipher text
p - plain text
k - key
m - length of the key
i - index

Size of the key space is n^m where n is the number of alphabets.

1. By adding letters:

   Message: helloworld
   Key: code
   Cipher text: jsopqkrvnr
   Assign a number to each letter, then repeat the key to match the size of the plain text, and then do modulo 26 addition of the corresponding elements from the repeated key and the plain text to get cipher text.



2. Using a table

   Locate the first letter of the plaintext message in the first line of the tabula recta and the first letter of the key on the left column. The cipher letter is at the intersection.

   Ex:
   plain text: helloworld
   key: code
   cipher text: jsopqkrvnr

- **Autokey Cipher**
  Was invented in 1586 by Blaise de Vigenère. This is a cipher that incorporates the message into the key. There are two forms of Autokey Cipher

  – **key-autokey cipher**
    A key-autokey cipher uses previous members of the keystream to determine the next element in the keystream.

– **text-autokey cipher**
A text-autokey uses the previous message text to determine the next element in the keystream.

Basically it starts with a primer and then appends the few characters of the text to the primer and this is the key. With this new key use Vigenere cipher to encrypt.
Ex:
plaintext: helloworld
primer: code
key: codehello
ciphertext: jsopvazczf

Cipher texts produced by classical ciphers will reveal statistical information about the plain text, and that information is sufficient to crack the code.
Language letter frequencies were very helpful to crack few of the above mentioned classical ciphers.

To increase the difficulty of frequency analysis attacks the following ciphers were used which would flatten the frequency distribution.

1.2.3 Homophonic Ciphers

Plain text letters map to more than one cipher text symbols. Usually the highest frequency plain text symbols are given more equivalents than lower frequency letters.
This technique requires the cipher text to have more than 26 symbols. However, language letter group frequencies can be used to crack such codes.

1.3 Codes and Ciphers

- Codes generally operate on *semantics*, meaning, while ciphers operate on *syntax*, symbols.
- A code is stored as a mapping in a codebook, while ciphers transform individual symbols according to an algorithm.
- Its hard to have a code for all possible words, but with ciphers we can easily generalize it to any word.
- **Morse code** is a cipher because it encrypts each letter separately.

1.4 Machine Ciphers

The main drawback of the classical ciphers is that, even without any knowledge of the key, the cipher text could still be decrypted by **brute force**. This is because the key space was small. To overcome this machine ciphers were used

where the key space was very huge and to use brute force to crack the code takes years even on the fastest computers.

Most of the classical ciphers either implement transposition or substitution, the machine ciphers implement the combination of these two which is called **product cipher**.

Ex: In a *fractionation system*,[3] a substitution is first made from symbols in the plaintext to multiple symbols in the ciphertext, which is then superencrypted by a transposition.

1.4.1 Wheel ciphers

Introduced by Thomas Jefferson in 1795. A wheel cipher consists of a set of several disks with permutations of alphabet symbols arranged around their edges. The permutations are different for all disks. With a dozen or even tens of such disks stacked on one axle.

Figure 4: Wheel cipher

The encryption process is based on reading a text placed one or more rows above or below. A person decrypting the text arranges consecutive letters of the cryptogram in one row of the wheel and reads the plaintext from the appropriate row. If 10 discs are stacked together then the size of kep space is 10!

[3]Fractionation is a method of splitting letters so that each plaintext letter is represented by two or more symbols.

1.4.2 Rotor Machine

A rotor machine is an electro-mechanical stream cipher[4] device used for encrypting and decrypting messages.

A rotor is a small disk of insulating material, with perhaps 26 equally-spaced electrical contacts in a circle on each side. The contacts on one side are connected to the contacts on the other side in a scrambled order. Hence a rotor machine implements a basic substitution cipher. But the rotor rotates after every key press, hence the substitution cipher used varies for each letter. Thus making the key space large and cracking the code hard.

Consider a simple rotor machine for 4 alphabets (A, B, C, D). Once a key is pressed the rotor on the right side rotates, the wiring that connects the rotor on left and right side remains the same. In the following example using a simple rotor the plaintext **DAD** is encoded as **BAD**

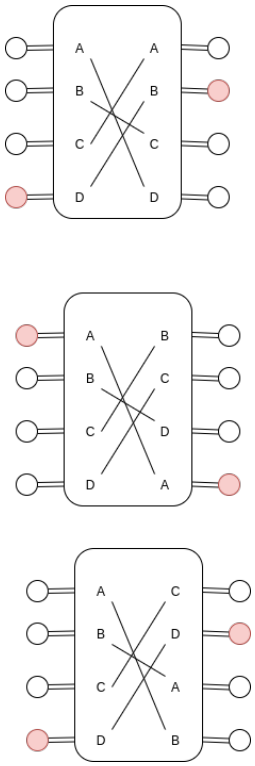


Figure 5: Rotor machine

[4]A stream cipher is a symmetric key cipher where plaintext digits are combined with a pseudorandom cipher digit stream (keystream).

The substitution cipher implemented before the any key press is

$$\begin{aligned} A &\rightarrow D \\ B &\rightarrow C \\ C &\rightarrow A \\ D &\rightarrow B \end{aligned}$$

Once the key D is pressed the substitution cipher used changes because one of the rotor rotates, the substitution cipher now used is

$$\begin{aligned} A &\rightarrow A \\ B &\rightarrow D \\ C &\rightarrow B \\ D &\rightarrow C \end{aligned}$$

Once the key A is pressed the new substitution cipher used is

$$\begin{aligned} A &\rightarrow B \\ B &\rightarrow A \\ C &\rightarrow C \\ D &\rightarrow D \end{aligned}$$

Thus for a simple 3 letter word, 3 different ciphers are used, hence cracking such code gets more difficult. The complexity can be further increased by stacking multiple rotors one after the other.

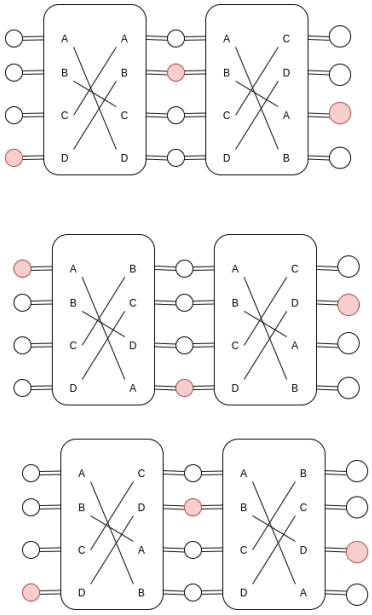


Figure 6: Cascaded Rotor machine schematic

By cascading we observe that the key space becomes extremely large by considering the fact that there are two rotors now and they can be rotated at different frequencies. To decrypt messages encrypted using cascaded rotor machines, the initial setting of the rotor and the frequency at which the rotors rotate must be shared.

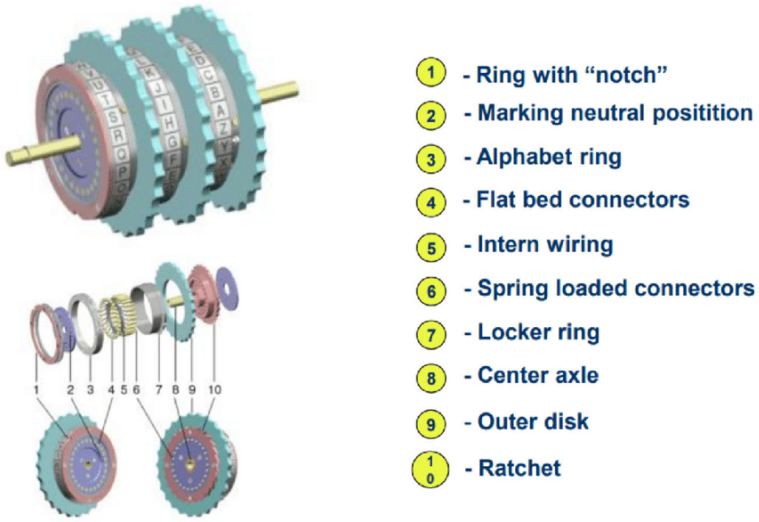


Figure 7: Cascaded Rotor machine

1.4.3 ENIGMA MACHINE

The Enigma machine is an encryption device developed and used in the early- to mid-20th century. It was employed extensively by Nazi Germany during World War II. The version of Enigma as used by Nazis has approximately $2^{67.1}$ possible settings. Hence it is nearly impossible to break this code using brute force

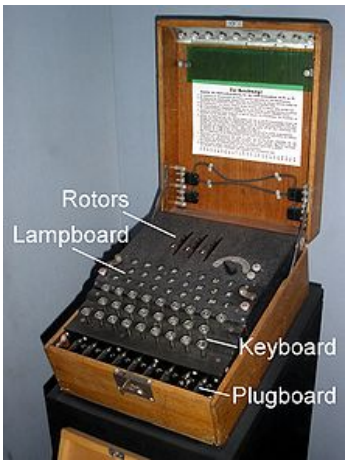


Figure 8: Enigma machine

The major drawback of the enigma machine is that it used a reflector, which would ensure that no letter would get mapped to itself.

1.5 Vernam Cipher

Vernam cipher is a symmetrical stream cipher in which the plaintext is combined with a random or pseudorandom stream of data (the "keystream") of the same length, to generate the ciphertext, using the Boolean "exclusive or" (XOR) function.

$$\text{plaintext} \oplus \text{key} = \text{ciphertext}$$
$$\text{ciphertext} \oplus \text{key} = \text{plaintext}$$

If the keystream is truly random and used only once, this is effectively a one-time pad. Then mathematically speaking it would take forever to crack this code. If, however, the same keystream is used for two messages,the effect of the keystream can be eliminated

$$\text{ciphertext1} \oplus \text{ciphertext2} = \text{plaintext1} \oplus \text{plaintext2}$$

RC4 is an example of a Vernam cipher that is widely used on the Internet.

1.5.1 One time pad

One-time pad (OTP) is an encryption technique that cannot be cracked, but requires the use of a one-time pre-shared key the same size as, or longer than, the message being sent. In this technique, a plaintext is paired with a random secret key. Then, each bit or character of the plaintext is encrypted by combining it with the corresponding bit or character from the pad using modular addition.

The resulting ciphertext cannot be decrypt if the following conditions are met

1. The key must be truly random
2. The key must be at least as long as the plaintext.
3. The key must never be reused in whole or in part
4. The key must be kept completely secret.

It has also been proven that any cipher with the property of perfect secrecy must use keys with effectively the same requirements as OTP keys.

NOTE: A vigenere cipher with length of the key equal to length of the message is an example for one-time pad.

1.6 Brute Force Attack

One of the simplest possible technique to break a code is to try all possible keys, but this would also take a lot of time. The computational effort required to carry out a brute force attack is of the order of the keyspace **O(keyspace)**.

Size of the keyspace serves as the upper limit on the strength of the cipher.

Spurious messages: There may be additional keys that produce meaningful but incorrect plain text messages these are known as spurious messages.

Unicity distance: The more cipher text we have lesser is the chance of ending up with spurious messages, the minimum amount of cipher text required to reduce the number of spurious messages to zero is called Unicity distance.

Strength: The number of keys to be tried in order to break the code

NOTE: A cipher is computationally secure if the best attack to crack the code is not much better than brute force.

1.7 Key distribution problem

If a secret agency has **N** agents then it is best practice to have pairwise keys. i.e. every agent uses a different keys to send message to other agents
$\implies$ every agent should have **N-1** keys. This way even if one key is compromised it wont affect the agency much and it is easy to identify which key was compromised. Hence pairwise keys provide greatest security.

But this wont scale when the value of **N** is large. So an alternative is to have a common key for a large group of users. This will reduce the number of keys required, but however larger the group more valuable is the key (adversary are willing to spend more resources to get hold of that key) and a compromise of that key will be very difficult to handle for the agency. As it is not straightforward to identify the source of the key compromise.

2 Decryption techniques for Classical ciphers

2.1 Decrypting Alberti cipher

Requires the Alberti cipher disk, initial position, period and shift

2.2 Decrypting Trimethius cipher

To decrypt the text, successively shift the letters in the direction opposite to that used while encrypting it.

2.3 Decrypting the Vigenere cipher

Assume the plaintext is represented as **P**, the ciphertext as **C**, the key as **K**, and the total number of letters in plaintext as **N**

ENCRYPTION

$$\mathbf{C} = (\mathbf{P} + \mathbf{K}) \bmod \mathbf{N}$$

DECRYPTION

$$\mathbf{P} = (\mathbf{C} - \mathbf{K}) \bmod \mathbf{N}$$

Ex:
plaintext: helloworld key: code ciphertext: jsopqkrvnr

j	s	o	p	q	k	r	v	n	r
c	o	d	e	c	o	d	e	c	o
(9-2) % 26	(18 - 14) %26	(14 - 3) %26	(15 - 4) %26	(16 - 2) %26	(10 - 14) %26	(17 - 3) %26	(21 - 4) %26	(13 - 2) %26	(17 - 14) %26
7	4	11	11	14	22	14	17	11	3
h	e	l	l	o	w	o	r	l	d

Figure 9: Decrypting Vigenere cipher

NOTE:

- If the key used in the vigenere cipher contains all the 26 alphabets occuring only once, then the cipher text letters on average occur with uniform frequency.
- If key is of same length as plaintext, and is chosen randomly and never reused, then Vigenere cipher is a one-time pad

3 Types of Cryptanalytic Attacks

The main aim of cryptanalysis is to compromise one or more of

- Confidentiality
- Integrity
- Authentication

The mere information about with whom and how frequently a secret agency is communicating sheds a lot of light on their plans.

3.1 Traffic Analysis

Traffic analysis is the process of intercepting and examining messages in order to deduce information from patterns in communication, which can be performed even when the messages are encrypted.[1] In general, the greater the number of messages observed, or even intercepted and stored, the more can be inferred from the traffic.

Traffic analysis method can be used to break the anonymity of anonymous networks. There are two methods of traffic-analysis attack

1. **Passive traffic-analysis:** the attacker extracts features from the traffic of a specific flow on one side of the network and looks for those features on the other side of the network.
2. **Active traffic-analysis:** the attacker alters the timings of the packets of a flow according to a specific pattern and looks for that pattern on the other side of the network; therefore, the attacker can link the flows in one side to the other side of the network and break the anonymity of it.

3.2 Mathematical Attacks

- When the attacker has access only to cipher text, then they must perform **ciphertext-only attack**
  Ex: Brute force attack
- When the attacker has access to cipher text and some plain text, then they perform **Known-plaintext attack**
  Ex: Enigma Intercept

  With classical encrypting techniques, by knowing any two of the three (plaintext, ciphertext, key) the third one could be easily obtained
- When the attacker can trick the sender to send some message of his choosing and later intercept the cipher text corresponding to that message, it is called **Chosen-plaintext attack**

- Chosen-plaintext attack is very lethal by itself, but an even more powerful version of Chosen-plaintext attack exists and this is called **Adaptive-plaintext attack**. Once you trick the sender to encrypt a message of your choice, further tricking him to send additional messages to leverage the information already disclosed from your previous messages.

  NOTE: Chosen-plaintext attack can be carried out easily, but it is difficult to carry out Adaptive-plaintext attack.

- If you can trick your enemy into decrypting encrypted messages of your choice it is called **Chosen/Adaptive ciphertext attack**

- **Related-key attack** is any form of cryptanalysis where the attacker can observe the operation of a cipher under several different keys whose values are initially unknown, but where some mathematical relationship connecting the keys is known to the attacker.

3.3 Side Channel Attacks

Side-channel attacks do not target the weaknesses in the algorithm but target weaknesses in the computer system that implement the algorithm. Timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited.

3.4 Social Engineering Attacks

Exploiting human weaknesses like laziness, fear to get knowledge of the key.

- **Rubber hose cryptanalysis:** Using threats or violence to get the key from a person

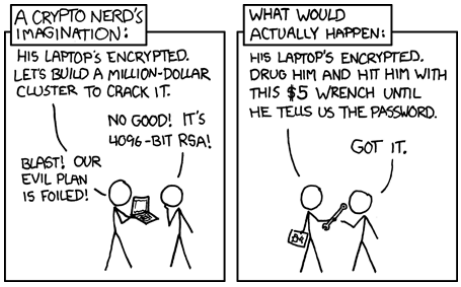


Figure 10: Rubber hose attack

- Careless user behaviour
- Weak/default passwords
- Naive trust

4 Frequency Analysis of Monoalphabetic ciphers

- Non-randomness in the plain-text conveys meaning
- Non-randomness in the cipher-text gives hints

With monoalphabetic ciphers all that we are doing is replace one letter with another, and hence the non randomness required for the plain-text continues to exist even after encrypting it.

Letter	Frequency	Letter	Frequency
e	12.7020%	m	2.4060%
t	9.0560%	w	2.3600%
a	8.1670%	f	2.2280%
o	7.5070%	g	2.0150%
i	6.9660%	y	1.9740%
n	6.7490%	p	1.9290%
s	6.3270%	b	1.4920%
h	6.0940%	v	0.9780%
r	5.9870%	k	0.7720%
d	4.2530%	j	0.1530%
l	4.0250%	x	0.1500%
c	2.7820%	q	0.0950%
u	2.7580%	z	0.0740%

Figure 11: Relative frequencies of English alphabets c

In a long piece of text, if the letters are used uniformly then they will have a probability of occurence close to 4%, but from the above table it can be seen that the occurence of letter **e** is nearly thrice that of the average.

So when presented with a monoalphabetic cipher text we can safely assume the most frequently repeated alphabet to be e, if that fails then we can assume it to be t and so on. This technique is very effective to crack monoalphabetic ciphers, and is much more effective than brute force.

NOTE: The letter frequencies may change based on the topic, person writing it, language and many other factors.

- Usually it is preferable to have short cipher texts, because with that the frequency of letter distribution significantly deviates from that in the table.
- In most cases less frequently used alphabets are used for encrypting punctuation marks, which again changes the frequency of occurence of letters.
- Frequency analysis is not like a magic bullet, that will crack any monoalphabetic cipher. Always use it with a grain of salt.

Recognising patterns is the key to deciphering the cipher text.

4.1 Multicharacter frequency analysis

Often using monoalphabetic frequency analysis alone is not sufficent to break the entire cipher, hence it is essential to do multicharacter frequency analysis. Searching for frequently appearing combination of letters.

- There are $26^2 = 676$ possible digraphs (2 letter sequences)
- There are $26^3 = 17{,}576$ possible trigraphs (3 letter sequences)
- There are $26^4 = 456{,}976$ possible trigraphs (4 letter sequences)

Frequently occuring

- Digraph: **th, he, in, er, an, re**
- Trigraph: **and, ing, ent, ion**
- Quadgraph: **tion**

Keypoints to identify the letters

- Q is almost always followed by U
- J, Z are almost always followed by a vowel
- If the above occurences are not observed then we can suspect that it is the space between two words.
- More than half of words end in E, T, D, S
- Plaintext qq, jj do not occur

Often plotting the frequency of single letter against the frequency repeated double letter digraph, is very helpful in cracking the cipher. But note that it is still a time consuming process and does not crack the cipher instantly. But as and when we get more cipher text, the mappings in the this graph become more clear and obvious.

5 Cracking Polyalphabetic Ciphers

Since cracking monoalphabetic ciphers was compartively easy, there were several attempts made to increase the strength of the cipher like

- Choosing a cipher text set with more characters than that in plaintext set, and choosing multiple characters to map the frequently occuring characters and lesser number of characters to map less frequently occuring characters. These are called **homophonic ciphers**
- Polyalphabetic ciphers

Cracking polyalphabetic ciphers is not an easy task.!!

5.1 Cracking Caesar shift cipher

Frequency analysis is the go to tool for cracking the Caesar shift cipher. The first step would be to calculate the frequency distribution of the ciphertext characters, and then compare it with the frequency distribution of English alphabets, and by shifting the two frequency distributions relative to one another we could find the shift that was used to encipher the plain text. The **Chi-squared statistic** is a way for a computer to essentially perform this procedure.

5.1.1 Chi Squared statistic

The Chi-squared Statistic is a measure of how similar two categorical probability distributions are. If the two distributions are identical, the chi-squared statistic is 0, if the distributions are very different, some high number will result.

$$\chi^2(\mathbf{C},\, \mathbf{E}) = \sum_{i=A}^{i=Z} \frac{C_i - {E_i}^2}{E_i}$$

where C_i is the count of the ith in the cipher text and E_i is the expected count of the ith letter.

5.2 Cracking Vigenere Cipher

- A keypoint to notice with the Vigenere cipher is that, the number of different Caesar ciphers used is equal to the length of the key.

If the length of the key used for encryption is known then this is sufficient to crack the vigenere cipher. Both **Babbage** and **Kasiski** proposed methods that could crack the Vigenere cipher.

5.2.1 Coincidence analysis

Index of Coincidence is a measure of how frequently you expect to find coincidental matches between two unrelated strings of text written in the same language.

Text1: ITWASADARKANDSTORMYNIGHT
Text2: WHILEEVERYONEATCOURTWASB

In the above two strings we can see that coincidence occurs at 3 places and the length of the string is 24

$$\text{Rate of coincidence} = \frac{3}{24} = 12.5\%$$

NOTE: Index of Coincidence helps identify the key length in polyalphabetic ciphers.

Let there be two languages, where the frequencies of the i^{th} letter are given by f_i and g_i respectively. If there are N common letters in these two languages, the rate of coincidence in any two strings created using these languages will be

$$r = \sum_{i=1}^{N} f_i * g_i$$

In cryptography most of the time the plain text and cipher text will be of the same language, hence we get

$$r = \sum_{i=1}^{N} f_i^2$$

For English this rate is **6.65%** If all the alphabets have the same frequency i.e. Uniform distribution of the frequencies serves as the worst case to calculate the coincidence rate. Having uniform frequency distribution simply means that alphabets are chosen randomly and hence this is called **random coincidence rate**

For English alphabets the random coincidence rate is

$$r_{rand} = \frac{1}{26} = 3.85\%$$

To compare coincidence rates for languages with different frequency distribution and different character sizes it is always better to first normalize the coincidence rate by dividing it with the random coincidence rate

- Index of Coincidence is the coincidence rate normalized to the random rate

Let the length of the text be **N** and let the size of the alphabet be **n**. Consider the ith letter a_i in the alphabet. Suppose a_i appears in the given text F_i times. Since the number of a_i's in the text is F_i, picking the first a_i has F_i different choices and picking the second a_i has only F_i-1 different choices because one a_i has been selected. Since there are N(N-1) different ways of picking two characters from the text, the probability of having two a_i's is

$$IC = \frac{1}{N(N-1)} \sum_{i=1}^{N} F_i(F_i - 1)$$

For English this is $\frac{6.65\%}{3.85\%} = 1.73$

5.2.2 Babbage's vital breakthrough

Consider the plain text **The sun and the man in the moon** and a key **king**. Since the length of the key is 4, there are possible Caesar ciphers that can be used.
Considering the Caesar cipher with shifts of 10, 8, 13, 6 the possible cipher texts for the word **the** are *dpr*, *buk*, *gno* and *zrm*. Now lets look at the cipher text for the above plain text keyword combination:

ciphertext: dpr yev ntn buk wia ox buk wwbt

Owing to the nature of the polyalphabetic cipher each letter is encoded with a different letter on each occurence. But an important thing to notice here is, the pattern **buk** is repeated twice and surprisingly this repetition coincides with the repetition occurred in the plain text. Charles Babbage found out that

Repetitions in the cipher text indicated repetitions in the plain text and the space between such repetitions hinted at the length of the keyword

5.2.3 Procedure to crack the Vigenere Cipher

- Make a guess about the maximum length of the key
- Generate subsequences and obtain the Index of Coincidence
  - Start with a guess that key length is 2
  - Create 2 substrings s0 and s1
  - Split the cipher text into separate sub strings based on the index of characters
  - Add the character to s0 if the index%2 = 0 else add it to the substring s1
  - Obtain the index of coincidence for each of these substrings
  - Calculate the average index of Coincidence

Repeat the above procedure for all possible values of the length key upto the guess made in the previous step.

  - For key length 3 the three substrings will have indices
    s0 = [0,3,6,9,12,15,.....]
    s1 = [1,4,7,10,13,.....]
    s2 = [2,5,8,11,14,.....]

  And save all the average index of coincidence obtained for each key length considered

- Possible length of the key corresponds to the one with maximum value of Index of Coincidence
  - If this does not work, consider the factors of the obtained value
- Now that we have obtained the key length (**K**), we will next have to crack these many Caesar shift ciphers
- Split the cipher text into **K** substrings based on index as done in step 2
- Chi squared statistic can be used for this purpose
- Once the key is obtained, it is straight forward to obtain the original plain text using modular arithmetic.

6 Modern Ciphers

With the advancement in technology more complex means where used to encrypt data, this paved way for modern ciphers. With classical cryptography ciphers they deal with the characters but the modern ciphers deal with the binary bit sequences.

Classic Cryptography	Modern Cryptography
It manipulates traditional characters, i.e., letters and digits directly.	It operates on binary bit sequences.
It is mainly based on 'security through obscurity'. The techniques employed for coding were kept secret and only the parties involved in communication knew about them.	It relies on publicly known mathematical algorithms for coding the information. Secrecy is obtained through a secrete key which is used as the seed for the algorithms. The computational difficulty of algorithms, absence of secret key, etc., make it impossible for an attacker to obtain the original information even if he knows the algorithm used for coding.
It requires the entire cryptosystem for communicating confidentially.	Modern cryptography requires parties interested in secure communication to possess the secret key only.

Figure 12: Difference between classical and modern cryptography

6.1 Block cipher vs Stream cipher

Block ciphers work on one block of data at a time, the size of the block is fixed and all the bits of the block should be present before the encryption of the block takes place.
Streamm ciphers work on a bit/byte at a time, hence it processes the data as a stream.

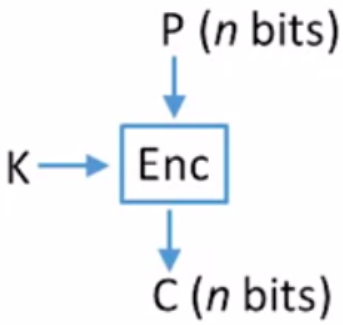


Figure 13: Block diagram for Block cipher

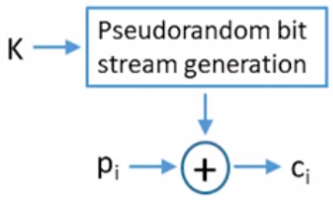


Figure 14: Block diagram for stream cipher

In case of block cipher if the length of the plain text is not divisible by the size of the block,then bits are padded at the end to make the length of the plaintext divisible by the size of the block.

Block ciphers generally have more applications as compared to stream ciphers.

6.2 Ideal Block Cipher

The requirements of a block cipher are:

- Fixed size blocks of n bits $\implies$ 2^n possible block options $\implies$ $2^n!$ possible mappings
- It should be a reversible function

$$\text{Dec(K, Enc(K,X))} = \text{X}$$

- Given the key K, the computation of the function is deterministic and easy

6.3 Feistel Cipher

The use of ideal block cipher is limited because the amount memory required increases exponentially with the key length. There was a need for block cipher designs that could ensure decent amount of security even with smaller key length. One such design was suggested by an IBM researcher **Horst Feistel**.

Feistel wanted an approximation of ideal block cipher, built out of components that are easily realizable. He suggested that the ideal block cipher can be approximated using the concept of product cipher. As the resulting encryption provides more security than the individual components themselves. In particular Feistel proposed the use of a cipher that alternates substitutions and permutations. This structure is called **Feistel cipher/ Feistel network**

Feistel cipher uses a key of k bits where

$$\text{k} < \text{n x } 2^n \text{ bits}$$
$$2^k < 2^n! \text{ mappings}$$

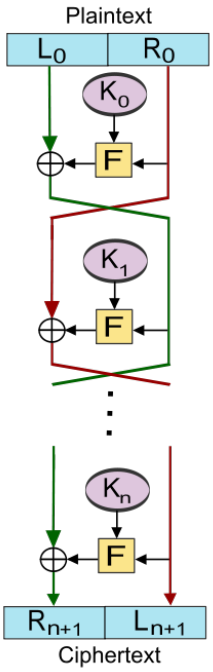


Figure 15: Encryption process in Feistal cipher

As it can be seen from the figure the plaintext is split into two halves. The right half is passed through a function and then XORed with the left half (this is the substitution part). After XORing the two halves they are swapped at each stage (this is the permutation part). In the i^{th} round

$$\mathrm{L}_i = \mathrm{R}_{i-1}$$
$$\mathrm{R}_i = \mathrm{L}_{i-1} \oplus \mathrm{F}(\mathrm{R}_{i-1}, \mathrm{K}_i)$$

The main advantage with Feistal cipher is that, the function F need not be reversible, as the same function can be used to decrypt the ciphertext. The same hardware can be used for encryption and decryption.

The parameters of Feistal cipher design are

- Block size
- Key size
- Number of rounds (n)
- Subkey generation (K_i)
- Round Function (F)

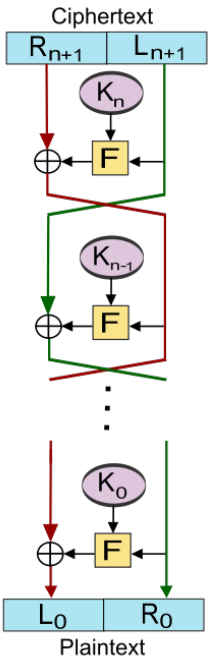


Figure 16: Decrypting process for Feistal cipher

6.4 Data Encryption Standard (DES)

This is the most widely used block cipher, which is based on Feistel cipher. This was developed by **IBM** and **NSA**. The parameters of DES are:

- Block size: 64 bits
- Key size: 56 bits
- Number of rounds: 16 rounds
- Subkey generation:
- Round Function:

DES uses the same structure as Fesital cipher, but adds additional permutation blocks at the beginning and the end.
Subkey generation process takes a 56 bits key and outputs 16 keys each of size 48 bits.

6.4.1 DES Round function

Round function takes two inputs of size 32bits (R_i) and 48bits(K_i). The 32-bit input first goes through an expansion block, where it undergoes reordering

and is expanded to 48 bits. This output is then XORed with K_i and passed through 8 blocks (6 bits input to every block). The data undergoes substitution (mapping through a look-up table) and compression in these blocks and the output will be of 4bits from each S-block. These 32 bits go through another permutation, the output of the permutation block is the output of the round function.

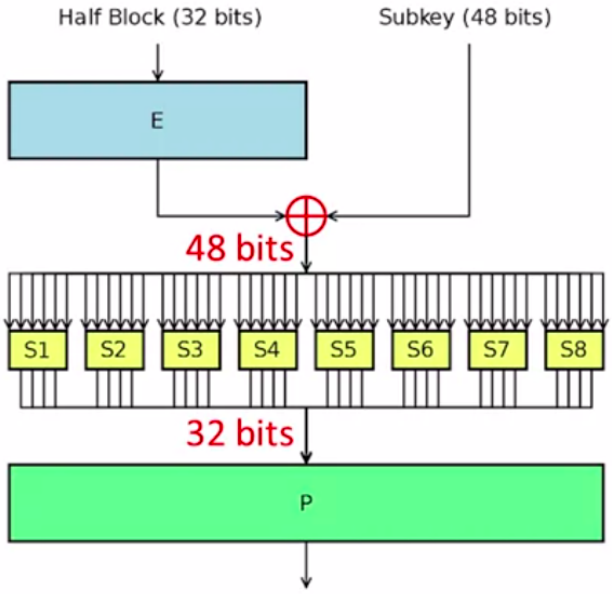


Figure 17: DES round function

6.4.2 DES Subkey generation

Permutation and compression (PC) blocks are used to generate 16 subkeys of size 48bits from a single key of size 64bits. To ensure that every PC block gets a different input, the data undergoes left-circular shifts in between two PC blocks.

6.4.3 DES security

The design of the encryption ensures that even if one plaintext bit changes it results in the changing of nearly half of the cipher text bits. This helps in security against differential analysis[5].

As DES uses 56 bit key, if an attacker uses Brute force then the complexity involved is $O(2^{55})$. DES is broken now i.e. it is easy to crack the code eve using brute force attack, and hence is not recommended for securing systems and applications.

[5]Attacker analyses how difference in plaintext affects the difference in cipher text

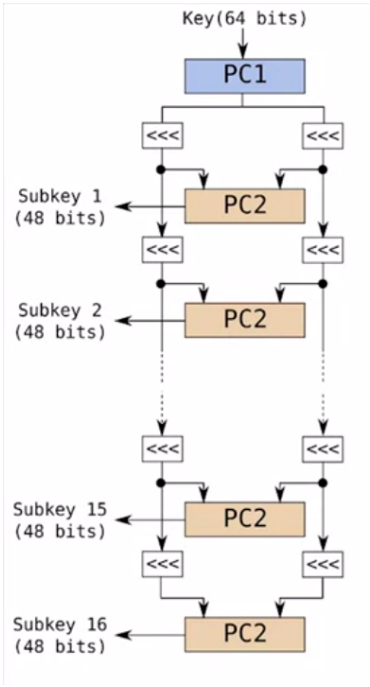


Figure 18: DES subkey generation

6.5 Double DES

As DES is no longer secure, an alternative which provides more security against brute force attack is necessary. One such alternative is **Double DES** which uses multiple encryptions with DES with multiple keys and the same hardware as DES.

Double-DES uses two encryption stages and two keys K_1, K_2.

$$\text{ciphertext} = \text{Enc}(K_2, \text{Enc}(K_1, \text{plaintext}))$$
$$\text{plaintext} = \text{Dec}(K_1, \text{Dec}(K_2, \text{ciphertext}))$$

A brute force attack on this encryption scheme involves a complexity of $O(2^{111})$

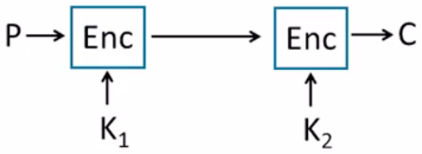


Figure 19: Double DES

In the above diagram the block enc implements DES.

6.5.1 Meet in the Middle Attack

Double DES encryption is vulnerable to meet in the middle attack. In general any block encryption cipher which is sequentially processed is vulnerable to **Meet in the Middle attack**.

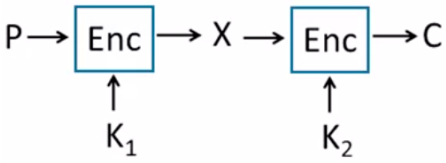


Figure 20: Meet in the middle attack

This is similar to known plain text attack, where the attacker has knowledge about a mapping $\mathbf{P} \rightarrow \mathbf{C}$.
This attack is carried out as follows

- Compute and store all 2^{56} mappings of a known plain text **P** using different K_1's. Call this X.
- Compute and store all possible (2^{56}) decryptions for **C** using different K_2's . Call this X
- Compare the 2^{56} X's obtained in each of the previous steps. If a match is obtained, then the corresponding K_1 and K_2 could be the possible keys.
- Verify the candidate K_1 and K_2 using a different plaintext and ciphertext mapping.

Hence using this type of an attack reduces the complexity from $O(2^{111})$ to $O(2^{57})$. Hence The complexity is just twice that involved for a brute force attack with DES.

6.6 Triple DES

Since double DES is not secure enough, 3 DES ciphers are applied sequentially.

$$\text{ciphertext} = \text{Enc}(K_3, \text{Dec}(K_2, \text{Enc}(K_1, \text{plaintext})))$$
$$\text{plaintext} = \text{Dec}(K_1, \text{Enc}(K_2, \text{Dec}(K_3, \text{ciphertext})))$$

The choice of keys for triple DES

1. $K_1 = K_2 = K_3$
   Triple DES becomes equivalent to DES
2. K_1, K_2 independent; $K_3 = K_1$
3. K_1, K_2, K_3 are independent

Options 2,3 result in a complexity of $O(2^{112})$ even with a meet-in-the-middle-attack. Option 2 is called **2-Key Triple DES**. Option 3 is called **3-Key Triple DES**. Both the options provide equal security.

6.7 Advanced Encryption Standard (AES)

Though triple DES provides greater security than DES, it takes thrice the amount of computation as required by DES. Hence it is slower. Hence another algorithm was deviced by Vincent Rijmen and Joan Daemen. Which provides the same level of security as triple DES but also needs lesser computation time.

AES processes the data on bytes. The block length is 16 bytes (4 columns and 4 rows) and is called **state array**. Each row/column in a state array is called a **word**.

AES supports 128/192/256 bit keys. AES is not based on Feistal Cipher structure but still uses combination of substitution and permutation. The key size used for an AES cipher specifies the number of transformation rounds

- 10 rounds for 128-bit key
- 12 rounds for 192-bit key
- 14 rounds for 256-bit key

Each round involves multiple steps as follows

1. SubBytes: Substitution using look-up table
2. ShiftRows: Row-based transposition
3. MixColumns: Column-based mapping
4. AddRoundKey: XOR with 16 byte round key

Before the above mentioned rounds, the data is XORed with a 16 byte round key.

NOTE: In AES finite field based arithmetic is used for MixColumns and Key-Expansion (used for round key generation)

The decryption process reverses the encryption process one step at a time, so the last step performed during encryption is the first to be reversed during decryption. The various steps performed during decryption are

1. AddRoundKey (Inverse of XOR is XOR)
2. InverseMixColumns
3. InverseShiftRows
4. InverseSubBytes

AES decryption implementation is not same as AES encryption implementation. Hence same hardware/software cannot be used for both.

7 Block Cipher Operation modes

Block cipher operation modes give different ways to handle plain text that is longer than a block in a secure manner.
Commonly used variables

- b: block length
- P_i: ith plaintext block (b bits)
- C_i: ith ciphertext block (b bits)
- K: Key

7.1 Electronic Code Book (ECB)

This is the simplest mode of operation of a block cipher. In this mode encryption of each block of input plaintext is done independently and output is in form of blocks of encrypted ciphertext. Generally, if a message is larger than b bits in size, it can be broken down into bunch of blocks and the procedure is repeated.

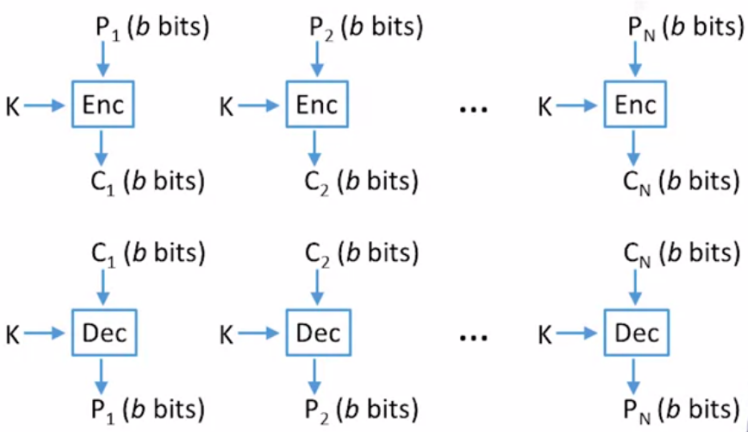


Figure 21: Electronic Code book encryption and decryption

7.1.1 Advantages of ECB

- Parallel encryption of blocks of bits is possible, thus it is a faster way of encryption.
- It is the simplest operating mode

7.1.2 Disadvantages of ECB

Since the ciphertext is directly related to that particular block of plaintext, it is vulnerable to many different types of attacks.

Hence ECB is usually preferred for encrypting when the amount of data to be transferred is less.

7.2 Cipher Block Chaining (CBC)

Each previous cipher text block is chained with the current plaintext block.

$$\text{Encryption: } C_i = \text{Enc}(K, [C_{i-1} \oplus P_i])$$
$$\text{Decryption: } P_i = \text{Dec}(K, C_i) \oplus C_{i-1}$$

Is used when large amounts of data is to be transferred securely.

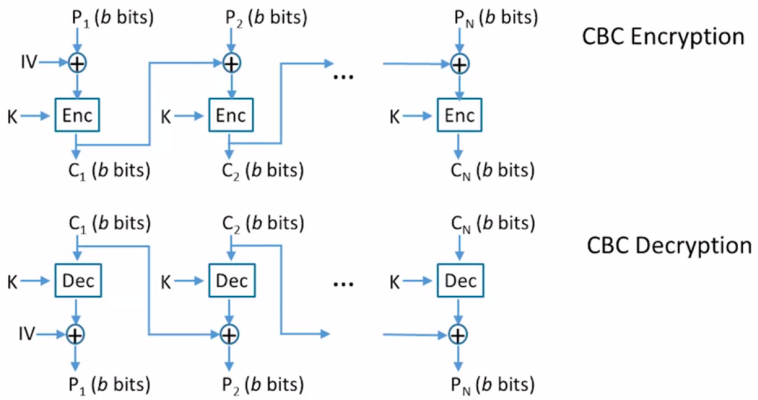


Figure 22: Cipher Block Chaining encryption and decryption

IV is the initialization value used to chain the input and this must be protected from attackers.
Using this technique the repeating patterns in the plaintext are not exposed in the cipher text.

It is possible to convert block cipher to stream cipher, the stream modes of operation for block cipher are:

- Cipher Feedback (CFB) mode
- Output Feedback (OFB) mode
- Counter (CTR) mode

7.3 Cipher Feedback (CFB)

Encryption: $C_i = P_i \oplus \text{Enc}(K, C_{i-1})$
Decryption: $P_i = C_i \oplus \text{Enc}(K, C_{i-1})$

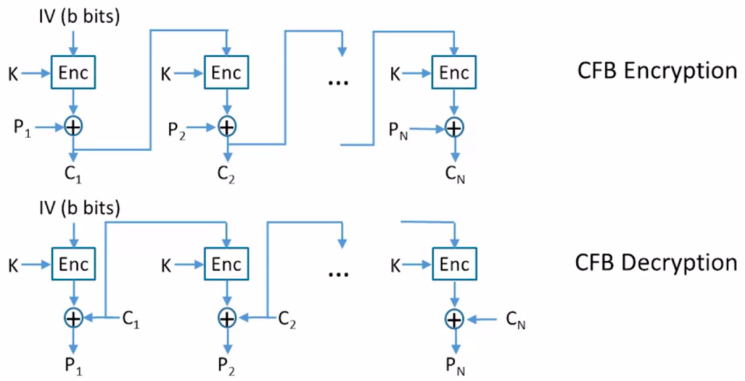


Figure 23: Cipher Feedback encryption and decryption

Note that encryption function does not take plaintext as input, it is only used to generate a pseudo random number which is then XORed with plain text.

An error in decoding at any particular stage will result in an avalanche and all further decoding stages will result in an erroneous output.

7.4 Output Feedback (OFB)

Encryption: $C_i = P_i \oplus \text{Enc}(K, [C_{i-1} \oplus P_{i-1}])$
Decryption: $P_i = C_i \oplus \text{Enc}(K, [C_{i-1} \oplus P_{i-1}])$

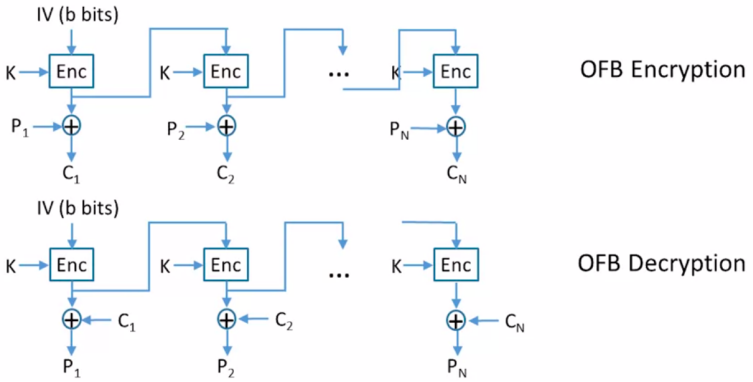


Figure 24: Output Feedback encryption and decryption

Error in cipher text does not propagate. This mode is vulnerable to attacker's message stream modification.

7.5 Counter (CTR)

There is no explicit chaining/feedback. The random number generation does not depend on any input from previous stages.

$$\text{Encryption: } C_i = P_i \oplus \text{Enc}(K, T_i)$$
$$\text{Decryption: } P_i = C_i \oplus \text{Enc}(K, T_i)$$

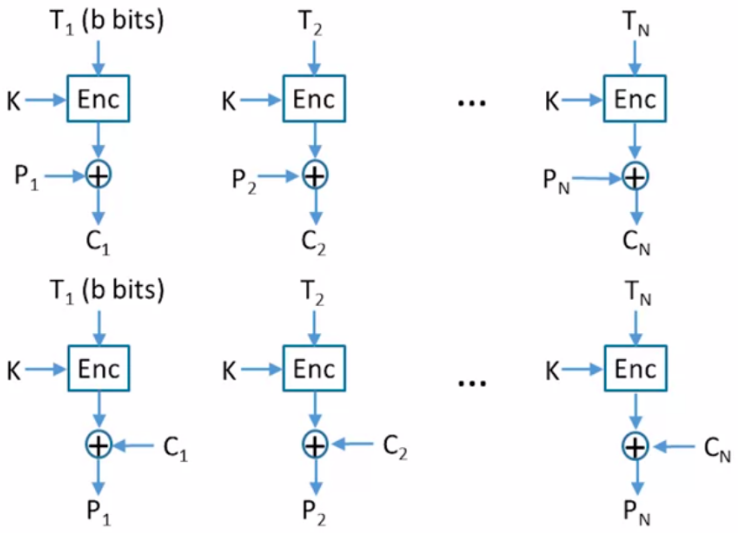


Figure 25: Counter mode encryption and decryption

A counter value is used at each stage to generate the pseudo random number. As there is no chaining it supports parallel processing

8 Hash functions

It is a mathematical algorithm that maps data of arbitrary size (often called the "message") to a bit array of a fixed size (the "hash value", "hash", or "message digest") and is a one-way function, that is, a function which is practically infeasible to invert. Ideally, the only way to find a message that produces a given hash is to attempt a brute-force search of possible inputs to see if they produce a match.

Cryptographic hash functions are a basic tool of modern cryptography

Hash function is a **many-to-one mapping**, this makes inverting the hash function to obtain the message from the hash infeasible.

- When two messages produce the same hash/digest it is called **Hash collision**. Cryptographic hash function needs to be collision-resistant.

The Random Oracle is as secure as a Hash function can get

The Random Oracle provides an useful model for the cryptographic hash functions.

8.1 Hash functions to achieve data integrity and message authentication

Size of the hash function is the number of bits in the digest it produces. If the hash function has N-bits then it can produce 2^N unique digests.

Some interesting numbers

- Atoms in Universe $\tilde{2}^{265}$
- 256-bit hash requires half the mass of the universe at 1atom/bit

These numbers show that to use a brute force attack on such a system would nearly take around half a million years.

NOTE: Real hash functions introduce significant vulnerabilities

Consider an example where Alice uploads some documents to the drive and Bob will have to download these files from the drive. In order to make sure that the document Bob is downloading is exactly similar to the one Alice uploaded and there hasnt been any incident where another party has modified the document.

Alice shares the message digest of the contents of the documents with Bob. Once Bob downloads the documents, he can again hash them and then compare the message digest that he is getting with the one Alice sent to him. If the document was not modified, then both the message digest will be identical, if the document was modified by Malory on the drive, then message digest that Bob gets after hashing the contents of the document will not match with the one Alice sent him. **Hence cryptographic hash functions can be used to ensure message integrity.**

8.1.1 Preimage attack

If all Eve wants to know the original message from the message digest she got access to, then this is called a **preimage attack**. Eve wants to find one out of the infinite number of messages that can correspond to the same digest. Though there are infinitely many candidates, pondering upon is very less likely, lets look at the math.

$$\begin{aligned}
\text{Consider a N bit hash function} \\
\text{Number of possible digests is: D} &= 2^N \\
\text{Chance of success per attempt: } &\frac{1}{D} \\
\text{Chance of failure after k attempts: } &= \left(1 - \frac{1}{D}\right)^k \\
\text{Number of attempts for 50/50 chance of success} &= \mathcal{O}(2^N)
\end{aligned}$$

Hence to even get a 50/50 chance of success the number of attempts required is extremly huge.
Preimage attack is computationally infeasible for 128-bit, 256-bit Cryptographic Hash functions Note that it is possible to construct even larger hash functions but which can be easily attacked. For example consider a hash function where the message itself is the hash. This is a very famous hash function used in programming languages, to store data.

To store integer values in memory (hash table), the hash used is the value of the integer itself.

This is an example for a **perfect hash** where hash collision is impossible. But at the same time it is trivial to break such a hash function. Because basically the hash function is just an Identity function.

$$\text{Knowing digest} \implies \text{knowing message}$$

8.1.2 Second preimage attack

Note that even if Malory gets to know the message digest, it is computationally infeasible for her to modify the document contents such that it would result in the same message digest as before. This type of attack where Malory tries

to produce a message digest similar to the one Alice used but for a different message is called **Second preimage attack**.
As Malory has to find a second message that exists in the same preimage set for a given message digest. (Remember Hash function is a many to one mapping)

With Random Oracle the second preimage attack reduces to pre image attack.

The **perfect hash** is immune to second preimage attack.

8.1.3 Man in the middle attack

If Malory also gets access to the message digest, then she can modify that as well to match the message digest that Bob would get when he hashes the document modified by Malory. This is called **Man in the middle attack**

8.1.4 Hash Collision Attack (Birthday Attack)

Finding two messages that give the same message digest is called **Hash collision attack**. This can be for any two messages unlike the preimage attack where we wanted to find the preimage for a particular message digest
It is easier to carry out a hash collision attack as compared to a pre image attack.

- The **Birthday paradox** makes this attack very strong.

The number of people we need in a room such that on a given date there are two birthdays.

The intuitive answer to this is 366.
As there are 365 days in a year, with 366 people in the room we are guaranteed to have two people such that their birthdays occur on the same date.

The first person can have his birthday on any of the 365 days

The second person can have his birthday on any of the remaining 364 days

Continuing the same way, we get 365-(k-1) possible days for the k^{th} person

The correspodning probability is:

$$p_{none}(k) = \frac{365}{365} * \frac{364}{365} * ... \frac{365 - (k-1)}{365}$$

The above value of p_{none} is around 0.52 when k = 22

So on an average to even have a 50/50 chance for having such a condition we need to have only around 22/23 people in the room.

Using this analogy we see that the number of attempts required to implement the **Hash collision attack** is very less.

With the hash functions, 365 is replaced with D as this is the number of available digests

$$p_{none}(k) = \prod_{i=0}^{k-1} \frac{D-i}{D}$$

$$p_{none}(k) = \prod_{i=0}^{k-1} (1 - \frac{i}{D})$$

For small values of x, the exponential function can be approximated using the Taylor series

$$e^x \approx 1 + x \text{ (for } \|x\| < 1)$$

$$p_{none}(k) \approx \prod_{i=0}^{k-1} e^{-\frac{i}{D}}$$

$$ln(p_{none}(k) = \sum_{i=0}^{k-1} (-\frac{i}{D}) = \frac{-k(k-1)}{2D}$$

$$ln(p_{none}(k) \approx -\frac{k^2}{2D}$$

$$k = \sqrt{2Dln(\frac{1}{p_{none}})}$$

$$\boxed{k = 1.177\sqrt{D} = 1.177.2^{N/2}}$$

Hence the number of attempts required to perform the attack is little more than the square root of number of digests. Hence the number of attempts required to carry out Hash collision attack is half of that required to carry out preimage attack.

8.2 Modification Detection Codes

When a hash function is used to combat malicious message modification, the digest is sometimes referred to as Modification Detection Codes (MDC). MDC by itself cannot prevent man in the middle attack, since Malory can provide the altered MDC for the modified message. So one way of tackling the man in the middleattack is

- Using an "Authentic channel", i.e. A trusted third party who can send Alice's message digest safely to Bob.
- Alice computes the digest for the message, and then creates a new message by appending the digest to the original message and then encrypts the result (appended message). Now Malory would have to figure out two things as even the Hash is encrypted. And this increases the computational security of the message

- **Sign** the digest by encrypting with private key. Share both the message and hash, once Bob decrypts the message using his private key and then hashes it, if it matches the hash received, then Bob can be sure that the message is authentic. Because the encryption is possible with only one key and that is Alice's private key.
- Use a Message Authentication Code (MAC)

Hashes can provide authentication when only the originator can provide the digest So to achieve this we would need the sender to have some extra information which is not available to the adversaries.

8.3 Keyed Hash functions

Designing hash functions that take not only the message as input but also a key along with it and then hash that combination. Since the key is available with only the sender, if the message Bob receives has the same message digest that the sender had sent, it not only reflects message integrity but also authenticates that Alice was the sender.

A very simple keyed hash function can be formed by appending the key to the original message and then hashing it, however these type of keyed hash functions are susceptible to **Length-extension attacks**

8.4 Properties of a cryptographic hash function

- it is deterministic, meaning that the same message always results in the same hash
- it is quick to compute the hash value for any given message
- it is infeasible to generate a message that yields a given hash value i.e. It should be a **one-way function** (not invertible)
  $\implies$ Preimage and second preimage resistant
- it is infeasible to find two different messages with the same hash value i.e. Collision resistant
- a small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value

Failing to achieve uncorrelated mapping from message to digest may result in **Gradient descent attacks**. Where we start from a random point and tweak a few bits to push the digest closer to the desired digest. This method is much faster than the brute force to perform any of the before mentioned attacks.

NOTE:

- Collision resistance $\implies$ second preimage resistance
- Collision resistance $\not\Rightarrow$ preimage resistance

9 Mathematical foundations for Cryptography

Modern Cryptography is all about understanding integer mathematics

Since most of Cryptography deals with exact calculations and there cannot be any approximations as such, only integers are used.

$$\begin{aligned}
\text{Integers: } & \mathcal{Z} = \{..., -3, -2, -1, 0, 1, 2, 3, ...\} \\
\text{Positive Integers: } & \mathcal{Z}^{+} = \{1, 2, 3, ...\} \\
\text{Negative Integers: } & \mathcal{Z}^{+} = \{..., -3, -2, -1\} \\
\text{Nonpositive Integers: } & \mathcal{Z}^{0-} = \{...., -3, -2, -1, 0\} \\
\text{Nonnegative Integers: } & \mathcal{Z}^{0+} = \{0, 1, 2, 3, ...\}
\end{aligned}$$

Def: m | a (m divides a) $\iff \exists_k$: a = km
NOTE:

- Divisibility is a statement, not an operation
- Sign has no effect
  6 | 42, 6 |-42, -6 | 42
- m | 0 is true. $\forall$ m (Possible by choosing k = 0)

Theorem 1 *If m | a and m | b, then m | (any result obtained by combination of a and b)*

Consider the famous water puzzle from DieHard 3.
Lets use a more general setting of the problem, where the capacity of both the jugs is **a** and **b**.

STATE MACHINE

States: pairs (x,y), where
x denotes the number of litres in **a** litre jug
y denotes the number of litres in **b** litre jug

Start state: (0,0)

Transitions:

- Emptying
  (x,y) → (0,y)
  (x,y) → (x,0)
- Filling
  (x,y) → (a,y)
  (x,y) → (x,b)
- Pouring
  (x,y) → (0, x+y) , $x+y \leq b$
  (x,y) → (x + y - b, b), $x + y \geq b$
  (x,y) → (x+y, 0), $x + y \leq a$
  (x,y) → (a, x + y - a), $x + y \geq a$

Solution to the above problem: (0,0) → (0,5) → (3,2) → (5,2) → (4,3)

Proof for the general case By Induction.
Assume m | a and m | b
Preposition:
P(n) = "If (x,y) is the state after n transitions then m | x, m | y"

Base case: (0,0) $\implies$ m | 0
Hence P(0) is true.

Inductive step: Assume (x,y) is the state after n transitions, m | x and m | y
After another transition the possible litres of water in jug is,
(0,0), (0,b), (a,0), (x+y, 0), (0, x+y), (a, x+y-a),(x+y-b,b)

We know that m | 0, m | a, m | b, m | x, m | y $\implies$ linear combinations of x, y, a, b are also divisible by m
Hence P(n+1) is true.

Ex: Is it possible to measure 4 litres from 33 litre and 55 litre jug??

A possible value for m such that m | 33 and m | 55 is m = 11.
So any combination of 33 and 55 must be divisible by 11.
Since 4 is not divisible by 11.
It is not possible to measure exactly 4 litres from 33 and 55 litre jug.

9.1 Prime numbers

Def: A prime number is any number greater than 1, whose only positive divisors are 1 and itself.

NOTE: Prime numbers cannot be negative.

Def: If a given positive number has more than two positive divisors, then it is called a Composite number.

NOTE: 0 and 1 are neither prime nor composite

9.2 Fundamental theorem of Arithmetic

Theorem 2 *Every integer greater than 1 either is a prime number itself or can be represented as the product of prime numbers and that, moreover, this representation is unique, except for the order of the factors.*

NOTE: This theorem is one of the main reasons why 1 is not considered a prime number: if 1 were prime, then factorization into primes would not be unique.
Ex: 2 = 2x1 = 2x1x1

9.3 Greatest Common Divisor (GCD)

The GCD (c) for a given pair of numbers (a,b) is the largest integer that divides both a and b. $\implies$ c | a and c | b

NOTE: Since 0 divides any other number, GCD is not defined for the case where **a** and **b** both are zero.

Def: If *gcd* of two numbers is equal to 1, then they are called **coprime**
i.e. a and b are relatively prime with respect to each other.

Corollary: *gcd*(a,b) | (any result obtained by combination of a and b)

9.4 Modular Arithmetic

This is a system of arithmetic where we work with the remainders after dividing by the modulus.

Any number **n** can be represented using three other numbers **q, d, r** by the following relationship

$$\mathbf{n = q.d + r}$$

q: quotient
r: remainder/residue
d: divisor
n: dividend

But with the above formulation the representation is not unique. So we can add some restrictions to make the representation unique.

- **r** should satisfy $0 \leq \mathbf{r} < \mathbf{d}$

9.4.1 Congruence

Two integers are said to be **congruent** modulo n, if n is a divisor of their difference, that is, if there is an integer k such that a - b = kn.

$$\text{-2} \equiv 11 \equiv 24 \text{ (mod 13)}$$
$$\text{24 - (-2)} = 26 = \text{13x2}$$

When dealing with very huge numbers in the modulo world, computations can be very time consuming.

Consider two numbers **a** and **b**

$$a = k_a N + r_a$$
$$b = k_b N + r_b$$
$$c = a + b$$
$$c = (k_a + k_b)N + (r_a + r_b)$$

Sum of two numbers is equal to sum of their residues

$$c = a.b = (k_a N + r_a).(k_b N + r_b)$$
$$c = (k_a k_b + r_a k_b + r_b k_a)N + r_a r_b$$

Product of two numbers is equal to product of their residues

NOTE: A final reduction is still needed at the end.

9.4.2 Division in Modular arithmetic

We know that 15 is congruent to 3 modulo 6. Consider the following

$$\frac{15}{3} \equiv \frac{3}{3} \text{ (mod 6)}$$

But this requires

$$5 \equiv 1 \text{ (mod 6)}$$

Which is not true. Hence division is not defined in modular arithmetic.

9.4.3 Exponentiation in Modular arithmetic

Exponentiation is defined in modular arithmetic, but it should be carried out carefully.
Consider

$$3^{14} \mod 7 = 4782969 \mod 7 = 2$$
$$3^{14 \bmod 7} = 3^0 = 1$$

The above is a frequently made mistake. But remember that
In a mod-7 world exponents live in mod-6 this is given by **Euler's Totient theorem**.

$$3^{14 \bmod 6} = 3^2 \mod 7 = 2$$

9.5 Origin of divisibility rule for 9

Consider a number $\mathbf{n} > 5$, sum up the digits in **n!** recursively until you get a one digit number.

Notice that this number obtained is always divisible by 9

$$9 \mid \mathbf{n!} \ (\mathrm{n} > 5)$$

If $9 \mid \mathrm{x} \implies \mathrm{x} \bmod 9 = 0$
Expanding x using the place value of each digit we get the following

$$\begin{aligned} x &= \sum_{i=0}^{N-1} d_i 10^i \bmod 9 \\ &= \sum_{i=0}^{N-1} d_i \bmod 9 \end{aligned}$$

this is because $10^i \bmod 9$ is always equal to 1.

From the above we can see that if 9 divides x, then 9 also divides the sum of digits in x.

Further this can be generalized to any base **B** system as well.

For a given number in base **B** system,
if the sum of the digits is divisible by **B-1**,
then the number is also divisible by **B-1**

9.6 Multiplicative Inverse

The analogous operation to division in a modular world is the multiplicative inverse.

In a **mod N** world 1 and N-1 are their own multiplicative inverses.

9.6.1 Existence of multiplicative inverse

Theorem 3 *gcd(a,b) is a linear combination of a and b*

Proof:

From Theorem 1 we know that if

$$m|a$$
$$m|b$$
$$m|(\text{linear combination of a and b})$$

GCD of **a** and **b** is one of the candidates for m.

$$\implies gcd \mid (\text{linear combination of a and b})$$
$$\implies gcd \mid (sa \ + \ tb)$$

for any two integers s,t

$$\implies gcd \text{ x } N = sa \ + \ tb$$
$$gcd = \frac{sa + tb}{N}$$
$$= \frac{s}{N}a + \frac{t}{N}b$$

Let $\frac{s}{N} = \text{p}$ and $\frac{t}{N} = q$

$$gcd = pa \ + \ qb$$

Hence we see that $gcd(a,\ b)$ is a linear combination of a and b.

Theorem 4 *In a **mod N** world for a number **a** to have a multiplicative inverse, it should be relatively prime to **N**.*

Proof:

$$gcd(a,\ N) = 1$$

$\implies$ there exists s and t such that

$$sa + tN = 1$$
$$\iff \ni_s \ N|\,(1 - sa)$$
$$\iff sa \equiv 1(\text{mod } N)$$

s is the required multiplicative inverse of a in modulo N.

9.7 Euclid's Algorithm to find GCD of two numbers

Since to find the multiplicative inverse, we should make sure that the number is relatively prime to the base. We need a method to compute the GCD of two numbers to check if they are coprime or not. Euclid proposed an algorithm that recursively reduces the numbers whose gcd is to be found, and will eventually give the GCD.

$$\text{gcd(a,b)} = \text{gcd(b, a\%b)} \ \text{(a>b)}$$

Proof:

Assume m | a and m | b, $\implies$ m | b-qa (From theorem 1)
We know that b - qa = rem(b,a)

$\therefore$ m | rem(b,a)
gcd is one of candidates for m, $\implies$ *gcd*(a,b) also divides rem(b,a)
$\implies$ *gcd*(a,b) $\leq$ *gcd*(rem(b,a),a)

- rem(b,a) $\neq 0$ then m | rem(b,a) and m | a $\implies$ m | b
  this is because b is obtained by linear combination of a, r
- rem(b,a) = 0 then m | a $\implies$ m | b
  this is because $b = aq$

From the above two cases it can be seen that any number that divides a and rem(b,a) also divides b.
$\implies$ *gcd*(rem(b,a),a) $\leq$ *gcd*(a,b)

Hence, from the above two inequalities obtained it is proven that

$$gcd(\text{rem(b,a)},\text{a}) = gcd(\text{a,b})$$

9.7.1 Efficiency of Euclid's algorithm

The number of iterations required to compute the GCD using Euclid's algorithm is **(Olog(b))**. In other words, the number of iterations is at most linear in the number of digits in the smaller input number.
Ex:

$$\begin{aligned} gcd(2017, 1024) &= gcd(1024, 993) \\ &= gcd(993, 31) \\ &= gcd(31, 1) \end{aligned}$$

Notice that the above numbers are in base10, and $\log_{10}(1024) = 3.01$
Hence after three iterations we get the GCD to be 1.

9.8 Extended Euclid's algorithm to find modular multiplicative inverse

Given two integers **x** and **y**, the extended Eucliean algorithm finds two integers **a** and **b** such that

$$\text{ax + by = gcd(x,y)}$$

The above equation is called **Bézout's identity**.
In case of coprimes (gcd(x,y) = 1) we obtain the following from the above equation we see that

$$ax \equiv 1(\text{ mod y }) \implies a \equiv x^{-1}(\text{ mod y })$$
$$by \equiv 1(\text{ mod x }) \implies b \equiv y^{-1}(\text{ mod y })$$
$$ax \equiv 1(\text{ mod b }) \implies a \equiv x^{-1}(\text{ mod b })$$
$$by \equiv 1(\text{ mod a }) \implies b \equiv y^{-1}(\text{ mod a })$$

- **Many solutions**
  If x = 1, y = 0 $\implies$ a = 1, b = *anything*
- **No solutions**
  If x = 2, y = 4
- **One solution**
  If x = 7, y = 5
  3(7) + (-4)5 = 1
  $\implies$ 7 and 8 are multiplicative inverses in **mod 5** and **mod 11** world.

Hence we get a lot of information from the extended Euclid Algorithm. Which is a overkill if all we wanted to find was the multiplicative inverse for a given number in a particular modulo world.

9.8.1 Recurrence relation of Euclidean algorithm

$$\text{gcd(m, x) = gcd(m, x mod m)}$$

Using an array to store all the intermediate values we get

$$\text{gcd(r[0], r[1]) = gcd(r[1],r[2])}$$

where r[0] = m, r[1] = x

$$\text{r[2] = r[0] mod r[1]}$$

In general

$$\text{r[i] = r[i-2] mod r[i-1]}$$

Repeatedly solving this recurrence relation if we get

- r[i] = 1 $\implies$ inverse exists
- r[i] = 0 $\implies$ no inverse

9.8.2 Recurrence relation to find multiplicative inverse

If the inverse exists (i.e. the number and base are co primes) we get

$$ax \equiv 1(\text{mod m})$$

We know that 1 is the final value of r[i]. We can define an auxiliary congruence relation as follows

$$\text{a[i]x} \equiv \text{r[i]} \text{ (mod m)}$$

Rewriting the recurrence relation obtained previously for r[i] in the integer world we get

$$\text{r[i]} = \text{r[i-2]} \text{ mod r[i-1]}$$
$$\text{r[i-2]} = \text{q[n]r[i-1]} + \text{r[i]}$$

where

$$\text{q[n]} = \left\lfloor \frac{r[i-2]}{r[i-1]} \right\rfloor$$

$$\text{r[i]} \equiv \text{r[i-2]} - \text{q[n]r[i-1]} \text{ (mod m)}$$
$$\text{a[i]x} \equiv \text{a[i-2]x} - \text{q[n]a[i-1]x} \text{ (modm)}$$

Assuming multiplicative inverse exists for x we get

$$\text{a[i]} \equiv \text{a[i-2]} - \text{q[n]a[i-1]} \text{ (modm)}$$

If our assumption is wrong, then we will end up with some garbage and since our assumption is wrong, it means there is no multiplicative inverse for x. The sequence of remainders will reveal if our assumption was correct or wrong.

$$\text{a[0]x} \equiv \text{r[0]} \equiv \text{m (mod m)} \implies \text{a[0]} = 0$$
$$\text{a[1]x} \equiv \text{r[1]} \equiv \text{x (mod m)} \implies \text{a[1]} = 1$$

Ex: Lets find multiplicative inverse of 7 in mod 11 world
i = 0

$$\text{r[0]} = 11$$
$$\text{a[0]} = 0$$
$$\text{q[0]} = -$$

i = 1

$$\text{r[1]} = 7$$
$$\text{a[1]} = 1$$
$$\text{q[1]} = -$$

i = 2

$$\begin{gathered} r[2] = 11 \ \% \ 7 = 4 \\ q[2] = \lfloor 11/7 \rfloor = 1 \\ a[2] = 0 - 1.1 = -1 \pmod{11} \equiv 10 \end{gathered}$$

i = 3

$$\begin{gathered} r[3] = 7 \ \% \ 4 = 3 \\ q[3] = \lfloor 7/4 \rfloor = 1 \\ a[3] = 1 - 1.10 = -9 \pmod{11} \equiv 2 \end{gathered}$$

i = 4

$$\begin{gathered} r[4] = 4 \ \% \ 3 = 1 \\ q[4] = \lfloor 4/3 \rfloor = 1 \\ a[4] = 10 - 1.2 = 8 \pmod{11} \end{gathered}$$

Hence we obtain the multiplicative inverse of 7 in mod 11 world to be 8.

9.9 Performing Modular exponentiation in a easier way

9.9.1 Square and multiply

This technique reduces the complexity involved in computing modular exponential to $\mathcal{O}(\log(N))$. Hence is very useful while dealing with big numbers.

Square and multiply uses the binary representation of the exponent

$$0 : \text{square } 1 : \text{square and multiply}$$

Ex:

$$\begin{gathered} 47^{69} \pmod{143} \\ 69 = 1000101 \end{gathered}$$

Using the square and multiply rule with 1000101 we get

$$\begin{aligned} 1 &: 47 \\ 10 &: 47^2 \\ 100 &: (47^2)^2 \\ 1000 &: ((47^2)^2)^2 \\ 10001 &: (((47^2)^2)^2)^2 * 47 \\ 100010 &: ((((47^2)^2)^2)^2 * 47)^2 \\ 1000101 &: (((((47^2)^2)^2)^2 * 47)^2)^2 * 47 \end{aligned}$$

So solving the original problem now requires only 7 operations

$$\begin{aligned}
47 (\text{mod } 143) &= 47 \\
47^2 (\text{mod } 143) &= 64 \\
(47^2)^2 (\text{mod } 143) &= 64^2 (\text{mod } 143) = 92 \\
((47^2)^2)^2 (\text{mod } 143) &= 92^2 (\text{mod } 143) = 27 \\
(((47^2)^2)^2)^2 * 47 (\text{mod } 143) &= 27^2 * 47 (\text{mod } 143) = 86 \\
((((47^2)^2)^2)^2 * 47)^2 (\text{mod } 143) &= 86^2 (\text{mod } 143) = 103 \\
(((((47^2)^2)^2)^2 * 47)^2)^2 * 47 (\text{mod } 143) &= 103^2 * 47 (\text{mod } 143) = 125
\end{aligned}$$

Hence after evaluating 7 modulo operations we get the final answer.

9.9.2 Computer method

Compute 3^{94} (mod 17)

$$94 = 64 + 16 + 8 + 4 + 2$$

$$\begin{aligned}
3^2 (\text{mod } 17) &= 9 (\text{mod } 17) \\
3^4 (\text{mod } 17) &= 13 \equiv -4 (\text{mod } 17) \\
3^8 = (3^4)^2 &\equiv (-4)^2 (\text{mod } 17) \equiv -1 (\text{mod } 17) \\
3^{16} \equiv (3^8)^2 &\equiv (-1)^2 (\text{mod } 17) \equiv 1 (\text{mod } 17) \\
3^{64} \equiv (3^{16})^4 &\equiv (1)^4 (\text{mod } 17) \equiv 1 (\text{mod } 17)
\end{aligned}$$

$$\begin{aligned}
3^{94} &\equiv 3^{64+16+8+4+2} \\
&\equiv 3^{64} 3^{16} 3^8 3^4 3^2 \\
&\equiv (1)(1)(-1)(-4)(9) \\
&\equiv 36 \\
&\equiv 2
\end{aligned}$$

9.10 Euler's Totient function

Euler's totient function counts the positive integers up to a given integer n that are relatively prime to n. It is written as ϕ(n). In other words, it is the number of integers k in the range $1 \leq$ k $\leq$ n for which the greatest common divisor *gcd*(n, k) is equal to 1. The integers k of this form are sometimes referred to as totatives of n.

$$\phi(N) = N \prod_{i=1}^{p} \frac{p_i - 1}{p_i}$$

Where p_i are the prime factors of N.

- If **N** is a prime number then $\phi(N)$ = N - 1
- Totient of product of primes is the product of one less than each prime.
- Totient of a prime power (p^k)
  $\phi(N = p^k) = N\frac{p-1}{p}$
- Totient of an arbitrary number is the product of the totient of the prime powers.
- ϕ(a.b) = ϕ(a).ϕ(b) if gcd(a,b) = 1

Proof for Totient formula:

To prove the Totient formula, all we have to do is prove the Totient formula for a prime power ($N = p^k$), and for the case of product of relatively prime numbers and then from Fundamental theorem of Arithmetic it can be generalized to all numbers.

Number of candidates which are relatively prime to N is (N-1), out of these (N-1) candidates we have to find out how many candidates satisfy gcd(x,N)> 1.

Since x is not relatively prime to N
$\implies$ x, N share a common prime factor (p)
$\therefore$ gcd(x,N) > 1 only if x = m.p for ($1 \leq m < N/p$), hence the number of such candidates is (N/p - 1)

$$\phi(N = p^k) = N - 1 - (\frac{N}{p} - 1)$$
$$\phi(N = p^k) = N\ (1 - \frac{1}{p}\ )$$

Hence the totient for prime power is proved.

The next step is to prove the formula for a product of coprimes.

Let **m** and **n** be relatively prime to each other. Make a rectangular table of the numbers 1 to mn with m rows and n columns, as follows:

$$\begin{array}{ccccc} 1 & m+1 & 2m+1 & \cdots & (n-1)m+1 \\ 2 & m+2 & 2m+2 & \cdots & (n-1)m+2 \\ 3 & m+3 & 2m+3 & \cdots & (n-1)m+3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m & 2m & 3m & \cdots & nm \end{array}$$

The numbers in the rth row of this table are of the form km + r as k runs from 0 to m - 1.

Let d = gcd(r, m). If d > 1 then no number in the r^{th} row of the table is relatively prime to mn, since d | (km + r) for all k. So to count the residues relatively prime to mn we need only to look at the rows indexed by values of r such that gcd(r, m) = 1, and there are ϕ(m) such rows. If gcd(r, m) = 1 then every entry in the r^{th} row is relatively prime to m, since gcd(km + r, m) = 1 by the Euclidean algorithm. The entries in such a row form a complete residue system modulo n. Thus, exactly ϕ(n) of them will be relatively prime to n, and thus relatively prime to mn. We have shown that there are ϕ(m) rows in the table which contain numbers relatively prime to mn, and each of those contain exactly ϕ(n) such numbers. So there are, in total, ϕ(m)ϕ(n) numbers in the table which are relatively prime to mn. This proves the theorem.

From fundamental theorem of arithmetic we know that every number can be represented by an unique prime factorization. And any two given prime numbers are relatively prime. Using this we get the Euler totient function for any arbitrary number.

9.10.1 Euler's Totient theorem

$$x^{\phi(N} \equiv 1 \pmod{N}$$

when gcd(x,N) = 1
Proof:

Consider a set **S** containing all numbers that are less than N and relatively prime to N

$$S = \{ k|\ 0 < k < N \text{ and } gcd(k,N) = 1 \}$$

Let **m** be the cardinality of this set.
Choose an arbitrary integer **x** such that it is relatively prime to N. Now construct a new set by multiplying **x** to every element of the set **S** and then performing mod N.

$$R = \{ j \mid j = x.k \bmod(N) \text{ for each } k \in S \}$$

Properties of the sets defined

1. All elements of S are distinct
2. Cardinality of S and R is **m**
3. R $\subset$ S
4. All elements of R are distinct

Proof for the third point (By contradiction)

Assume j $\in$ R | gcd(j,N) = d where d¿1

$$j = k.x \pmod N$$
$$j = k.x + q.N$$

Since gcd(j,N) is d, j and N can be written as

$$j = c1.d$$
$$N = c2.d$$

Using these substitutions we get

$$c1.d = k.x + q.c2.d$$
$$c1 = \frac{k.x}{d} + q.c2$$

We know that c1 is an integer, for this to be true the first term on RHS should also be an integer. But we know that x,k are relatively prime to N which means x, k are relatively prime to any factor of N as well. $\implies$ The first term on RHS is not an integer as there are no common factors between the numerator and the denominator. Unless d = 1.

Hence every element of R is relatively prime to N as well. But we know that S contains all elements that are relatively prime to N, hence R is a subset of S.

Proof for the fourth point (By contradiction)

Assume that j_1 and $j_2 \in R$ exist such that $j_1 = j_2$

$$j_1 \equiv k_1.x \pmod N$$
$$j_2 \equiv k_2.x \pmod N$$

Since all elements in R are less than N,

$$j_1 = j_2 \implies j_1 \equiv j_2$$
$$\implies k_1.x \equiv k_2.x$$

Since x is relatively prime to N, multiplicative inverse exists for x. So we get

$$k_1 \equiv k_2$$

Since all elements of S are less than N

$$k_1 = k_2$$

We know that all elements of S are unique, hence the above is not possible. Hence all the elements of R are unique as well.

From the 4 properties of the sets, we can say that

$$\mathbf{S = R}$$

Let **P** denote the product of all elements of the sets. Since both the sets contain the same elements the product will be equal.

$$\prod_{i=1}^{m} k_i = \prod_{i=1}^{m} k_i = \prod_{i=1}^{m} k_i.x \pmod N$$
$$\prod_{i=1}^{m} k_i = x^m \prod_{i=1}^{m} k_i \pmod N$$

Since each k is relatively prime to N, it has a multiplicative inverse. Multiplying on both sides by the multiplicative inverse we get

$$1 \equiv x^m \equiv x^{\phi(N)} (\text{mod N})$$

Hence Euler's Totient theorem is proved.

9.11 Discrete Logarithms

Not all positive integers have discrete logarithm.

$$\log_{10}(100) = 2$$
$$\log_{10}(1000) = 3$$
$$\log_{10}(512) = \text{does not exist}$$

NOTE:

- If discrete logarithm exists for a given number, then it need not be unique.
  $\log_8(5) = \{3, 7, 11\}$ (mod 13)
- Discrete logarithms are not monotonic.
  $\log_7(4) \equiv 10$ (mod 13) $> \log_7(5) \equiv 3$ (mod 13)
- b^k is not unique for a given value of k
  4^3 (mod 13) = 12
  10^3 (mod 13) = 12
  12^3 (mod 13) = 12
- $\log_7(8) \equiv 9$ (mod 13)
  $\log_7(4) + \log_7(2) \equiv 10 + 11 \equiv 8$ (mod 13)
  Hence with discrete logarithms log(ab) $\neq$ log(a) + log(b)

Logarithms live in a mod-totient world hence most of the properties of continuous logarithms do not carry over to discrete case.

- $b^x \equiv b^y$ (mod N) if x $\equiv$ y (mod ϕ(N))

Look at the following table where the modulus value is 13

		base												
		0	1	2	3	4	5	6	7	8	9	10	11	12
	0	–	1	1	1	1	1	1	1	1	1	1	1	1
	1	0	1	2	3	4	5	6	7	8	9	10	11	12
e	2	0	1	4	9	3	12	10	10	12	3	9	4	1
x	3	0	1	8	1	12	8	8	5	5	1	12	5	12
p	4	0	1	3	3	9	1	9	9	1	9	3	3	1
o	5	0	1	6	9	10	5	2	11	8	3	4	7	12
n	6	0	1	12	1	1	12	12	12	12	1	1	12	1
e	7	0	1	11	3	4	8	7	6	5	9	10	2	12
n	8	0	1	9	9	3	1	3	3	1	3	9	9	1
t	9	0	1	5	1	12	5	5	8	8	1	12	8	12
	10	0	1	10	3	9	12	4	4	12	9	3	10	1
	11	0	1	7	9	10	8	11	2	5	3	4	6	12
	12	0	1	1	1	1	1	1	1	1	1	1	1	1

Primitive roots generate all relatively prime values.
If g is a primitive root

$$\gcd(a,N) = 1$$
$$a \equiv g^k \pmod{N}$$

for some k

Looking at the above table where N = 13, we obtain 2,6,7,11 as the primitive roots (Note that in the columns corresponding to these bases there are no repetitions).

The primitive roots are also called **generators of the group**

9.11.1 Discrete Logarithm problem

Suppose we know, for a particular choice of n,x,b, that there is a y such that $b^y = x \pmod{n}$; then finding that y is the **Discrete Logarithm Problem modulo n**.

Currently there are no known efficient methods to compute discrete logarithms. Many critical cryptosystems rely on the fact that there is no efficient solution to Discrete Log problem.

Example: Discrete logarithm of 7 base 2 mod 9 ($\text{dlog}_2\ 7 \pmod{9} = ?$)

The above question requires us to find the exponent to which 2 must be raised such that when that number is divided by 9, the remainder obtained is 7 i.e.

$$2^x \equiv 7 \pmod{9}$$

A possible value for x is 4.

$$2^4 = 16 \equiv 7 \pmod{9}$$

10 Famous Theorems in Discrete Mathematics

10.1 Chinese Remainder Theorem

Different representations of a number reveal different types of information about the structure of a number. Chinese Remainder Theorem is another way to represent integers.

Consider a number 123

$$123 \equiv 3 \pmod{12}$$
$$123 \equiv 4 \pmod{7}$$
$$123 \equiv 3 \pmod{5}$$

With CRT the above can be represented as

$$123 \equiv (3, 4, 3) \pmod{(12,7,5)}$$

NOTE: CRT representation for a given integer is not unique

The number of encoding using a given modulus set is equal to product of the modulus values. In the above example the number of possible encodings are 420 (12x5x7)

- The CRT moduli must be pairwise coprime

10.1.1 Converting from CRT to an integer

Steps involved

- $X \equiv (R_1, R_2, .., R_k) \pmod{(M_1, M_2, .. , M_k)}$
- $N = M_1 \times M_2 \times ... \times M_k$
- $$X \equiv \big(\sum_{i=1}^{k} A_i R_i\big) \bmod N$$
- $A_i \equiv 0 \pmod{M_j}\ \forall\ i \neq j$
  - To satisfy this we can make use of the previous point
  $$X \equiv \big(\sum_{i=1}^{k} A_i \frac{N}{M_i} R_i\big) \bmod N$$
- $A_i R_i \equiv R_i \pmod{M_j}$ for $i = j$
  - To satisfy this, we can make use of the fact that all moduli are co-prime
  $$A_i = \Big(\big(\frac{N}{M_i}\big)\Big[\big(\frac{N}{M_i}\big)^{-1} (\bmod M_i)\Big]\Big) (\bmod N)$$

Example: (3,4,3) (mod(12,7,5))

- N = 12x7x5 = 420
- $b_1 = \frac{420}{12} = 35$
- $b_1^{-1} \equiv 35^{-1} \pmod{12} = 11$
- $A_1 = b_1.b_1^{-1} = 35.11 \bmod 420 = 385$
- $b_2 = \frac{420}{7} = 60$
- $b_2^{-1} \equiv 60^{-1} \pmod{7} = 2$
- $A_2 = b_2.b_2^{-1} = 60.2 \bmod 420 = 120$
- $b_3 = \frac{420}{5} = 84$
- $b_3^{-1} \equiv 84^{-1} \pmod{5} = 4$
- $A_3 = b_3.b_3^{-1} = 84.4 \bmod 420 = 336$
- The required number

$$X \equiv (385x3 + 120x4 + 336x3) \pmod{420}$$
$$X \equiv 123 \pmod{420}$$

10.1.2 CRT Addition

This can be achieved by adding the residues.
Ex:

$$23 = (2,3,3) \pmod{(3,4,5)}$$
$$46 = (1,2,1) \pmod{(3,4,5)}$$
$$69 = (0,1,4) \pmod{(3,4,5)}$$

By adding the residues of 23 and 46 we get (3,5,4). Lets convert this to an integer

- $A_1 = 40$
- $A_2 = 45$
- $A_3 = 36$
- $X = (40x3 + 45x5 + 36x4) \pmod{60} \equiv 9 \pmod{60}$

We know that the representation of 9 is (0,1,4). Hence we get

$$(3,5,4) \pmod{(3,4,5)} \equiv (0,1,4) \pmod{(3,4,5)}$$

Hence we get the CRT representation of the sum by adding the residues of individual numbers

10.1.3 CRT Multiplication

1. Multiplication by an integer is trivial using the distributive property.

$$2.(2,3,3) \equiv (4,6,6)$$

2. Multiplying two CRT representations

   Consider the moduli (3,4,5)

$$z \equiv (40.x_3 + 45.x_4 + 36.x_5)(40.y_3 + 45.y_4 + 36.y_5)$$
$$z_3 = (40.x_3).(40.y_3) \pmod 3$$
$$z_4 = (45.x_4).(45.y_4) \pmod 3$$
$$z_5 = (36.x_5).(36.y_5) \pmod 3$$

   Ex:

$$23.46 = 1058 \equiv 38 \equiv (2,2,3) \pmod{60}$$
$$(2,3,3).(1,2,1) \equiv (2,6,3) \equiv (2,2,3) \pmod{60}$$

3. Finding Multiplicative inverse

   Again consider the same moduli

$$x^{-1}.x \equiv (1,1,1) \pmod{60}$$
$$(x_3^{-1}.x_3,\ x_4^{-1}.x_4,\ x_5^{-1}.x_5) \equiv (1,1,1) \pmod{60}$$

   Ex:

   - $23^{-1} \equiv (2,3,3)^{-1} \equiv (2^{-1},\ 3^{-1},\ 3^{-1}) \pmod{60}$
   - $(2,3,3)^{-1} \equiv (2,3,2) \pmod{60} \equiv 47$
   - $(2,3,3).(2,3,2) \equiv (1,1,1) \pmod{60}$

10.1.4 CRT Exponentiation

Consider moduli 60 (3,4,5)

$$x^n \equiv (x_3,\ x_4,\ x_5)^n \pmod{60}$$
$$x^n \equiv (x_3^n,\ x_4^n,\ x_5^n) \pmod{60}$$

NOTE: Using CRT representation in the exponent is challenging because the numbers in the exponent live in the totient world

10.2 Prime Number Theorem

The prime number theorem describes the asymptotic distribution of prime numbers among the positive integers. This theorem was proved by **Jacques Hadamard** and **Charles Jean de la Vallee Poussin** using ideas introduced by **Riemann**.

$$\pi(\mathrm{N}) \approx \frac{N}{ln(N)}$$

where π(N) is the **prime-counting function**. From the above we can see that the probability that a random integer not greater than N is prime is very close to $\frac{1}{ln(N)}$.
In other words, the average gap between consecutive prime numbers among the first N integers is roughly $ln(N)$.

The exact form for the prime-counting function was given by **Riemann**. The form of that function can be seen in the following link

`https://en.wikipedia.org/wiki/Prime-counting_function`

10.3 Sieve of Eratosthenes

This is an ancient algorithm for finding all prime numbers up to any given limit.

It does so by iteratively marking as composite the multiples of each prime, starting with the first prime number 2.

The time complexity of this technique is $\mathcal{O}(nln(ln(n)))$

10.3.1 General number field Sieve

In number theory, the **general number field Sieve** is the most efficient classical algorithm known for factoring integers

10.4 Fermat's Little Theorem

Fermat's Little Theorem is proved as a corollary of Euler's theorem.

It states that if p is a prime number, then for any integer $a \in$ (1, p-1), $a^p - a$ is an integer multiple of p.
In the notation of modular arithmetic, this is expressed as

$$a^p \equiv a(\mathrm{mod}\ p)$$

If a and p are coprime ($\implies$ a^{-1} exists), Fermat's little theorem is equivalent to the statement that $a^{p-1} - 1$ is an integer multiple of p.

$$a^{p-1} \equiv 1(\mathrm{mod}\ p)$$

Proof: Since p is a prime number, all the numbers less than p are relatively prime with p $\implies \phi(p) = p - 1$. Using this in Euler's theorem gives

$$a^{\phi(p)} = a^{p-1} \equiv 1(\text{mod } p)$$

Hence Fermat's Little theorem is proved.

Corollary: $a^{-1} = a^{p-2}$ (mod p)

10.5 Primality Test

Primality tests seek to determine if a number is prime without factoring it.

10.5.1 Fermat's primality test

This test gives a probability that a given number can be prime.

- If a number fails Fermat's primality test, it is composite for sure
- If a number passes Fermat's primality test, it may be a prime number.

The algorithm is as follows

- Pick a random value for **p** of the right size
- Pick value of **a** such that 1 < a < p-1
- Check if $a^{p-1} \equiv 1$ (mod p)
  - if **NO**: **a** is a Fermat witness to compositeness of **p**
  - if **YES**: two possibilities
    - **p** is prime
    - **p** is a pseudoprime to base **a**; **a** is a Fermat liar

Consider a toy example:

- Choose **p** is 15
- Pick **a** = 4
- Check if $4^{14} \equiv 1 \pmod{15}$
  - The above relation is true.

Hence we see that 15 passes Fermat's primality test. But we know that 15 is a composite number. Hence based on the above result all we can claim is:
15 is a pseudoprime to base 4; 4 is a Fermat liar.

Since it passed lets consider a different value of **a** and repeat the test.

- Pick **a** = 2
- Check if $2^{14} \equiv 1 \pmod{15}$
  - No it is congruent to 4.

Hence 15 fails the primality test. And we can confidently say that:
15 is a composite number; 2 is a Fermat witness for 15

10.5.2 Carmichael numbers

Carmichael numbers are composite numbers which pass a Fermat primality test to every base b relatively prime to the number, even though it is not actually prime.

The smallest Carmichael number is 561. There are infinitely many Carmichael numbers, but are very rare.

This makes tests based on Fermat's Little Theorem less effective than strong probable prime tests such as the **Baillie–PSW primality test** and the **Miller–Rabin primality test**.

Fermat's test is a very simple test, and does not take much time. Hence this is usually the first test conducted in a suite of Primality tests. Few of the stronger primlaity tests take more time. Lets see some of the other simpler tests that can be performed even before going for Fermat's primality test

- Odds of a random 500-bit number being a prime is 1 in 350
- Sieved against 2, increases this probability to 1 in 175
- Sieved against first 15 primes, increases this probability to 1 in 50
- Sieved against first 95 primes, increases this probability to 1 in 30 (**Diminishing returns**)

10.5.3 Miller-Rabin Primality test

Miller-Rabin primality test uses the idea of finding non-trivial square roots of 1 as a means to establish probabilistically if a given number is prime or not.

$$x^2 \equiv 1 \pmod{N}$$

The trivial roots for this are x = 1 and x = -1 (x = N-1). But however if N is a composite number then we have non trivial roots to this equation as well.
Ex: $3^2 \equiv 1 \pmod 8$

$$x^2 - 1 \equiv 0 (\text{mod N})$$
$$(x-1)(x+1) \equiv 0 (\text{mod N})$$

From the above we see that **N** divides the product of these two factors. But in case **N** is composite some of its factors may divide (x-1) and some of the other factors divide (x+1).

If **N** is prime, it will either divide (x - 1) or (x + 1) since it is not possible to split the factors of N in a non trivial way (unlike the composite case). Which means that

$$x \equiv 1 (\text{mod N})$$
$$x \equiv -1 (\text{mod N})$$

Hence there are only trivial roots to the equation if **N** is prime.

To find the non trivial square root, we can use Fermat's Little theorem.
Remember that

$$a^{(n-1)} \equiv 1 \pmod{n} \text{ IF n is prime}$$

We know that 2 is the only prime number, so any other candidate will be an odd number. And hence that number minus 1 will be even. So (n-1) can be represented as

$$n - 1 = 2^s d$$
$$a^{(n-1)} = a^{2^s d} = x^2$$

So if this is congruent to 1, then the square root of this number could be the potential nontrivial root. Now if

$$a^{2^{s-1} d} \equiv 1 \pmod{n}$$

Then the square root of the above number could be a potential candidate, and we can proceed our way backward taking square root at every step and then checking if that value is congruent to 1 until we reach a^d. At this point computing square root is not a simple task, and hence we assume it is prime, and then repeat the same procedure with the next candidate for a.

So the algorithm proceeds in the following manner
(going backwards from $a^{(n-1)}$ to a^d)

- If $x^2 \not\equiv 1 \pmod{n}$, then **n** is not prime
- If $x^2 \equiv 1 \pmod{n}$, then **n** may be prime, is x trivial?
  - If $x \not\equiv \{1,-1\} \pmod{n}$, n is not prime
  - If $x \equiv -1 \pmod{n}$, test is passed, n may be prime
  - If $x \equiv 1 \pmod{n}$, n may be prime, test continues

    $$x \equiv y^2 \equiv 1 \pmod{n}$$

    test continues with y

Miller-Rabin primality test algorithm

going forward from a^d to $a^{(n-1)}$

- **Pick a random value for *n***
  - **Prefilter with sieve or Fermat's Primality Test if desired**
- **Compute (*n*-1)**
- **Determine the value of *s***
  - **Number of times division by 2 needed to get an odd *d***
- **Calculate $x \equiv a^d$**
  - **If congruent to -1, +1, end test with PASS (possible prime)**
- **Square *x* a total of s times and, at each step**
  - **If $x \equiv -1 \pmod{n}$, end test with PASS (possible prime)**
- **End test with COMPOSITE (is not prime)**

Ex:

- Pick n = 561
- $560 = 2^4 \text{x} 5 \text{x} 7 \implies$ s = 4, d = 35
- Choose a = 7
- $7^{35} \equiv 241 \pmod{561}$
  - Not congruent to {1, -1}, so continue the test by squaring it
- $7^{35^2} \equiv 298 \pmod{561}$
  - Not congruent to {1, -1}, so continue the test by squaring it
- $7^{70^2} \equiv 166 \pmod{561}$
  - Not congruent to {1, -1}, so continue the test by squaring it

- $7^{140^2} \equiv 67 \pmod{561}$
  - Not congruent to {1, -1}, so continue the test by squaring it
- $7^{280^2} \equiv 1 \pmod{561}$
  - This is the final step (s=4) and the value obtained is not congruent to -1. Hence 561 is not prime.

Ex:

- Pick n = 563
- 562 = 2x281 $\implies$ s = 1, d = 281
- Choose a = 7
- $7^{281} \equiv 1 \pmod{563}$
  - Test is passed. 563 may be a prime number

We can consider a different value for **a** and repeat the test, to make the probability of 563 being a prime number even higher.

11 Asymmetric Cryptography

In Asymmetric Cryptography or Public Key cryptograhy a pair of keys are used for encryption and decryption. Unlike symmetric cryptography where the same key is used for encryption and decryption.

With symmetric cryptography or private-key cryptography the security relies on the key secrecy, if the information about the key is compromised then the attacker can easily break the crypto system.

With asymmetric cryptography or public-key cryptography two separate keys are used for encryption and decryption. Anyone with the knowledge of the public key can encrypt messages or verify signatures but cannot decrypt messages or create signatures.

Assymetric cryptography was developed to solve the most difficult problem associated with symmetric cryptography which are:

- key distribution
- Digital Signature[6]

In asymmetric cryptography one key is used for encryption and the other is used for decryption. It is infeasible to derive the private key from the public key or the cipher text.

11.0.1 Terminology

- plaintext - p
- ciphertext - c
- Private key (k_i) - User i's private key
- Public key (K_i) - Associated with user i and paired with k_i

For any p

$$\text{Dec}_1(k_i, \text{Enc}_1(K_i,p)) = p \text{ for confidentiality}$$
$$\text{Dec}_2(k_i, \text{Enc}_2(K_i,p)) = p \text{ for authentication}$$

11.0.2 Authentication and confidentiality

Alice encrypts a message using her private key and transmits it to Bob. Bob uses Alice's public key to decode the message. Since the key used for decrypting is public anyone can decrypt it. But since Alice's public key is used for decoding, this ensures that Alice was the actual source of the message and hence **authenticates** her identity.

[6] A digital signature is a mathematical scheme for verifying the authenticity of digital messages or documents

When Alice wants to communicate with Bob she uses Bob's public key for encrypting data, this ensures that only Bob will be able to decrypt it. As he is the only one with the knowledge of his private key. This provides **confidentiality**.

- Alice uses K_B Bob uses k_B $\implies$ Confidentiality
- Alice uses k_A Bob uses K_A $\implies$ Authentication

11.0.3 Requirements

- Generation of key pair should be computationally easy
- Encryption and Decryption process should be computationally easy
- Computationally infeasible to derive private key from public key
- Computationally infeasible to derive plaintext from ciphertext and public key
- Encryption with public key can be decoded using private key.
  Encryption using private key can be decoded using the public key

11.0.4 One way function

Computation of $y = f(x)$ is easy, but the computation of the inverse is infeasible. In cryptography terms

- $y = f_k(x)$
  Easy to compute with knowledge of x and k
- $x = f_k^{-1}(y)$
  Easy to compute with knowledge of y and k
- $x = f_k^{-1}(y)$
  Infeasible to compute if k is unknown

The development of a practical asymmetric cryptography scheme depends on the discovery of a suitable trapdoor one-way function

11.0.5 Applications

1. Encryption
2. digital signature
3. Key exchange

11.1 RSA Algorithm

Published by Rivest, Shamir and Adleman in 1976. This is the most widely used asymmetric encryption technique. Keys used are typically 1024-1096 bits long. The security provided by RSA algorithm depends on an assumption that given a large number n it is difficult to obtain its prime factorization.

$$\text{n} = \text{p x q}$$

Where p and q are large integers.
In RSA, the publick key e and private key d are derived from p and q. n and e are public, due to the assumption that prime factorization of large numbers is difficult, even with the knowledge of n, the attacker wont be able to obtain p and q.

11.1.1 Working of RSA algorithm

- Generate two distinct prime numbers p and q
- Let $n = pq$
- Since p,q are prime ϕ(n) = (p-1)(q-1)
- Select an integer e s.t. e $<$ ϕ(n) and $gcd(e, \phi(\text{n})) = 1$
- Public key is the pair (e, n)
- Choose d s.t. $de \equiv 1 \pmod{\phi(\text{n})} \implies \text{d} \equiv \text{e}^{-1} \pmod{\phi(\text{n})}$
- Secret key is the pair (d, n)

Encryption is done as follows

$$c = \text{rem}(m^e, n) = m^e \pmod{\text{n}}$$

Decryption is done as follows

$$m = \text{rem}((c)^d, n) = (c)^d \pmod{\text{n}}$$

11.1.2 Proof of RSA algorithm

$$m = (m^e)^d \pmod{\text{n}} = (m)^{ed} \pmod{\text{n}} \tag{1}$$

We know that

$$\begin{aligned} & de \equiv 1 (\text{mod } (p-1)(q-1)) \\ \implies & de - r(p-1)(q-1) = 1 \end{aligned}$$

Using this in (1) we get

$$m = m^{1+r(p-1)(q-1)} \pmod{\text{n}} \tag{2}$$

Since n is the product of p and q, equation (2) is also equal to

$$m = m^{1+r(p-1)(q-1)} \text{ (mod p)} \tag{3}$$
$$m = m^{1+r(p-1)(q-1)} \text{ (mod q)} \tag{4}$$

From Fermat's Little Theorem we know that

$$m^{(p-1)} \equiv 1 \text{ (mod p) if } m \not\equiv 0 \text{ (mod p)}$$
$$m^{(q-1)} \equiv 1 \text{ (mod q) if } m \not\equiv 0 \text{ (mod q)}$$

Using this (3) and (4) reduces to

$$m^{1+r(p-1)(q-1)} \text{ (mod p)} \equiv m.m^{h(p-1)} \text{ (mod p)} \equiv m.1^h \text{ (mod p)} \equiv m \text{ (mod p)}$$
$$m^{1+r(p-1)(q-1)} \text{ (mod q)} \equiv m.m^{j(q-1)} \text{ (mod q)} \equiv m.1^j \text{ (mod q)} \equiv m \text{ (mod q)}$$

Where h = r(q-1) and j = r(p-1).
Hence the decryption technique is proved to be correct for (mod p) and (mod q).

$\implies$ $p \mid ((c)^d - m)$ and $q \mid ((c)^d - m)$

Since n is a product of p and q

$$n \mid ((c)^d - m)$$

Hence we obtain

$$m = (c)^d \text{ (mod n)}$$

Example: Let p = 11, q = 13

- n = 11*13 = 143
- $\phi(n)$ = 10*12 = 120
- Select e = 11 which is coprime to 120
- Multiplicative inverse of e in mod 120 world is 11 itself $\implies$ d = 11

$$\text{de} \equiv 1 \text{ (mod } \phi(\text{n}))$$
$$11 \text{ x } 11 = 121 \equiv 1 \text{ (mod 120)}$$

- Assume the message to be transmitted is 7.

$$\text{c} = 7^{11} \text{ (mod 143)} \equiv 106 \text{ (mod 143)}$$

- The decrypted message is

$$\text{d} = 106^{11} \text{ (mod 143)} \equiv 7 \text{ (mod 143)}$$

11.1.3 RSA security

- Larger the number of bits in private key, better is the security provided against brute force attack
- **Timing based side-channel attacks**[7] is a serious threat to RSA encryption. One counter measure to tackle this is to add random delays after certain operations during encryption
- RSA is vulnerable to **Chosen ciphertext attack**. A possible counter measure is random padding of plain text.
- Another vulnerability is because RSA encryption is multiplicative

$$\text{Enc(m1).Enc(m2)} = \text{Enc(m1.m2)}$$

11.2 Diffie-Hellman Key Exchange

This is a standard protocol for key exchange. Its security is based on the computational hardness of solving the discrete logarithm problem given a large number.

11.2.1 Setup

Alice and Bob agree on using certain global parameters a and p. Where p is a prime number and a is a primitive root of p.

Each user randomly selects $X < p$ and computes

$$Y = a^x \pmod p$$

Y is public, X is private. $\{ X_A, Y_A \}$ for Alice and $\{ X_B, Y_B \}$ for Bob For an attacker monitoring this exchange, knowing the value of Y would have to solve a discrete logarithm problem to obtain the corresponding X.

X_A and X_B never leave the user, hence they are private, and the key obtained using these is also private. Though Alice and Bob know the key K, they do not have knowledge of the private key of the other person due to Discrete Logarithm problem.

[7] The size of the operands is inferred from the time taken to perform a particular operation

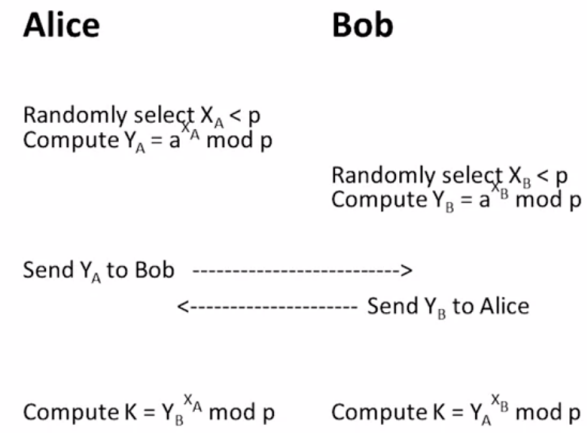


Figure 26: Diffie-Hellman Key exchange protocol

11.2.2 Man in the middle attack

This is an active attack, where the attacker not only eavesdrop but also modifies the contents of the data.

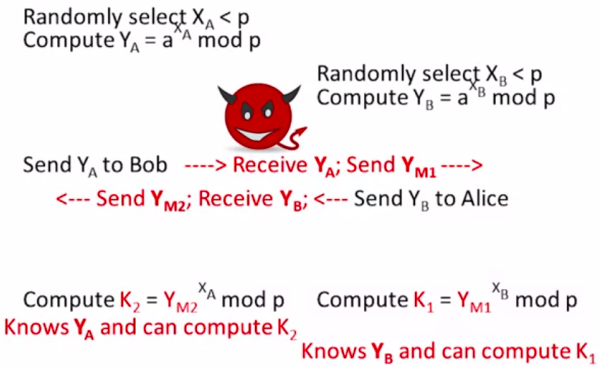


Figure 27: Man in the middle attack

After this attack instead of Bob and Alice sharing the same key, Bob and attacker share the key K1, Alice and attacker share the key K2.

If authentication can be used for the messages shared, then this attack can be prevented.

12 Key Distribution and Management

In case of symmetric cryptography the key used for encryption should also be used for decrypting the ciphertext. Hence the key must be shared between two parties. Similarly in case of asymmetric cryptography the public keys must be shared. Hence key distribution is a very important part of cryptography.

Key distribution approaches:

- User A can select a key and physically deliver it to User B
- Third party can physically deliver the key to A and B
- If A and B had communicated previously, they can use a previous key to encrypt and communicate the new key.
- Trusted third party to deliver the key

12.1 Key Hierarchy

1. Session keys
2. Master keys

Using session keys:
Techniques like Diffie-Hellman can be used to share the session key, Even if an attacker gains access to this he will be able to use it for a short term, as the key being used changes from session to session.

Purpose of Master key:
This is used to encrypt session keys, and is shared between the user and key distribution center[8]

12.2 Public Key Authority

The main idea here is to have a centralized directory, where the public keys of various users is stored. Whenever Alice wants to communicate with Bob, she must first contact the authority to obtain Bob's public key. The authority responds to this request with Bob's public key but encrypts it with its own private key. When Alice receives this message, she can use authority's public key to first authenticate the message and then decrypt it to obtain Bob's public key. She can now directly communicate with Bob. When Bob wants to communicate with Alice he must go through the same procedure.

[8]Key distribution center is responsible for distributing keys to pairs of users

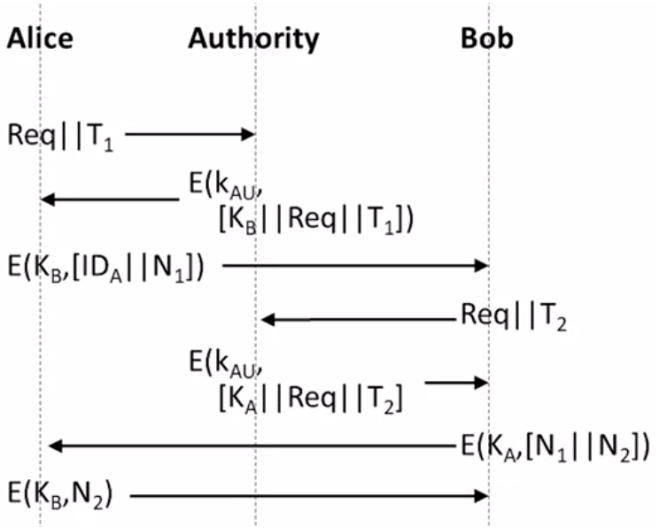


Figure 28: Public key authority

T_1 refers to the time stamp (time at which request is made), Authority appends the timestamp to the requested data to make sure to which request this response corresponds to. And moreover the message sent to Bob is encrypted with Bob's public key and hence only Bob will be able to decrypt it.

One drawback of this system is that it is very slow. And also there is no mechanism to ensure the integrity of the entries in the public directory maintained by the Authority. These drawbacks motivated the use of **Public key certificates**.

12.3 Public key Certificate

This builds on public-key authority. The main feature in public key certificate that overcomes the drawbacks of public-key authority is that it contains a **validity period**[9] and all the records in the directory are digitally signed by **Certificate Authority (CA)** this provides authentication.
Public-key Certificate Requirements

1. Any user can read a certificate
2. Any user can verify the certificate
3. Only CA can create/update the certificate

12.3.1 Public-key Certificate protocol

Alice first sends her public key to CA, CA then responds with a digitally signed certificate for Alice that includes validity period, public key of Alice, identity

[9]The duration for which this data entry can be trusted

of Alice. Alice can now share this certificate with anyone she wants to communicate with. Similarly Bob can go through the above procedure to get his own certificate and share it with anyone he wants to communicate with. Only after the certificates are shared, communication is possible.

Public Key Infrastructure is the system comprised of hardware, software, people policies and procedures needed to create, manage, store, distribute and revoke digital certificates.