PUBLIC KEY CRYPTOGRAPHY

AMAN SINGH

GAURAV KUMAR

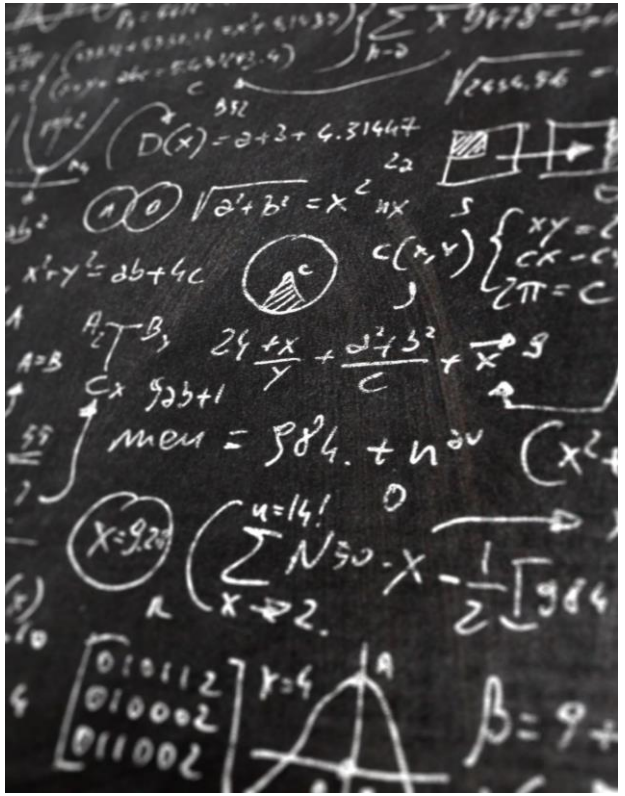
SHUBHAM TIWARI

DEEPENDRA SINGH

OUR TEAM MEMBER

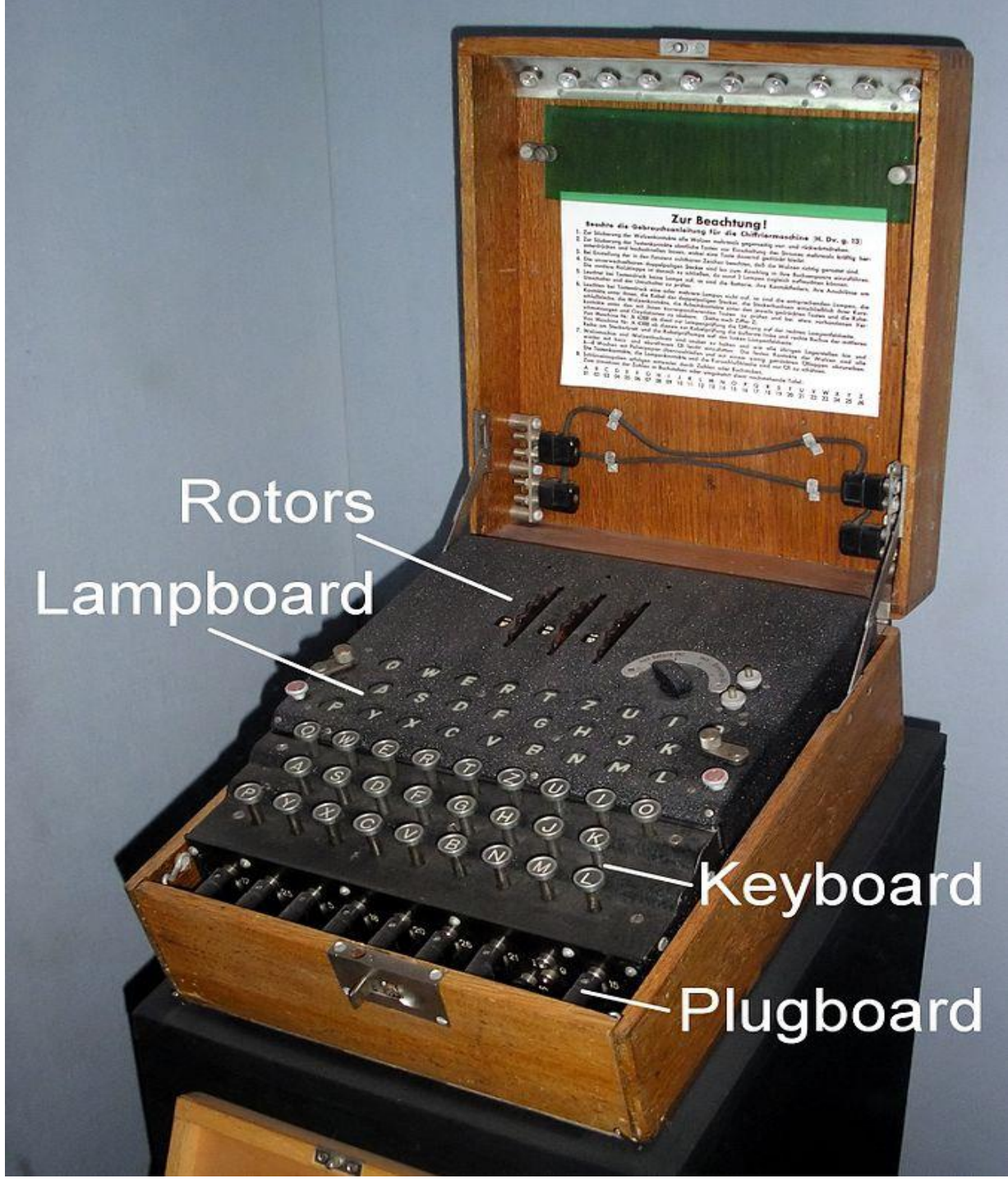
Brief Overview Of Cryptography

- The earliest form of cryptography was the simple writing of a message, as most people could not read (New World, 2007). In fact, the very word cryptography comes from the Greek words kryptos and graphein, which mean hidden and writing, respectively (Pawlan, 1998). Early cryptography was solely concerned with converting messages into unreadable groups of figures to protect the message's content during the time the message was being carried from one place to another. In the modern era, cryptography has grown from basic message confidentiality to include some phases of message integrity checking, sender/receiver identity authentication, and digital signatures, among other things (New World, 2007).



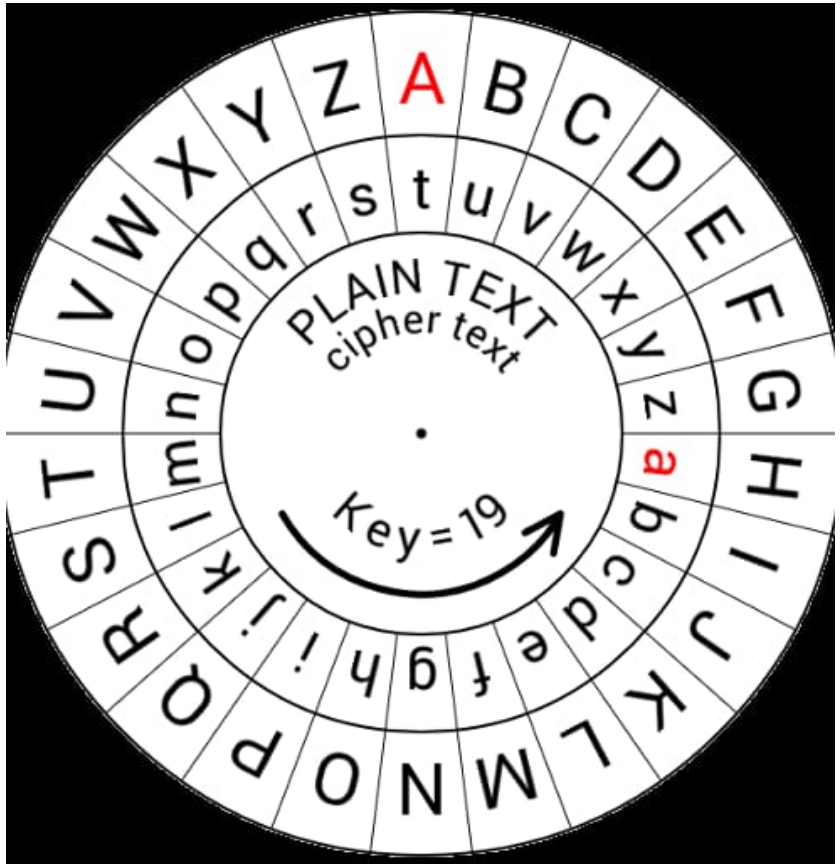
History Of Cryptography

- The history of cryptography goes way back to ancient times. One of the earliest methods was the Caesar cipher, named after Julius Caesar. In this method, each letter in the message is shifted by a certain number of positions down the alphabet. For example, if you shift each letter by three positions, "A" becomes "D," "B" becomes "E," and so on.
- Over the centuries, people developed more sophisticated methods. In the Middle Ages, there were secret codes and ciphers used by military leaders and diplomats. These codes could be based on replacing letters with symbols or using special grids to encrypt messages.
- During World War II, the Enigma machine was a famous cryptographic device used by the Germans. It scrambled messages in a complex way, and only those with the correct settings could decipher them. The Allies worked hard to break the Enigma code, and when they succeeded, it played a crucial role in winning the war.
- In the modern digital age, cryptography is essential for securing information online. It involves complex mathematical algorithms and keys to encrypt and decrypt messages. Public-key cryptography, for example, uses a pair of keys – one public and one private – to secure communications over the internet.



Historical Cryptography

- **Caesar Box**
- **The "Caesar Box," or "Caesar Cipher," is one of the earliest known ciphers. Developed around 100 BC, it was used by Julius Caesar to send secret messages to his generals in the field. In the event that one of his messages got intercepted, his opponent could not read them. This obviously gave him a great strategic advantage. So, what was the code?**
- **Caesar shifted each letter of his message three letters to the right to produce what could be called the ciphertext. The ciphertext is what the enemy would see instead of the true message. So, for example, if Caesar's messages were written in the English alphabet, each letter "A" in the message would become a "D," the "B's" would become "E's," and the "X's" become "A's." This type of cipher is appropriately called a "shift cipher."**
- **Caesar's magic number was three, however, a modern day use of the Caesar Cipher is a code called "ROT13." ROT13, short for "rotate by 13 places," shifts each letter of the English alphabet 13 spaces. It is often used in online forums to hide information such as movie and tv show spoilers, solutions to puzzles or games, or offensive material. The code is easily crackable, however, it hides the information from the quick glance.**
- **Cryptanalysis: Breaking a Caesar Box**
- **The hardest part of breaking a Caesar Box is figuring out the language of the message that it encodes. Once the code cracker figures this out, two scenarios are considered. Either the "attacker" utilizes a technique such as frequency analysis, or they use what is referred to as a brute force attack.**
- **In the first instance, the attacker knows that certain letters are used more frequently than others. For example, A,E,O, and T are the most commonly used letters, while Q, X, and Z are the least.**

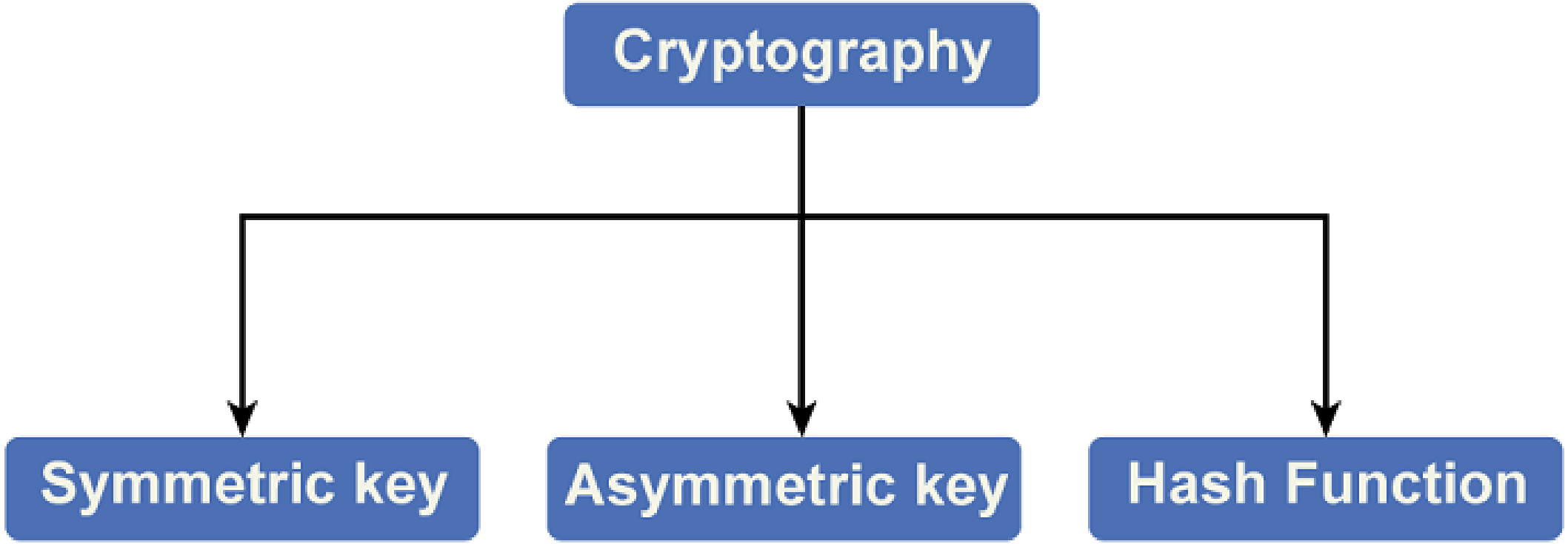


The Vigenere Cipher

- The Vigenère Cipher, created in the 16th century, uses an element not found in a Caesar Cipher: a secret key. The creator of the code picks any word or combination of letters at random to be the key, for example, “DOG.” The keyword chosen will then be matched to the plaintext message that you want to encrypt, for example, "ATTACK." You can see that the keyword "dog" is shorter than the word “attack" by three letters. In this case, repeat your key until it matches the number of letters in your plaintext message. In this case, you would then have "DOGDOG."
- The columns are the letters of the secret key, while the rows are the letters of the plaintext message. So for our example, the first letter of our key is “D,” while the first letter of our plaintext word is “A.” So, find where they intersect on the chart, and you will find the first letter of our ciphertext, which is “D.” Next, the second letters of our key and plaintext words are “O” and “T” respectively. They intersect at “H.” You would continue this until you complete all six letters.

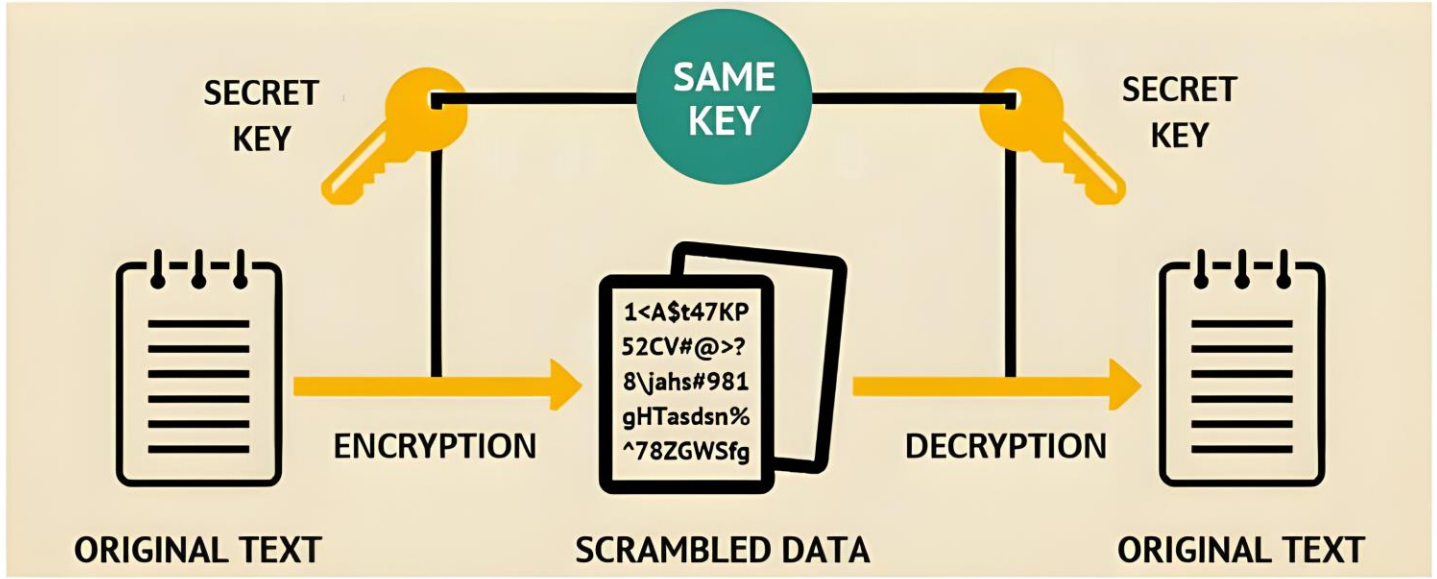
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T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

Types of cryptography



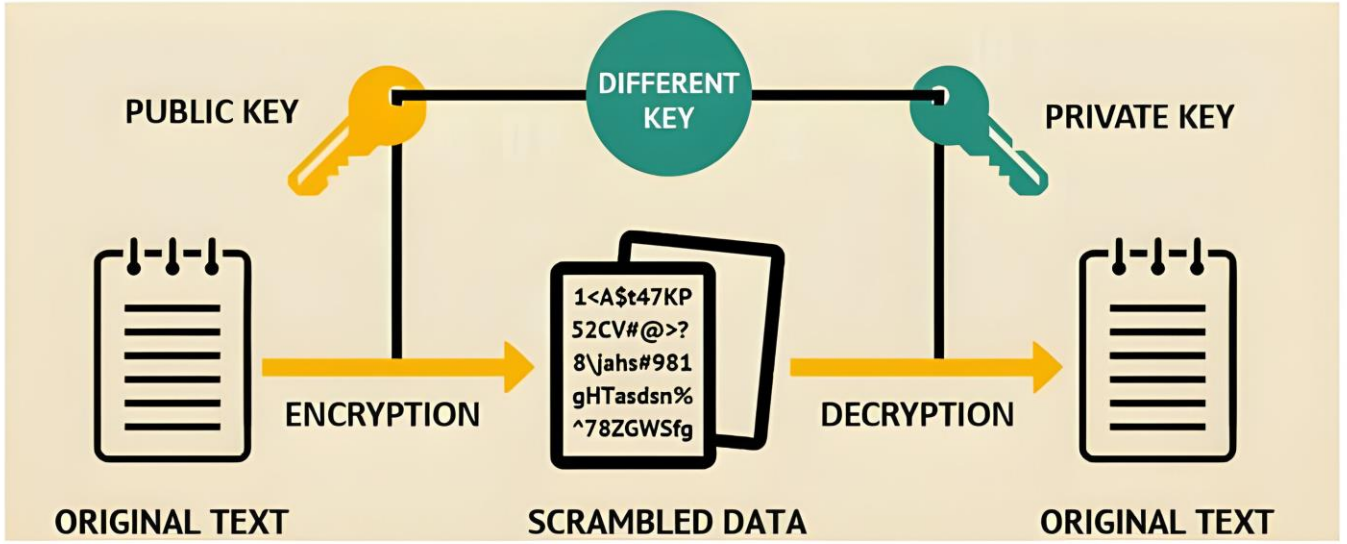
Symmetric Key Cryptography

Symmetric key cryptography is the category where the same key is used for both the encryption and decryption of information.



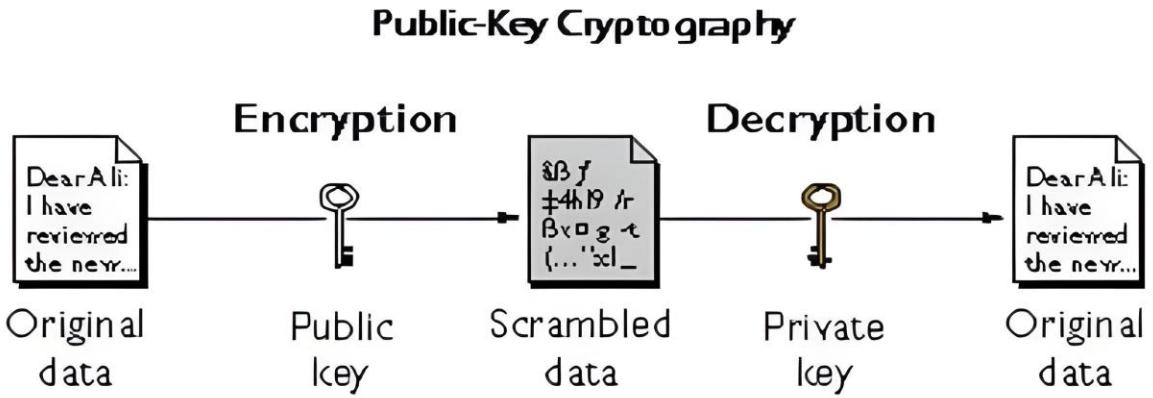
Asymmetric Key Cryptography

In asymmetric key cryptography, there are two keys at play. A public key and a private key. The public key is used to encrypt the data pre-transit, and the private key is used to decrypt the data post-transit.



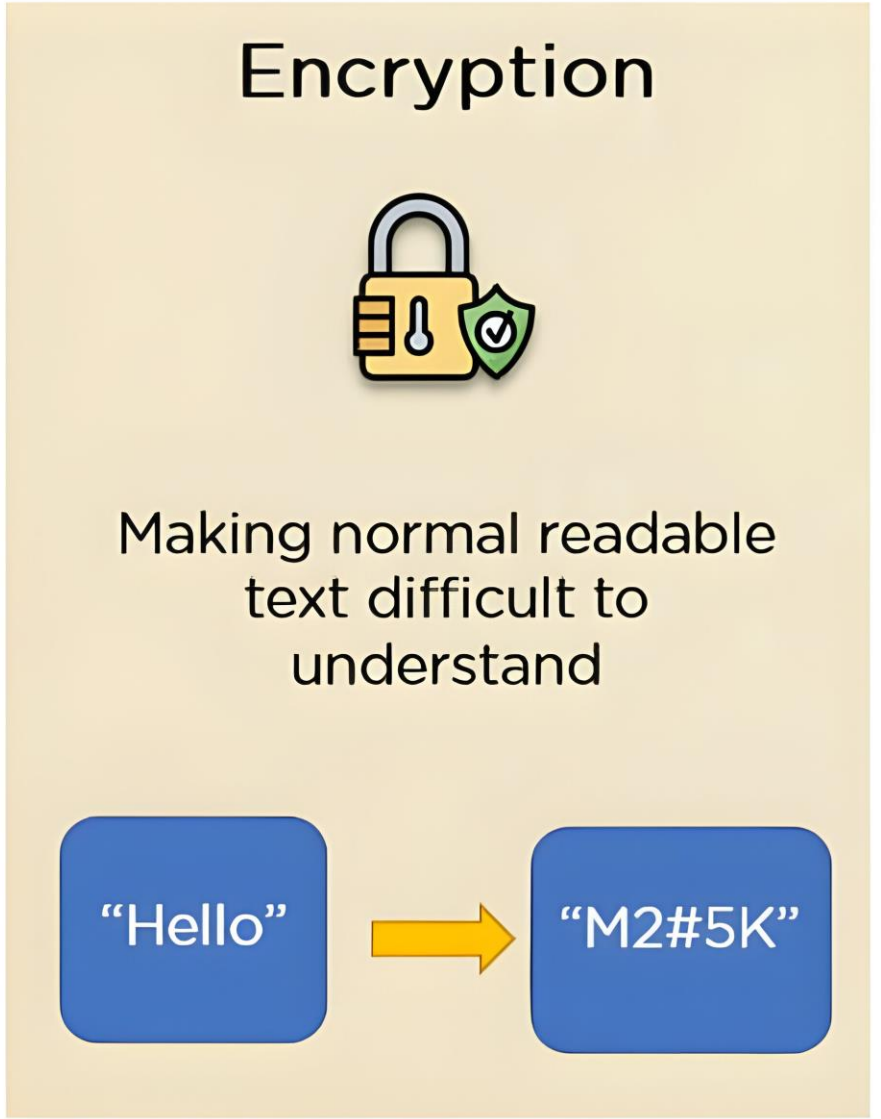
Public Key Cryptography

- Public key cryptography involves a pair of keys known as a public key and a private key (a public key pair), which are associated with an entity that needs to authenticate its identity electronically or to sign or encrypt data. Each public key is published and the corresponding private key is kept secret. Data that is encrypted with the public key can be decrypted only with the corresponding private key.



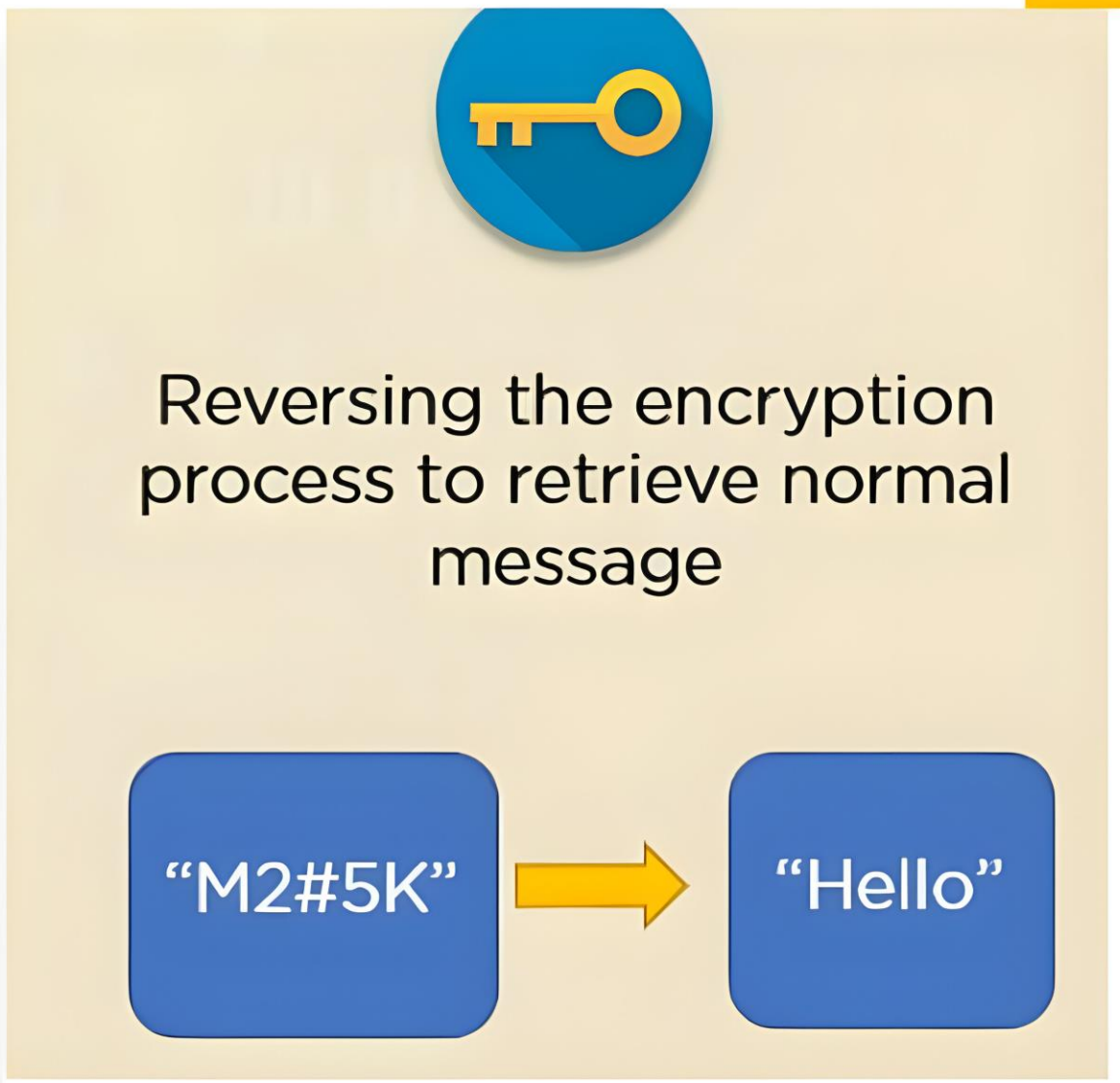
- **Encryption**

  Is the process of scrambling the information, to avoid third parties from comprehending the message even if it is intercepted. This scrambling is done using specific mathematical calculations and steps, often collectively known as ciphers. Along with the cipher, it uses an encryption key to encrypt the message.



- **Decryption**

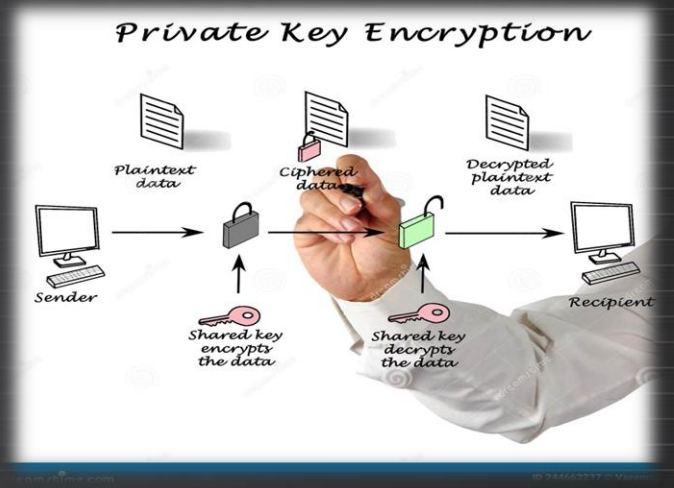
  is the process of reversing the work done by encryption. It converts the scrambled information into its original form so that the data is readable again. Usually, the encryption key which is used to scramble the data can decrypt the data, but that varies depending on the type of cryptography used. Irrespective of whether or not they are the same, a key is mandatory for both the encryption and decryption of data.



Important terms related to Cryptography :

Modular Arithmetic:

Definition: Modular arithmetic is a type of arithmetic that deals with integers and their remainders when divided by a fixed positive integer, called the modulus. It is denoted by the symbol %.



Prime Numbers:

Definition: A prime number is a natural number greater than 1 that is not a product of two smaller natural numbers. In cryptography, prime numbers play a crucial role in algorithms like RSA, where the security relies on the difficulty of factoring the product of two large prime numbers.

Plaintext
This is the readable message or data that is fed into the algorithm as input

.

Encryption algorithm

The encryption algorithm performs various transformations the plaintext.

Ciphertext

This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.

That is provided as input

Decryption algorithm:

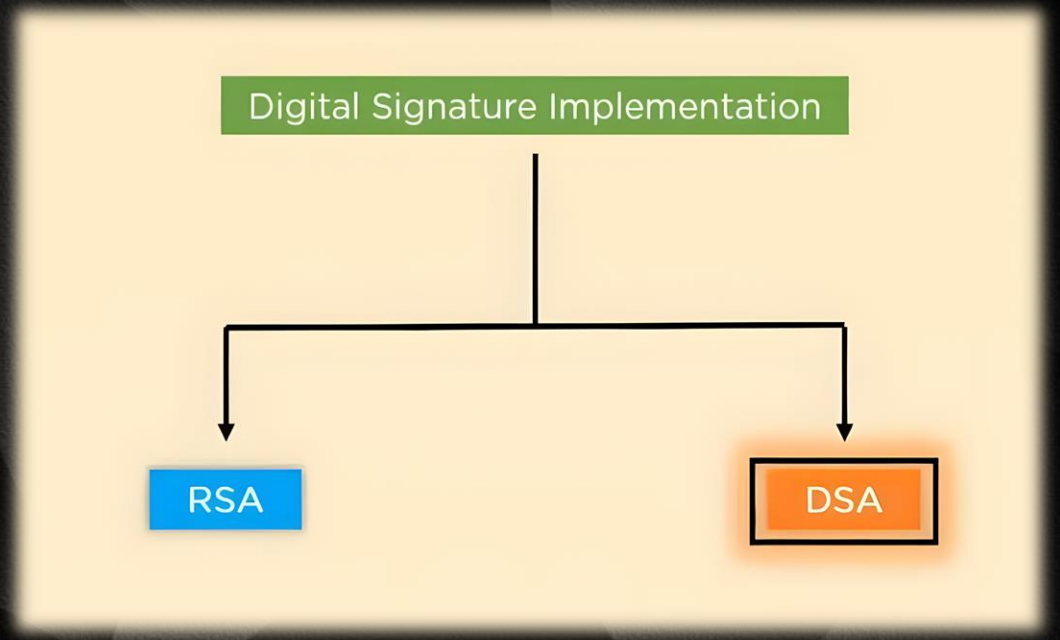
This algorithm accepts the ciphertext and the matching key and produces the original plain text.The essential steps are the following.

What Is Hashing?

Hashing is the process of scrambling a piece of information or data beyond recognition. They are designed to be **irreversible**. We pass the input through a hash function to calculate the **Hash Value** or **Digest**.

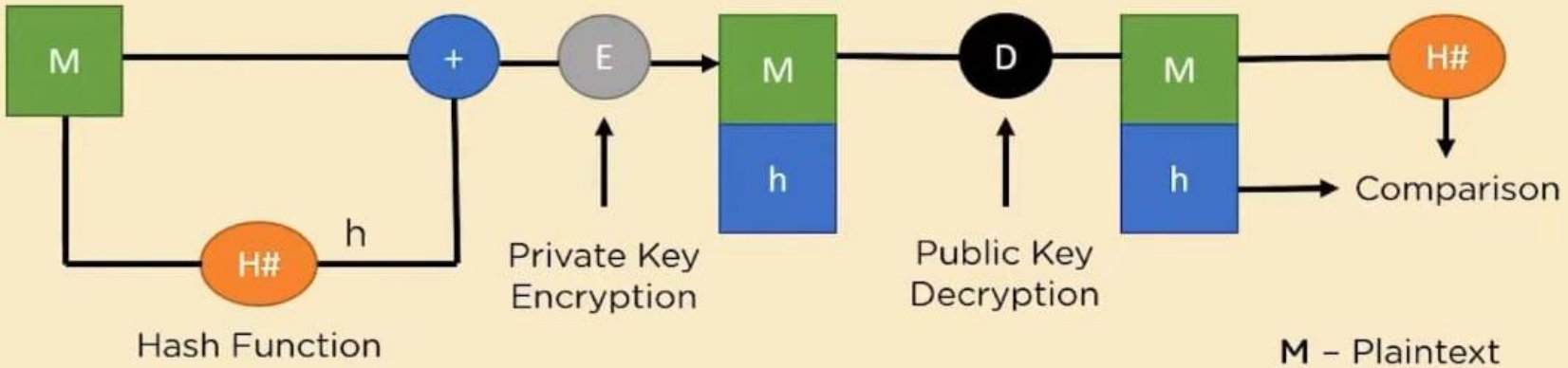
Digital signature:

The sender “signs” a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message



What Are Digital Signatures?

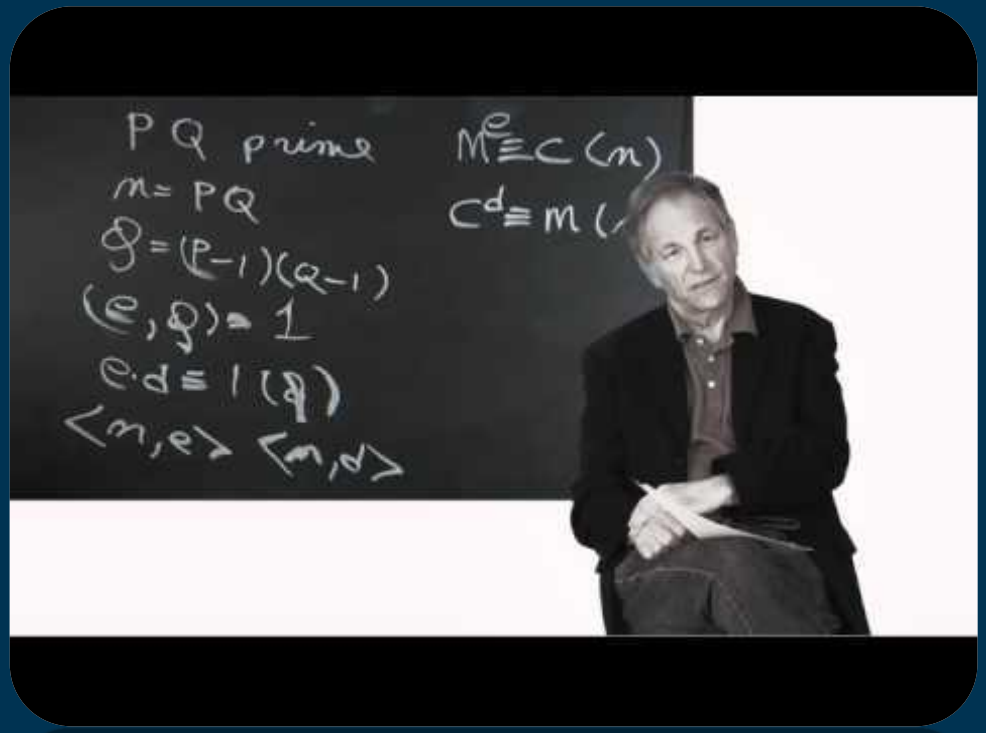
- Mechanism to **determine authenticity** of a document file
- Uses **public key** cryptography mechanism
- Helpful to authenticate **long distance** official communication channels



Every Egyptian received two names, which were known respectively as the true name and the good name, or the great name and the little name; and while the good or little name was made public, the true or great name appears to have been carefully concealed.

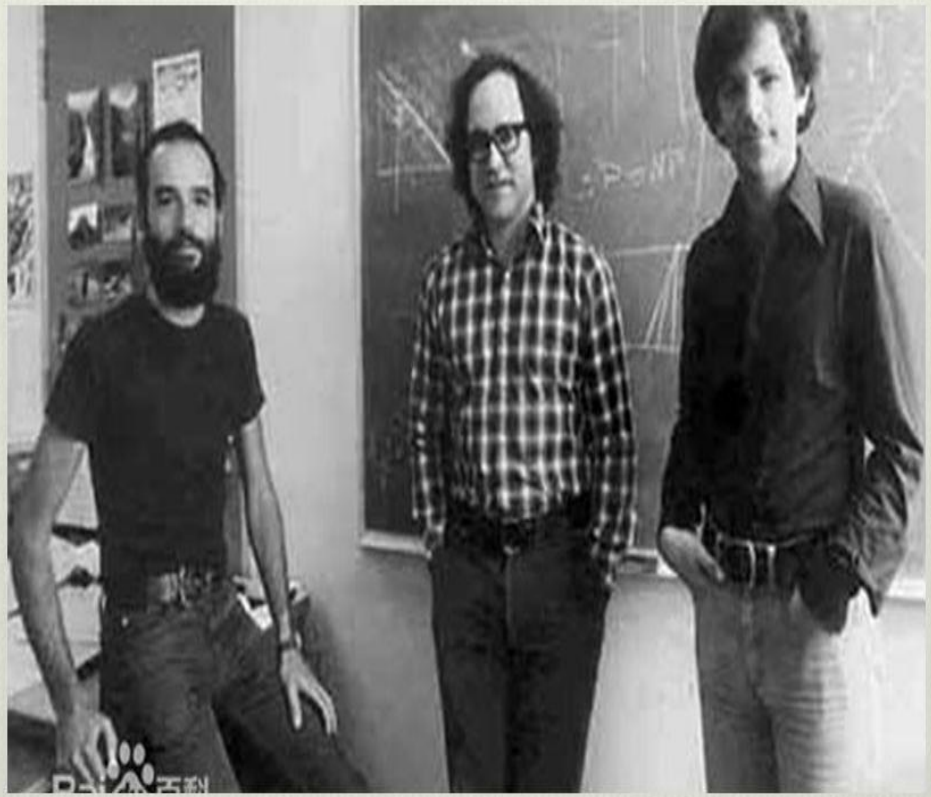
—*The Golden Bough, Sir James George Frazer*

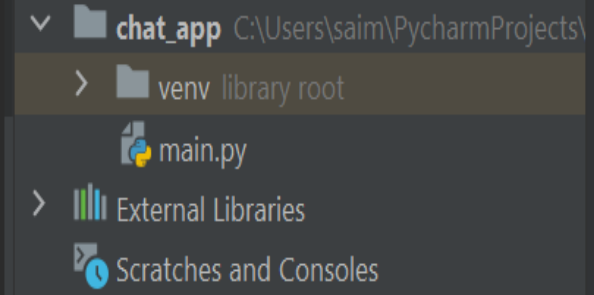
RSA ALGORITHM



1. The pioneering paper by Diffie and Hellman introduced a new approach to cryptography and, in effect, challenged cryptologists to come up with a cryptographic algorithm that met the requirements for public-key systems. A number of algorithms have been proposed for public-key cryptography. Some of these, though initially promising, turned out to be breakable.
2. Apparently, the first workable public-key system for encryption/decryption was put forward by Clifford Cocks of Britain's CESG in 1973 ; Cocks' method is virtually identical to RSA
3. But,the first successful responses to the challenge was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978 ,The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.

Ron Rivest, Adi Shamir and Leonard Adleman





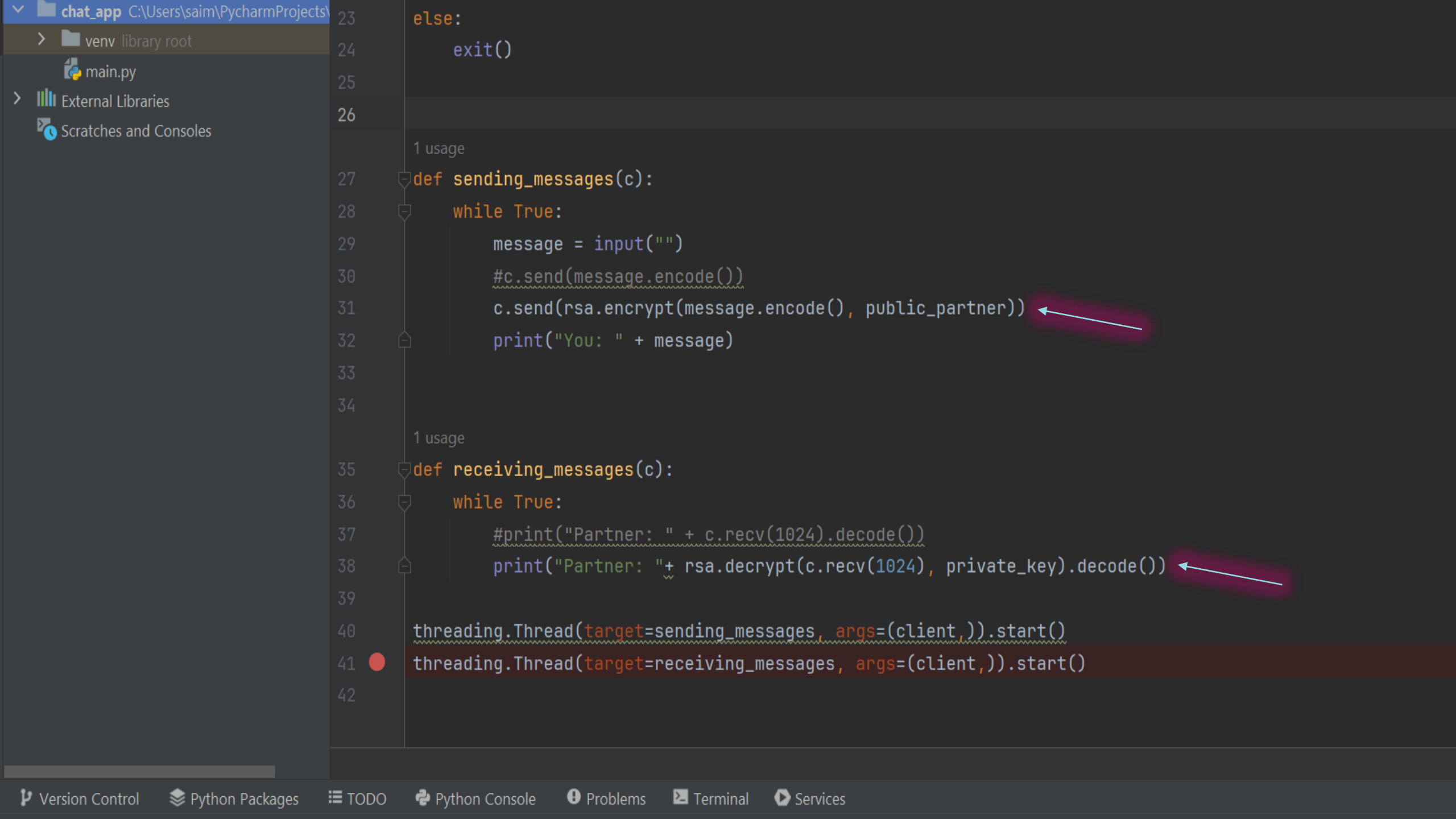

```python
import socket
import threading
import rsa

public_key, private_key = rsa.newkeys(1024)
public_partner = None

choice = input('Do you want to host (1) or to connect (2):')

if choice == '1':
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('192.168.237.250', 9999))
    server.listen()

    client, _ = server.accept()
    client.send(public_key.save_pkcs1('PEM'))
    public_partner = rsa.PublicKey.load_pkcs1(client.recv(1024))
elif choice == '2':
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(('192.168.237.250', 9999))
    public_partner = rsa.PublicKey.load_pkcs1(client.recv(1024))
    client.send(public_key.save_pkcs1('PEM'))
```


chat_app C:\Users\saim\PycharmProjects\
venv library root
main.py
External Libraries
Scratches and Consoles
else:
exit()
1 usage
def sending_messages(c):
while True:
message = input("")
#c.send(message.encode())
c.send(rsa.encrypt(message.encode(), public_partner))
print("You: " + message)
1 usage
def receiving_messages(c):
while True:
#print("Partner: " + c.recv(1024).decode())
print("Partner: "+ rsa.decrypt(c.recv(1024), private_key).decode())
threading.Thread(target=sending_messages, args=(client,)).start()
threading.Thread(target=receiving_messages, args=(client,)).start()
Version Control
Python Packages
TODO
Python Console
Problems
Terminal
Services

```
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saim>"C:\Users\saim\PycharmProjects\chat_app\main.py"
Do you want to host (1) or to connect (2):1
calling....,calling .... ALPHA one!,can you hear me?, we had been attacked b
y enemy forces, we have to access there satellite!, fast tell the security p
assword to open satellite communication! URGENT ORDER
You: calling....,calling .... ALPHA one!,can you hear me?, we had been attac
ked by enemy forces, we have to access there satellite!, fast tell the secur
ity password to open satellite communication! URGENT ORDER
Partner: HEaring alpha, bravo this side, we got your msg!, I can provide key
 but i suspect there may be an security breach, over
oops! I think I forgot to enable RSA
You: oops! I think I forgot to enable RSA
Partner: OMG! please fast enable it!
sure
You: sure
over
You: over
Partner: over
```

```
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saim>"C:\Users\saim\PycharmProjects\chat_app\main.py"
Do you want to host (1) or to connect (2):2
Partner: calling....,calling .... ALPHA one!,can you hear me?, we had been a
ttacked by enemy forces, we have to access there satellite!, fast tell the s
ecurity password to open satellite communication! URGENT ORDER
HEaring alpha, bravo this side, we got your msg!, I can provide key but i su
spect there may be an security breach, over
You: HEaring alpha, bravo this side, we got your msg!, I can provide key but
 i suspect there may be an security breach, over
Partner: oops! I think I forgot to enable RSA
OMG! please fast enable it!
You: OMG! please fast enable it!
Partner: sure
Partner: over
over
You: over
```

No.	Time	Source	Destination	Protocol	Length	Info
1086	381.662408	192.168.1.4	224.0.0.252	IGMPv2	36	Membership Report group 224.0.0.252
1087	381.662481	192.168.1.4	239.255.255.250	IGMPv2	36	Membership Report group 239.255.255.250
1088	381.662548	192.168.1.4	224.0.0.113	IGMPv2	36	Membership Report group 224.0.0.113
1089	381.662632	fe80::140a:f51c:13a…	ff02::c	ICMPv6	76	Multicast Listener Report
1090	381.662729	fe80::140a:f51c:13a…	ff02::fb	ICMPv6	76	Multicast Listener Report
1091	381.662801	fe80::140a:f51c:13a…	ff02::13a	ICMPv6	76	Multicast Listener Report
1092	381.662859	fe80::140a:f51c:13a…	ff02::1:3	ICMPv6	76	Multicast Listener Report
1093	381.662939	fe80::140a:f51c:13a…	ff02::1:ffa7:5ddc	ICMPv6	76	Multicast Listener Report
1094	391.673933	192.168.1.4	192.168.1.4	TCP	249	9999 → 58243 [PSH, ACK] Seq=252 Ack=252 Win=2160896 Len=205
1095	391.673984	192.168.1.4	192.168.1.4	TCP	44	58243 → 9999 [ACK] Seq=252 Ack=457 Win=326912 Len=0
1096	400.053394	127.0.0.1	127.0.0.1	WebSoc…	53	WebSocket Text [FIN] [MASKED]
1097	400.053433	127.0.0.1	127.0.0.1	TCP	44	52746 → 58160 [ACK] Seq=229 Ack=800 Win=2160384 Len=0
1098	400.054529	127.0.0.1	127.0.0.1	WebSoc…	47	WebSocket Text [FIN]
1099	400.054552	127.0.0.1	127.0.0.1	TCP	44	58160 → 52746 [ACK] Seq=800 Ack=232 Win=2160896 Len=0
1100	425.061225	127.0.0.1	127.0.0.1	WebSoc…	53	WebSocket Text [FIN] [MASKED]
1101	425.061261	127.0.0.1	127.0.0.1	TCP	44	52746 → 58160 [ACK] Seq=232 Ack=809 Win=2160384 Len=0
1102	425.068292	127.0.0.1	127.0.0.1	WebSoc…	47	WebSocket Text [FIN]
1103	425.068319	127.0.0.1	127.0.0.1	TCP	44	58160 → 52746 [ACK] Seq=809 Ack=235 Win=2160896 Len=0

```
> Frame 1094: 249 bytes on wire (1992 bits), 249 bytes captured (1992 bits) on interface \Device\NPF_Loop
> Null/Loopback
> Internet Protocol Version 4, Src: 192.168.1.4, Dst: 192.168.1.4
> Transmission Control Protocol, Src Port: 9999, Dst Port: 58243, Seq: 252, Ack: 252, Len: 205
> Data (205 bytes)
```

```
0000  02 00 00 00 45 00 00 f5  11 4d 40 00 80 06 00 00   ····E··· ·M@·····
0010  c0 a8 01 04 c0 a8 01 04  27 0f e3 83 5d 41 e7 dc   ········ '···]A··
0020  17 46 cf c6 50 18 20 f9  0c a4 00 00 63 61 6c 6c   ·F··P· · ····call
0030  69 6e 67 2e 2e 2e 2e 2c  63 61 6c 6c 69 6e 67 20   ing...., calling
0040  2e 2e 2e 2e 20 41 4c 50  48 41 20 6f 6e 65 21 2c   .... ALP HA one!,
0050  63 61 6e 20 79 6f 75 20  68 65 61 72 20 6d 65 3f   can you  hear me?
0060  2c 20 77 65 20 68 61 64  20 62 65 65 6e 20 61 74   , we had  been at
0070  74 61 63 6b 65 64 20 62  79 20 65 6e 65 6d 79 20   tacked b y enemy
0080  66 6f 72 63 65 73 2c 20  77 65 20 68 61 76 65 20   forces,  we have
0090  74 6f 20 61 63 63 65 73  73 20 74 68 65 72 65 20   to acces s there
00a0  73 61 74 65 6c 6c 69 74  65 21 2c 20 66 61 73 74   satellit e!, fast
00b0  20 74 65 6c 6c 20 74 68  65 20 73 65 63 75 72 69    tell th e securi
00c0  74 79 20 70 61 73 73 77  6f 72 64 20 74 6f 20 6f   ty passw ord to o
00d0  70 65 6e 20 73 61 74 65  6c 6c 69 74 65 20 63 6f   pen sate llite co
00e0  6d 6d 75 6e 69 63 61 74  69 6f 6e 21 20 55 52 47   mmunicat ion! URG
00f0  45 4e 54 20 4f 52 44 45  52                        ENT ORDE R
```

Before RSA

```
C:\Users\saim>"C:\Users\saim\PycharmProjects\chat_app\main.py"
Do you want to host (1) or to connect (2):1
now you can deliver the security pwd for satellite communication , now there
 is no security threat!
You: now you can deliver the security pwd for satellite communication , now
there is no security threat!
over
You: over
Partner: ok sir, over
Partner: here is the security pwd : 9564sShubham2004!!
thanks bravo. over
You: thanks bravo. over
Partner: over sir
```

```
KeyboardInterrupt.

C:\Users\saim>"C:\Users\saim\PycharmProjects\chat_app\main.py"
Do you want to host (1) or to connect (2):2
Partner: now you can deliver the security pwd for satellite communication ,
now there is no security threat!
Partner: over
ok sir, over
You: ok sir, over
here is the security pwd : 9564sShubham2004!!
You: here is the security pwd : 9564sShubham2004!!
Partner: thanks bravo. over
over sir
You: over sir
```

No.	Time	Source	Destination	Protocol	Length	Info
1193	650.154114	127.0.0.1	127.0.0.1	TCP	44	58160 → 52746 [ACK] Seq=890 Ack=262 Win=2160896 Len=0
1194	669.789453	192.168.1.4	192.168.1.4	TCP	56	58259 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
1195	669.789524	192.168.1.4	192.168.1.4	TCP	56	9999 → 58259 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
1196	669.789562	192.168.1.4	192.168.1.4	TCP	44	58259 → 9999 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
1197	669.824450	192.168.1.4	192.168.1.4	TCP	295	9999 → 58259 [PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=251
1198	669.824502	192.168.1.4	192.168.1.4	TCP	44	58259 → 9999 [ACK] Seq=1 Ack=252 Win=2160896 Len=0
1199	669.863320	192.168.1.4	192.168.1.4	TCP	295	58259 → 9999 [PSH, ACK] Seq=1 Ack=252 Win=2160896 Len=251
1200	669.863354	192.168.1.4	192.168.1.4	TCP	44	9999 → 58259 [ACK] Seq=252 Ack=252 Win=2160896 Len=0
1201	675.163299	127.0.0.1	127.0.0.1	WebSoc…	53	WebSocket Text [FIN] [MASKED]
1202	675.163349	127.0.0.1	127.0.0.1	TCP	44	52746 → 58160 [ACK] Seq=262 Ack=899 Win=2160384 Len=0
1203	675.164205	127.0.0.1	127.0.0.1	WebSoc…	47	WebSocket Text [FIN]
1204	675.164233	127.0.0.1	127.0.0.1	TCP	44	58160 → 52746 [ACK] Seq=899 Ack=265 Win=2160896 Len=0
1205	700.169106	127.0.0.1	127.0.0.1	WebSoc…	53	WebSocket Text [FIN] [MASKED]
1206	700.169144	127.0.0.1	127.0.0.1	TCP	44	52746 → 58160 [ACK] Seq=265 Ack=908 Win=2160384 Len=0
1207	700.169865	127.0.0.1	127.0.0.1	WebSoc…	47	WebSocket Text [FIN]
1208	700.169888	127.0.0.1	127.0.0.1	TCP	44	58160 → 52746 [ACK] Seq=908 Ack=268 Win=2160896 Len=0
1209	709.755944	192.168.1.4	192.168.1.4	TCP	172	9999 → 58259 [PSH, ACK] Seq=252 Ack=252 Win=2160896 Len=128
1210	709.755985	192.168.1.4	192.168.1.4	TCP	44	58259 → 9999 [ACK] Seq=252 Ack=380 Win=2160896 Len=0

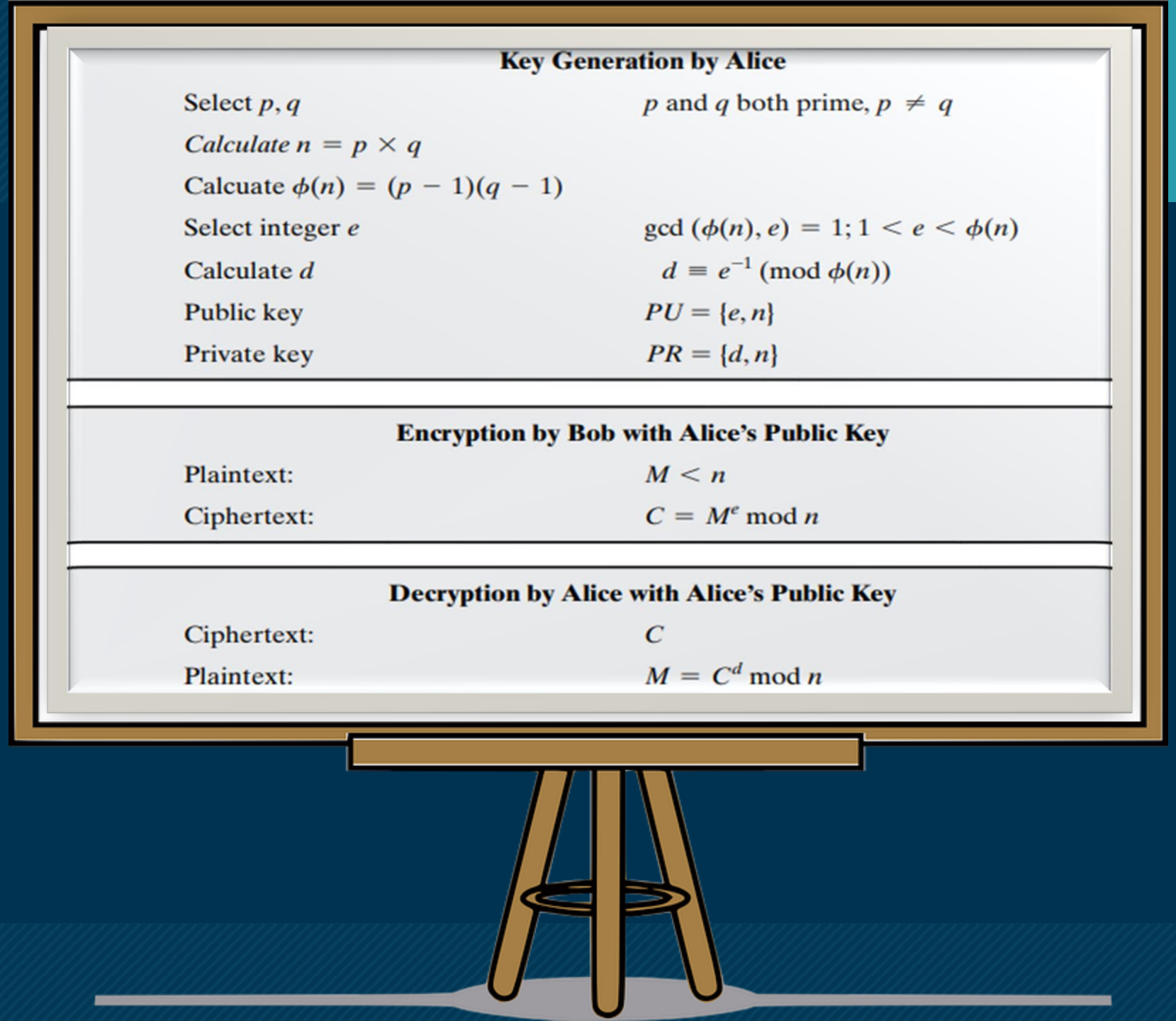
```
> Frame 1209: 172 bytes on wire (1376 bits), 172 bytes captured (1376 bits) on interface \Device\NPF_Loop
> Null/Loopback
> Internet Protocol Version 4, Src: 192.168.1.4, Dst: 192.168.1.4
> Transmission Control Protocol, Src Port: 9999, Dst Port: 58259, Seq: 252, Ack: 252, Len: 128
> Data (128 bytes)
```

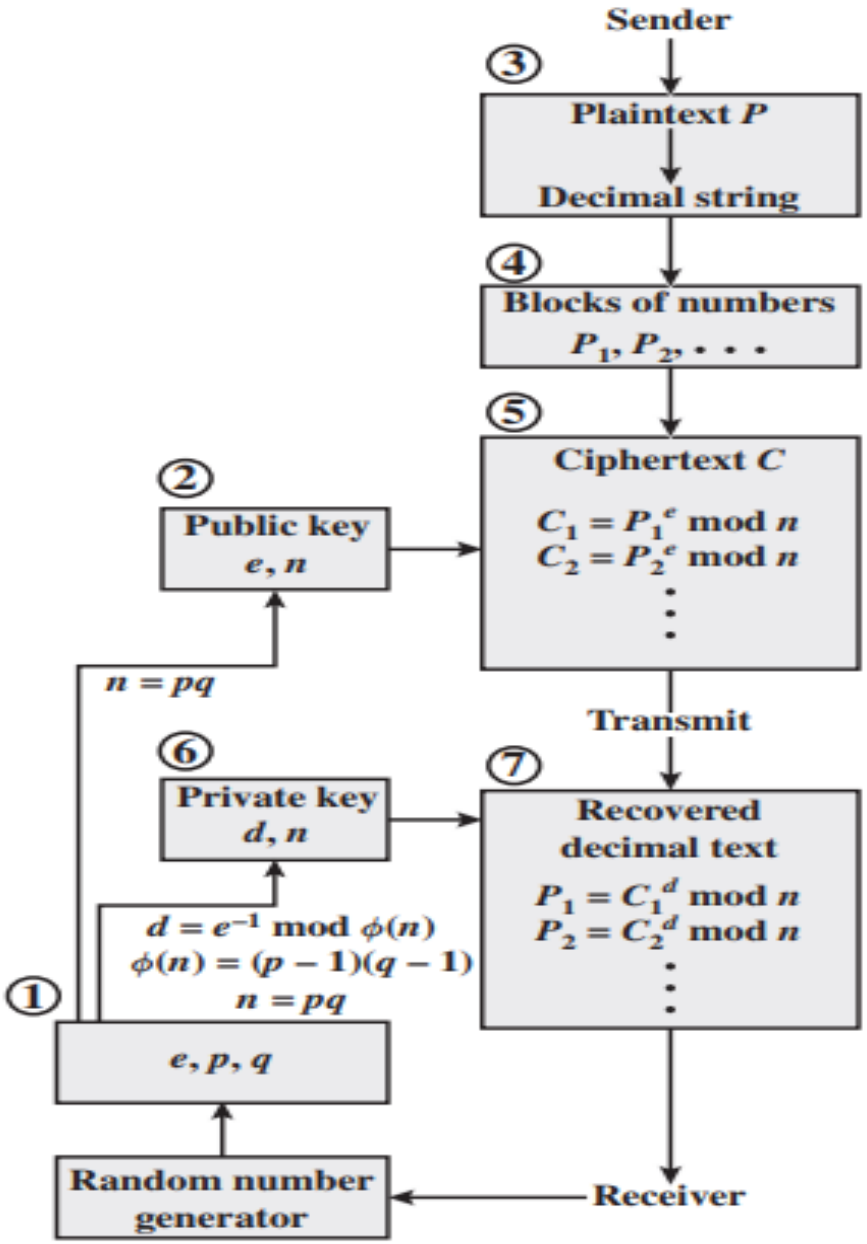
After RSA

```
0000  02 00 00 00 45 00 00 a8  11 63 40 00 80 06 00 00   ····E···  ·c@·····
0010  c0 a8 01 04 c0 a8 01 04  27 0f e3 93 3e 5a df 9e   ········  '···>Z··
0020  99 a5 25 c0 50 18 20 f9  22 87 00 00 16 41 37 9a   ··%·P· ·  "····A7·
0030  34 c2 e0 23 aa 21 31 90  d4 f3 4f f0 2b 41 f0 39   4··#·!1·  ··O·+A·9
0040  66 6d 90 0a 2a bc 48 62  aa 83 be 6d 54 32 32 f8   fm··*·Hb  ···mT22·
0050  75 81 be d1 12 c0 e8 ba  b6 a5 21 b8 9b 14 58 80   u·······  ··!···X·
0060  99 5f 80 d5 72 97 dc 10  99 d1 df a0 71 df ff fa   ·_··r···  ····q···
0070  26 21 d8 54 21 36 3a 1b  03 e3 eb f0 ec 14 45 9d   &!·T!6:·  ······E·
0080  fc a4 f1 22 2f 1c 6a 5f  96 5e 0d 9a f1 80 e3 b6   ···"/·j_  ·^······
0090  dd 7b d9 8a e9 71 d7 32  91 26 c0 68 60 79 fe 7e   ·{···q·2  ·&·h`y·~
00a0  15 a5 8e 6f f5 91 6c e5  2f 64 fe 1c               ···o··l·  /d··
```

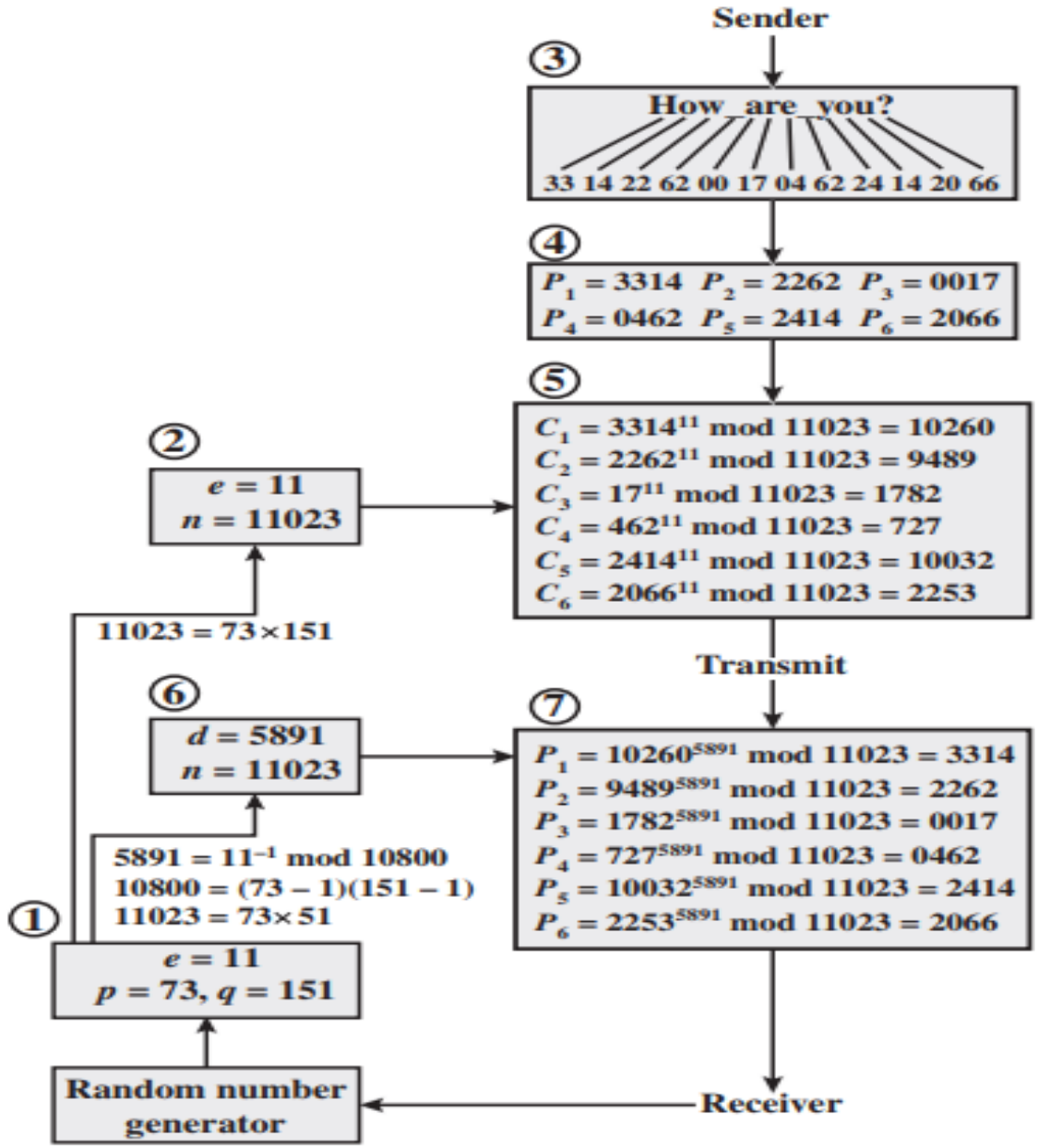
1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$; d can be calculated using the extended Euclid's algorithm

Char	Int	Char	Int	Char	Int	Char	Int
a	1	b	2	c	3	d	4
e	5	f	6	g	7	h	8
i	9	j	10	k	11	l	12
m	13	n	14	o	15	p	16
q	17	r	18	s	19	t	20
u	21	v	22	w	23	x	24
y	25	z	26	A	27	B	28
C	29	D	30	E	31	F	32
G	33	H	34	I	35	J	36
K	37	L	38	M	39	N	40
O	41	P	42	Q	43	R	44
S	45	T	46	U	47	V	48
W	49	X	50	Y	51	Z	52
0	53	1	54	2	55	3	56
4	57	5	58	6	59	7	60
8	61	9	62	:	63	-	64


Key Generation by Alice
Select p, q
p and q both prime, $p \neq q$
Calculate $n = p \times q$
Calcuate $\phi(n) = (p - 1)(q - 1)$
Select integer e
$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d
$d \equiv e^{-1} \pmod{\phi(n)}$
Public key
$PU = \{e, n\}$
Private key
$PR = \{d, n\}$
Encryption by Bob with Alice's Public Key
Plaintext:
$M < n$
Ciphertext:
$C = M^e \bmod n$
Decryption by Alice with Alice's Public Key
Ciphertext:
C
Plaintext:
$M = C^d \bmod n$



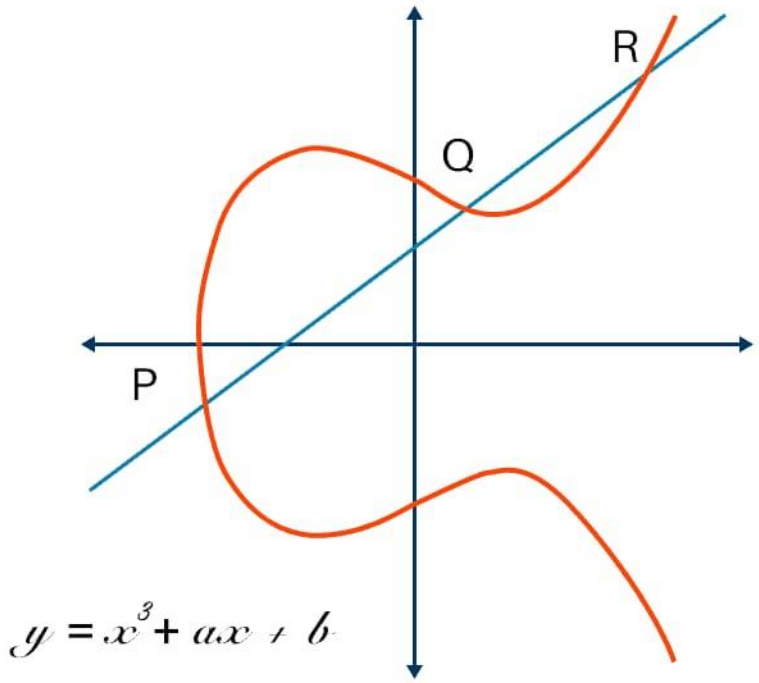
(a) General approach



(b) Example

Elliptic Curve Cryptography

- The principal attraction of ECC, compared to RSA, is that it appears to offer equal security for a far smaller key size, thereby reducing processing overhead.



Analog of Diffie–Hellman Key Exchange

Key exchange using elliptic curves can be done in the following manner. First pick a large integer q, which is either a prime number p or an integer of the form 2^m, and elliptic curve parameters a and b for Equation (10.5) or Equation (10.7). This defines the elliptic group of points $E_q(a, b)$. Next, pick a *base point* $G = (x_1, y_1)$ in $E_p(a, b)$ whose order is a very large value n. The **order** n of a point G on an elliptic curve is the smallest positive integer n such that $nG = 0$ and G are parameters of the cryptosystem known to all participants.

A key exchange between users A and B can be accomplished as follows (Figure 10.7).

1. A selects an integer n_A less than n. This is A's private key. A then generates a public key $P_A = n_A \times G$; the public key is a point in $E_q(a, b)$.
2. B similarly selects a private key n_B and computes a public key P_B.
3. A generates the secret key $k = n_A \times P_B$. B generates the secret key $k = n_B \times P_A$.

The two calculations in step 3 produce the same result because

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

It is the point Pm that will be encrypted as a ciphertext and subsequently decrypted.

As with the key exchange system, an encryption/decryption system requires a point G and an elliptic group $E_q(a, b)$ as parameters. Each user A selects a private key n_A and generates a public key $P_A = n_A \times G$.

To encrypt and send a message P_m to B, A chooses a random positive integer k and produces the ciphertext C_m consisting of the pair of points:

$$C_m = \{kG, P_m + kP_B\}$$

Note that A has used B's public key P_B. To decrypt the ciphertext, B multiplies the first point in the pair by B's private key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$$

Table 10.3 Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis (NIST SP-800-57)

Symmetric Key Algorithms	Diffie–Hellman, Digital Signature Algorithm	RSA (size of n in bits)	ECC (modulus size in bits)
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15{,}360$ $N = 512$	15,360	512+

Note: L = size of public key, N = size of private key.

- The first task in this system is to encode the plaintext message m to be sent as an (x, y) point Pm

Global Public Elements

$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation

Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A

$K = n_A \times P_B$

Calculation of Secret Key by User B

$K = n_B \times P_A$

Figure 10.7 ECC Diffie–Hellman Key Exchange

Types Of Attack On Public Key Cryptography

Ciphertext - Plaintext

Ciphertext:

2D570755676DFF11E71B6C8511EFE7A7D3B02A3CEE63165050AB5
F4C4D19A4AABO7656A636654C6F39A4ACOFEA2035CCDD7181CO
EBB482A6EBDAEF2AEB35CB5C325CBF0738AEC27D77BEC3938C
590CE77F62CBDCC3EA3D03E06A386BD70BC99A843DD6B7B975
3635C919FA17FC40A3C3DCBD13633D2D56A1A073EA0E73E60C60

Plaintext:

HELLO WORLD

Algorithm:

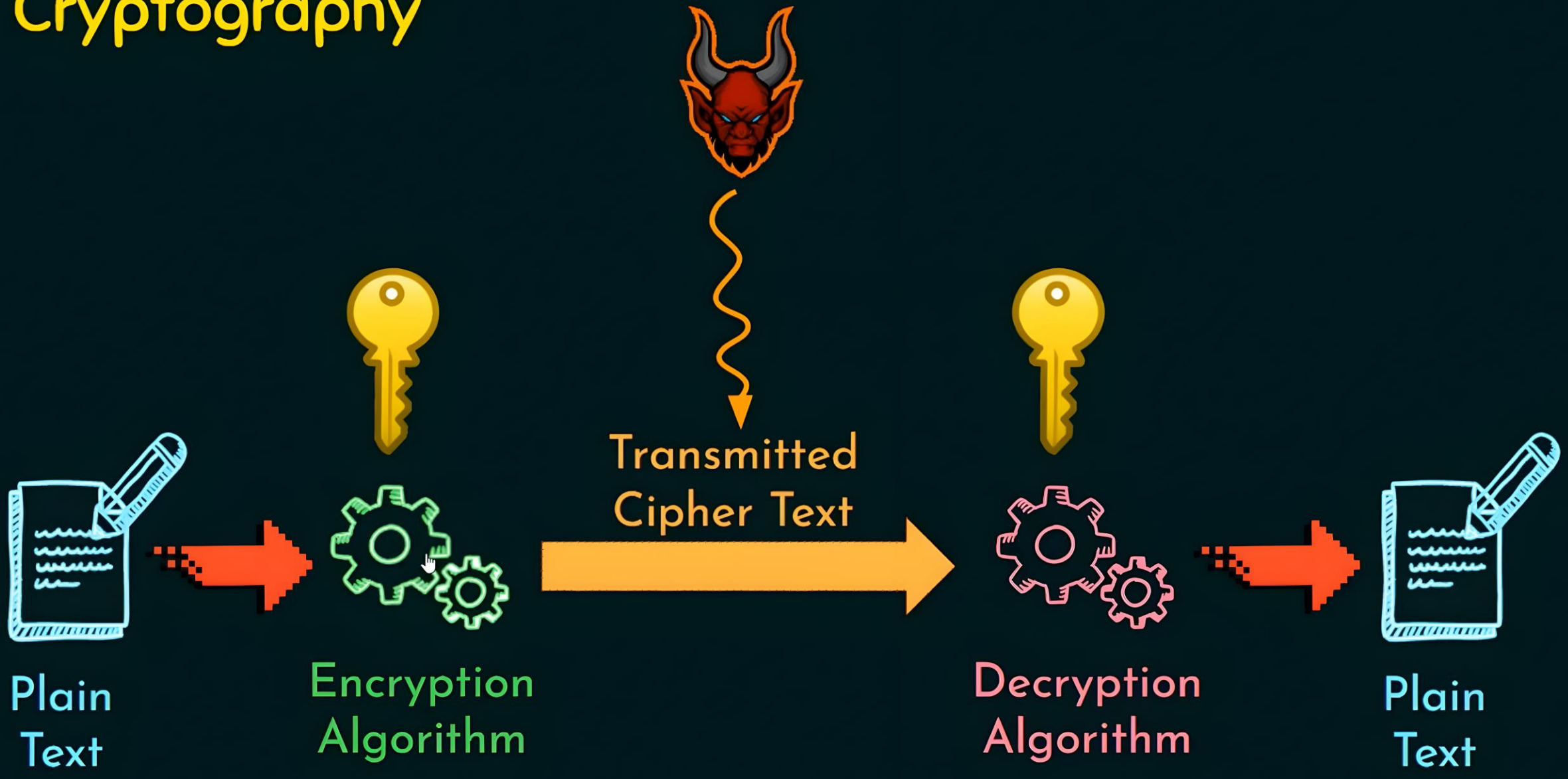
RSA Algorithm

Cryptanalysis

- Cryptanalytic attacks - Based on info know to the cryptanalyst.
- Most difficult : Ciphertext only (Not even encryption algorithm)
- Types of cryptanalytic attacks:
  - Ciphertext Only
  - Known Plaintext
  - Chosen Plaintext
  - Chosen Ciphertext
  - Chosen Text

Type of Attack	Known to cryptanalyst
Ciphertext Only	★ Encryption Algorithm ★ Ciphertext
Known Plaintext	★ Encryption Algorithm ★ Ciphertext ★ One or more PT-CT pairs formed with secret key
Chosen Plaintext	★ Encryption Algorithm ★ Ciphertext ★ PT message chosen by cryptanalyst, together with its CT generated with the secret key
Chosen Ciphertext	★ Encryption Algorithm ★ Ciphertext ★ CT chosen by cryptanalyst, together with its corresponding decrypted PT generated with the secret key
Chosen Text	★ Chosen Plaintext and Chosen Ciphertext

Cryptography



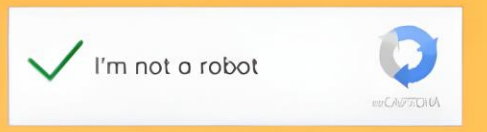
Brute–force attack

- ★ Trying every possible key.
- ★ Until an intelligible translation of the ciphertext into plaintext is obtained.
- ★ Guessing.
- ★ Exhaustive key search.
- ★ Software Tools that can perform brute-force attack.

Aircrack-ng	DaveGrohl	John the ripper
Cain and Abel	Hashcat	Rainbowcrack
Crack	Hydra	Ophcrack

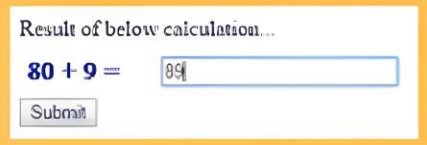
CAPTCHA

Text-based Captcha



ReCAPTCHA

3D Captcha



Mathematical Captcha

Image-based Captcha

A contrived acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart"

Brute-force attack

```
[80][http-get-form] host: 192.168.100.155   login: admin   password: password
[80][http-get-form] host: 192.168.100.155   login: admin   password: p@ssword
[80][http-get-form] host: 192.168.100.155   login: admin   password: 12345
[80][http-get-form] host: 192.168.100.155   login: admin   password: 1234567890
[80][http-get-form] host: 192.168.100.155   login: admin   password: Password
[80][http-get-form] host: 192.168.100.155   login: admin   password: 123456
[80][http-get-form] host: 192.168.100.155   login: admin   password: 1234567
[80][http-get-form] host: 192.168.100.155   login: admin   password: 12345678
[80][http-get-form] host: 192.168.100.155   login: admin   password: 1q2w3e4r
[80][http-get-form] host: 192.168.100.155   login: admin   password: 123
[80][http-get-form] host: 192.168.100.155   login: admin   password: 1
[80][http-get-form] host: 192.168.100.155   login: admin   password: 12
1 of 1 target successfully completed, 12 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-07-27 15:28:24
```

Brute-force attack

```
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\DefaultAccount :stealth1agent',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\DefaultAccount :Q4)sJu\Y8qz*A3?d',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\DefaultAccount :$uperP@ssword',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\WDAGUtilityAccount :stealth1agent',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\WDAGUtilityAccount :Q4)sJu\Y8qz*A3?d',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\WDAGUtilityAccount :$uperP@ssword',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\None :stealth1agent',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\None :Q4)sJu\Y8qz*A3?d',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\None :$uperP@ssword',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\Hazard :stealth1agent',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\Hazard :Q4)sJu\Y8qz*A3?d',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\Hazard :$uperP@ssword',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\support :stealth1agent',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\support :Q4)sJu\Y8qz*A3?d',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\support :$uperP@ssword',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\Chase:stealth1agent',
[+] 10.10.10.149:445      - 10.10.10.149:445 - Success: '.\Chase:Q4)sJu\Y8qz*A3?d'
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\Jason:stealth1agent',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\Jason:Q4)sJu\Y8qz*A3?d',
[-] 10.10.10.149:445      - 10.10.10.149:445 - Failed: '.\Jason:$uperP@ssword',
[*] 10.10.10.149:445      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Brute-force attack

```
$ /usr/sbin/john --wordlist=/usr/share/wordlists/rockyou.txt ~/passwords.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 XOP 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
password         (root)
1g 0:00:00:15 0.10% (ETA: 00:14:39) 0.06410g/s 1050p/s 1054c/s 1054C/s christal..spiderpig
1g 0:00:00:16 0.10% (ETA: 00:15:41) 0.06020g/s 1048p/s 1051c/s 1051C/s paramedic..eminem12
1g 0:00:00:34 0.21% (ETA: 00:20:41) 0.02886g/s 1053p/s 1054c/s 1054C/s 090506..zuniga
```

Eve has intercepted the ciphertext "UVACLYFZLJBYL". Show how she can use a brute-force attack to break the cipher.

Solution

Eve tries keys from 1 to 7. With a key of 7, the plaintext is "not very secure", which makes sense.

Ciphertext: UVACLYFZLJBYL

K = 1	→	**Plaintext:** tuzbkxeykiaxk
K = 2	→	**Plaintext:** styajwdxjhzwj
K = 3	→	**Plaintext:** rsxzivcwigyvi
K = 4	→	**Plaintext:** qrwyhubvhfxuh
K = 5	→	**Plaintext:** pqvxgtaugewtg
K = 6	→	**Plaintext:** opuwfsztfdvsf
K = 7	→	**Plaintext:** notverysecure

A CRYPTO NERD'S IMAGINATION:

HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.

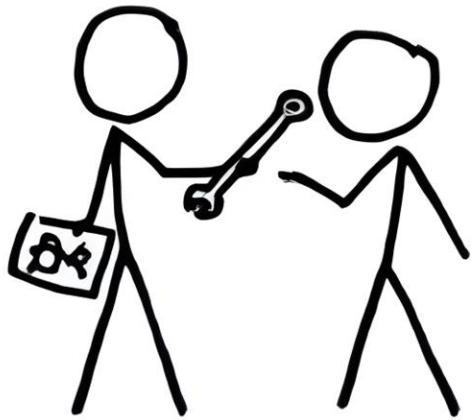
NO GOOD! IT'S 4096-BIT RSA!

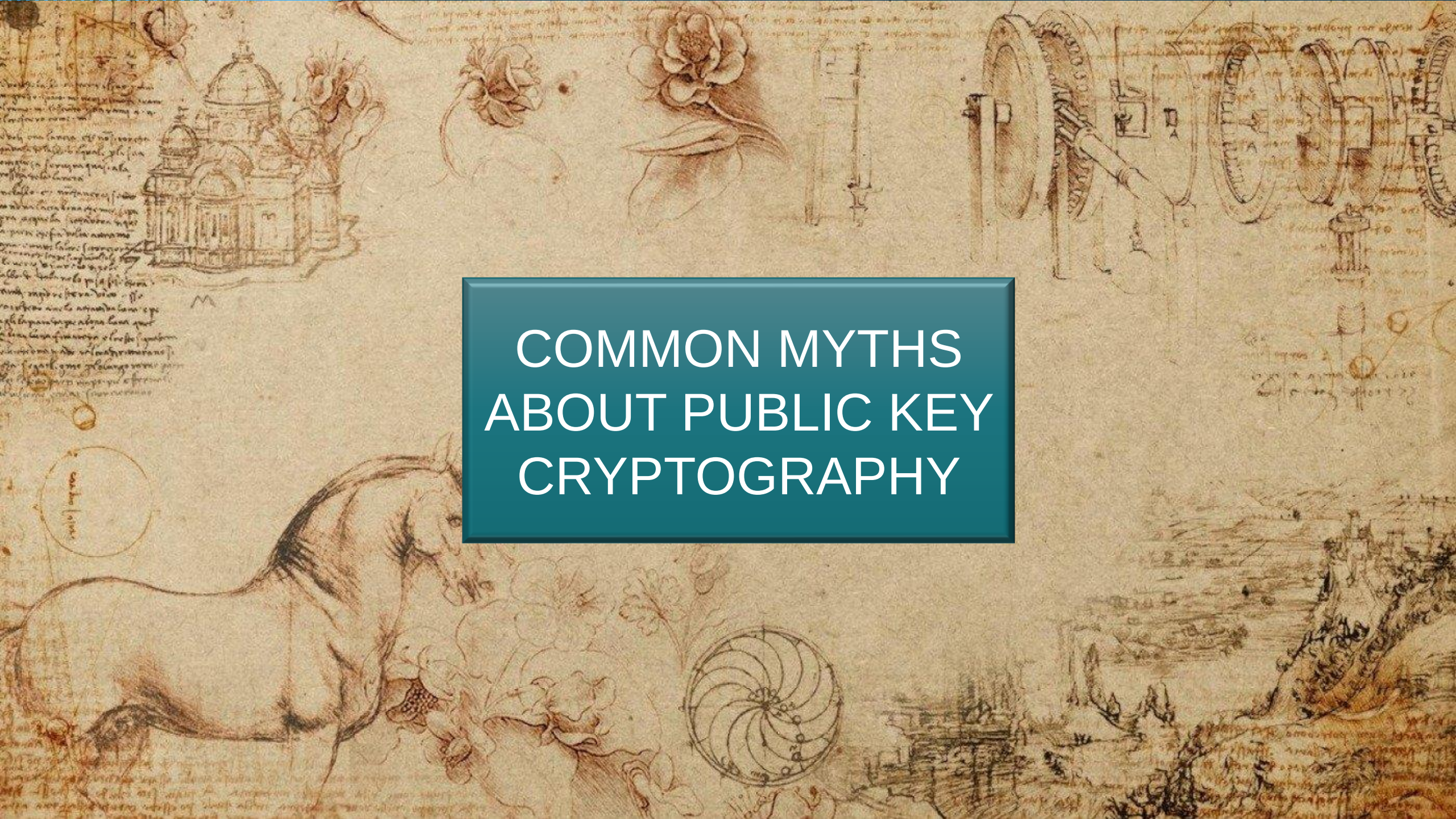
BLAST! OUR EVIL PLAN IS FOILED!

WHAT WOULD ACTUALLY HAPPEN:

HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.

GOT IT.




COMMON MYTHS
ABOUT PUBLIC KEY
CRYPTOGRAPHY

Myth 1: Public key encryption is unbreakable.

While PKE is considered a very secure cryptographic technique, it is not entirely unbreakable. Advances in computing power and algorithmic techniques could potentially lead to the development of methods to break PKE algorithms. However, such breakthroughs are not imminent, and PKE remains a robust and reliable means of protecting sensitive data.

Myth 2: Encryption protects data everywhere.

Encryption, including PKE, is most effective when applied at the endpoints of communication, such as devices and servers. Once data travels through unencrypted channels, such as the internet, it becomes vulnerable to interception and decryption. Therefore, it is crucial to employ encryption consistently throughout the data lifecycle.

Myth 3: Encryption is only for technical experts.

Encryption technology has evolved to become more accessible and user-friendly. Many applications, including email clients, messaging platforms, and file storage services, incorporate encryption seamlessly into their user interfaces.

M

Y

T

H

S

Debunking these myths is essential for promoting a comprehensive understanding of public key encryption and its role in protecting digital information. By recognizing these misconceptions, individuals and organizations can make informed decisions about their security practices and ensure the protection of their sensitive data.

Myth 4: Public key encryption is only used for secure email.

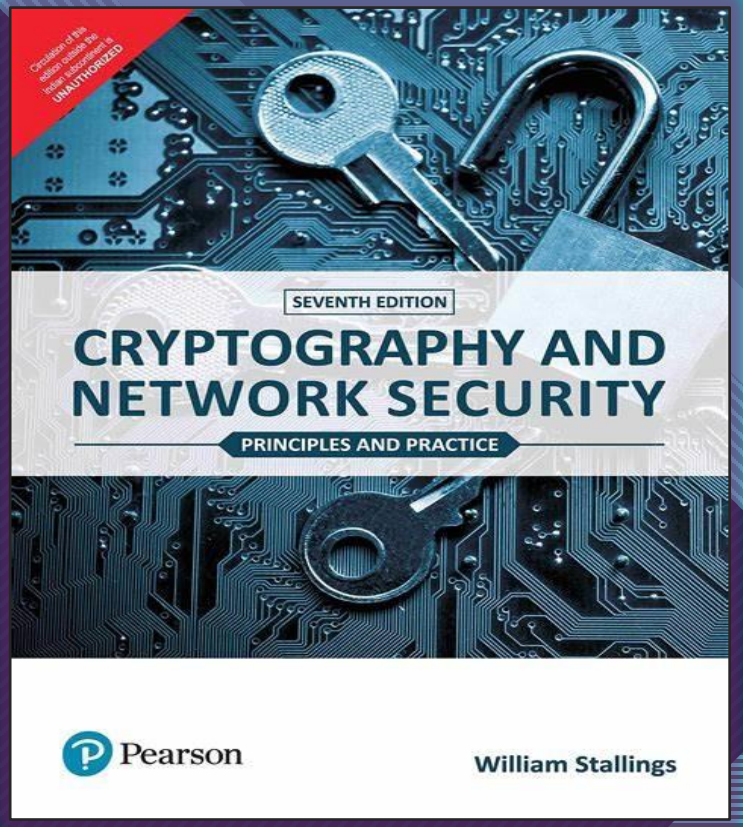
While public key encryption is commonly used for secure email, it is also used for a wide variety of other applications, including:

- Secure file transfers
- Secure web browsing (HTTPS)
- Digital signatures
- VPNs
- Blockchain technology

MYTHS

BIBLIOGRAPHY

1. Cryptography and Network Security, Principles and Practice, 6th ed by William Stallings
2. IBM Report: Average cost of a data breach in India touched INR 179 million in 2023
3. History of Cryptography – GeeksforGeeks
4. Farzi 2023 Thriller Web Series (https://youtu.be/LhwDsjFPtME?si=tCTR00W2y-PTgiM2)
5. https://chat.openai.com/share/71d1e0c7-0a4d-4314-bcc0-47b1daaa6cc8
6. https://chat.openai.com/share/c6da28ef-e289-4372-b08c-83132a15722a
7. https://g.co/bard/share/4ed45f94257f
8. The Alan Turing Cryptography Competition edition 2023 (manchester.ac.uk)
9. Images are taken from Various web sites, and collection stocks of Microsoft.



Thank You !