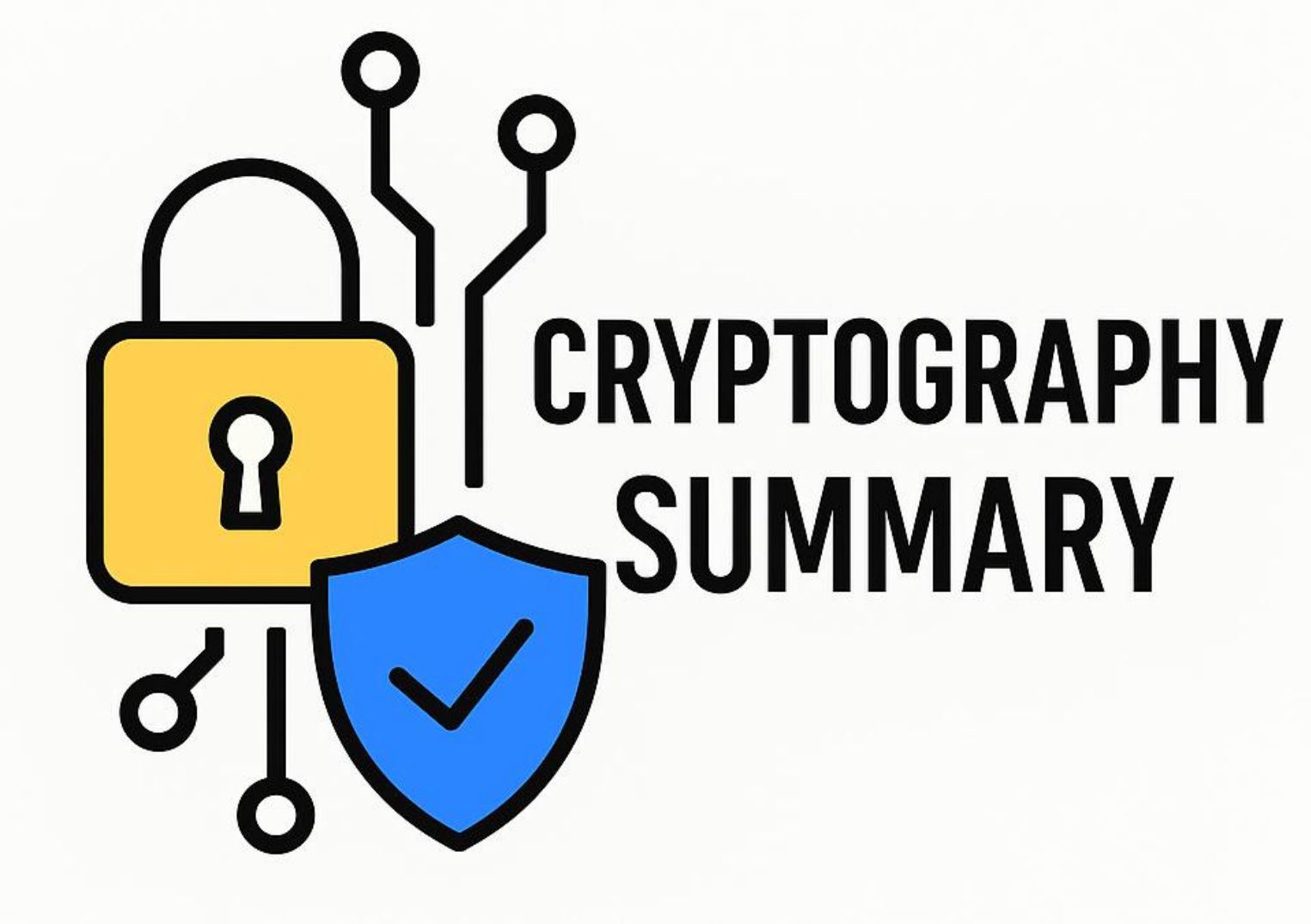


Done By Mohamed Elrimawi

1) Ciphertext-only attack:

the hacker only knows the encrypted message and the algorithm used to encrypt it.
Do they know exactly which algorithm was used for this message?
Usually, yes — companies often use known algorithms, and it's not a secret.
According to Kerkhoff's Principle, the secrecy of the system depends on the key, not the algorithm.
This is the hardest type of attack because the hacker doesn't have any extra information.
They have to try all possible keys.
For example, if the company uses a Caesar cipher, the hacker might try every letter shift.
But if the algorithm uses both letters and symbols, it gets much harder.
The hacker might also try to find weaknesses in the system to get more info.

2) Known-plaintext attack:

In this attack, the hacker has both the plaintext (original message) and the ciphertext (encrypted message).
How did the hacker get both?
Maybe the messages were leaked by mistake, or the hacker used another attack like Man-in-the-Middle to get them.
This attack is easier than ciphertext-only because the hacker has more clues to help break the encryption.

3) Chosen-plaintext attack:

Here, the hacker has access to the encryption system (the system the sender uses to encrypt messages).
The hacker chooses a plaintext, puts it into the encryption system, and gets the ciphertext result.
From this, the hacker can learn how the system works and try to break other encrypted messages.
For example, if the company uses Caesar cipher and the hacker enters "hello" and gets "khoor",
he can guess the system uses a 3-letter shift.
The hacker can also build a dictionary attack with common words and their ciphertexts.
This helps even with strong algorithms.
That's why companies use hash + salt — to make sure the same word doesn't always give the same encrypted result.

4) Chosen-ciphertext attack:

In this attack, the hacker has access to the decryption system (used by the sender, receiver, or another part of the system).
The hacker can enter any ciphertext and get the plaintext result.
This way, the hacker might find weak points in the system or guess the key.

5) Chosen-text attack:

In this attack, the hacker has access to everything from the four previous attacks.
This makes it the strongest and most dangerous type.
The hacker can use all methods together to break the encryption faster.
They have a much better chance of getting the key or reading the messages

Classical cipher algorithms

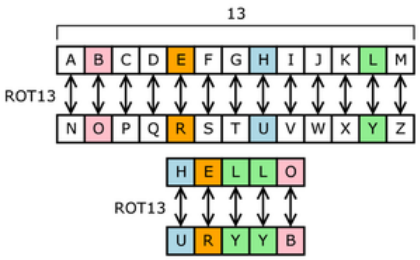
1) Additive Cipher (Shift Cipher / Caesar Cipher):

$E(K, P) = (P + K) \% 26$ (Encryption)

$D(K, C) = (C - K) \% 26$ (Decryption)

2) Monoalphabetic Cipher:

is a type of substitution cipher where each letter in the plaintext is replaced by another letter. One to One



3) Playfair Cipher:

is a way to encrypt messages by pairing up letters and then replacing each pair with a different pair using a special 5x5 grid of letters

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

Encryption: If pairs in the same row move right
If pairs in the same column move down

Decryption: If pairs in the same row move left
If pairs in the same column move up

- If the letters are not in the same row or column, swap them by making a rectangle and picking the other two corners
- If a pair has double letters (like 'AA' or 'OO'), insert an 'X' between them.

4) Hill Cipher :

$$\begin{bmatrix}15 & 17\\20 & 9\end{bmatrix} \times \begin{bmatrix}17\\6\end{bmatrix} = \begin{bmatrix}357\\394\end{bmatrix} \% \ 26 = \begin{bmatrix}19\\4\end{bmatrix} \rightarrow \begin{bmatrix}T\\E\end{bmatrix}$$

$$\begin{bmatrix}15 & 17\\20 & 9\end{bmatrix} \times \begin{bmatrix}22\\11\end{bmatrix} = \begin{bmatrix}517\\539\end{bmatrix} \% \ 26 = \begin{bmatrix}23\\19\end{bmatrix} \rightarrow \begin{bmatrix}X\\T\end{bmatrix}$$

TEXT

$E(K, P) = (K \times P) \% 26$ (Encryption)

$D(K, C) = (K^{-1} \times C) \% 26$ (Decryption)

A 3 x 3 key matrix takes 3 letters at a time.

$$A^{-1} = \frac{1}{|A|} \cdot \text{Adj } A$$

A 2 x 2 key matrix takes 2 letters at a time

5) Polyalphabetic :

Vigenère cipher:

The Vigenère Cipher is an encryption method that uses a repeated keyword to shift each letter in the message, making it stronger than simple ciphers like Caesar. However, it can be broken using the Kasiski Examination, a technique that finds repeating patterns in the ciphertext to guess the keyword length and eventually crack the code.

Key : BYE

H	E	L	L	O	Plain
B	Y	E	B	Y	Key
I	C	P	M	M	Cipher

- Encryption: $C_i = (P_i + K_i) \bmod 26$
- Decryption: $P_i = (C_i - K_i) \bmod 26$

The Auto-key Vigenère Cipher uses the message itself to create a longer key, making it harder to break by avoiding repeated keys

Vernam Cipher:

The Vernam Cipher encrypts a message using a key and the XOR operation, producing a random ciphertext, making it highly secure

Letter h in ASCII	0	1	1	0	1	0	0	0
Letter s in ASCII	0	1	1	1	0	0	1	1
XOR	0	0	0	1	1	0	1	1

- Encryption: $C_i = P_i \text{ XOR } K_i$
- Decryption: $P_i = C_i \text{ XOR } K_i$

One-time Pad:

uses a random key as long as the message. If the key is random, used once, and kept secret, it is unbreakable. However, it's not practical to implement due to the difficulty of securely sharing and storing the key.

Transposition Techniques

Keyless Transposition Ciphers:

encrypt messages by rearranging letters in a fixed pattern without using a secret key. The structure, like depth, is public making them easier to break.

She then creates the ciphertext "MEMATEAKETETHPR".

Keyed Transposition Ciphers

Keyed Transposition Ciphers encrypt messages by placing the plaintext in a grid with columns labeled by a secret key. The columns are then rearranged based on the alphabetical order of the key's letters. This makes the encryption stronger than keyless methods because the key adds a hidden structure that's hard to guess without knowing it

3	1	4	2 ← Permuted column Order
M	E	E	T
T	O	M	O
R	R	O	W

Summary: Stream Cipher & Block Cipher

Stream Ciphers:

- Encrypt data bit by bit or byte by byte.
- Examples: Autokeyed Vigenère, Vernam cipher.
- Very fast and good for real-time use.

Block Ciphers:

- Encrypt data in fixed-size blocks (like 64, 128, or 256 bits).
- Examples: DES , AES, Triple DES, Feistel cipher.
- If a block is not full, extra space (padding) is added.

Summary: Features of a Feistel Cipher

1. **Splitting:** The plaintext block is split into two halves (Left and Right).

2. **Rounds:** Encryption happens in many rounds (usually 16).

3. **Round Function (F):** Each round uses a function F that mixes the right half with a subkey.

4. **Key Mixing**: Every round uses a different subkey, generated from the main key.

5. **Swapping:** After each round, the halves (Left and Right) are swapped.

6. **Same for Encryption and Decryption**: The structure is the same for both, but in decryption the keys are used in reverse order.

7. **Security Principles:**

- Confusion: hides link between key and ciphertext (substitution).
- Diffusion: spreads plaintext changes across ciphertext (Permutation)

Examples: DES

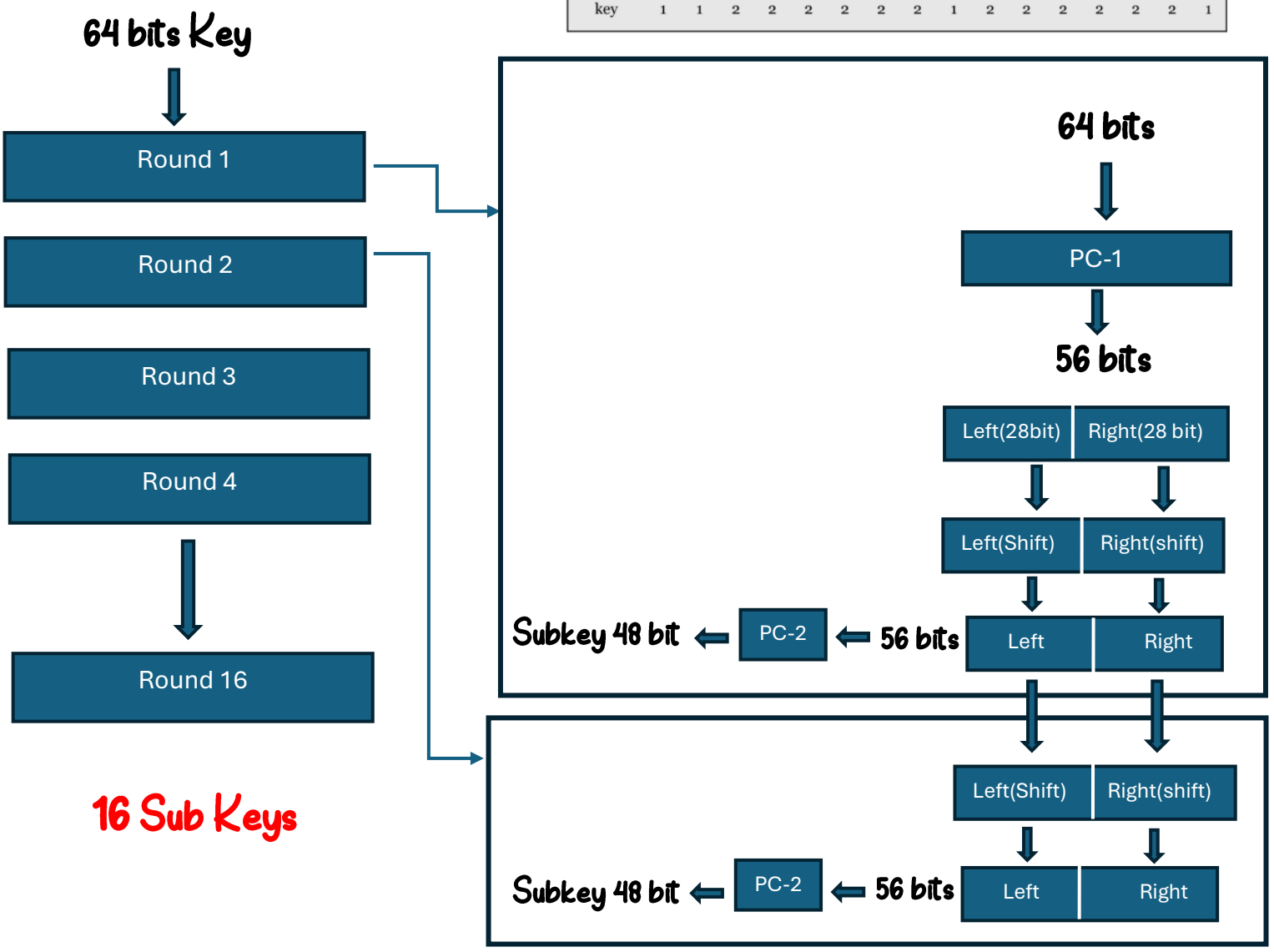
Non-Feistel Example: AES.

Data Encryption Standard (DES)

- Text Size: 64 bits
- Key Size: 64 bits

Key Expansion:

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
key	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1



Encryption:

Key Expansion:

Plaintext 64 bits

Initial Permutation

Round 1

Round 2

Round 3

Round 4

Round 16

Final Permutation

Ciphertext 64 bits

64 bits Key

Round 1

Round 2

Round 3

Round 4

Round 16

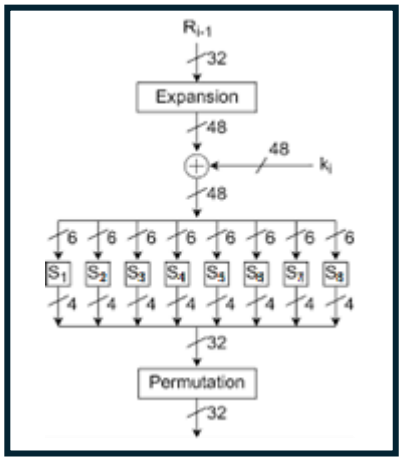
Subkey 48 bit

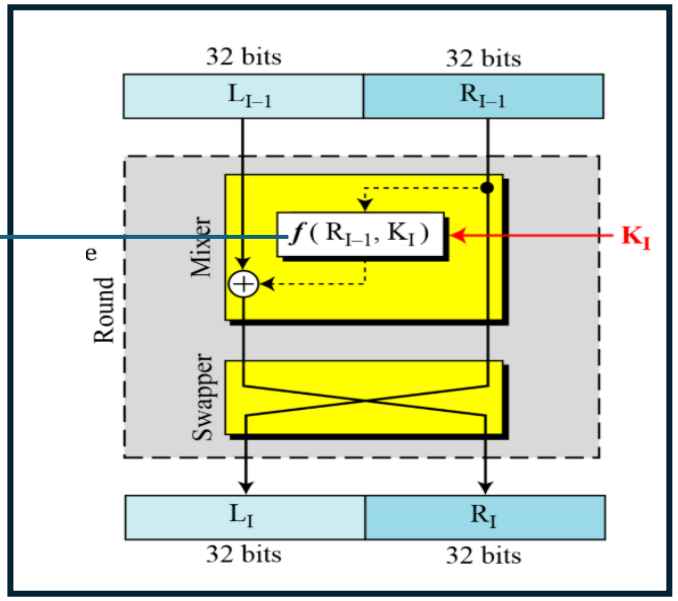
Subkey 48 bit

Subkey 48 bit

Subkey 48 bit

Subkey 48 bit



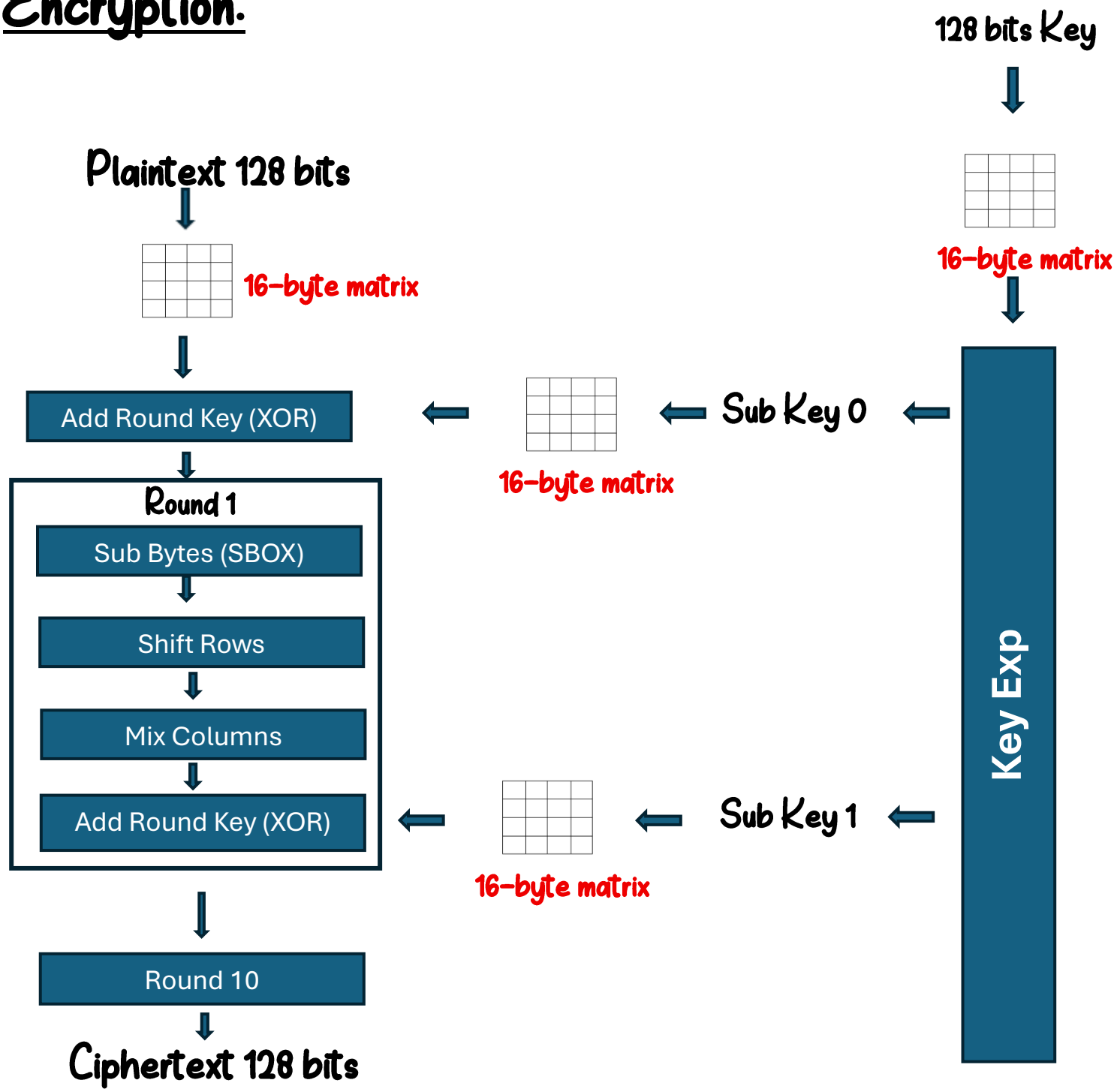


Advanced Encryption Standard (AES)

Key length (bits)	Number of rounds
128	10
192	12
256	14

- Text Size: 128 bits
- Key Size: 128 bits or 192 bits or 256 bits

Encryption:



All rounds include the Mix Columns step except the final round

AddRoundKey: input state⊕ Cipher key

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

⊕

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

=

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

SBox:

Shift Rows:

d4	e0	b8	1e	← No rotation
27	bf	b4	41	← Rotate 1 byte
11	98	5d	52	← Rotate 2 bytes
ae	f1	e5	30	← Rotate 3 bytes

AES S-Box: Substitution values in hexadecimal notation for input byte (*xy*)

	y															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
x 8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Mix Columns:

State (Input)

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

x

MixColumns Matrix

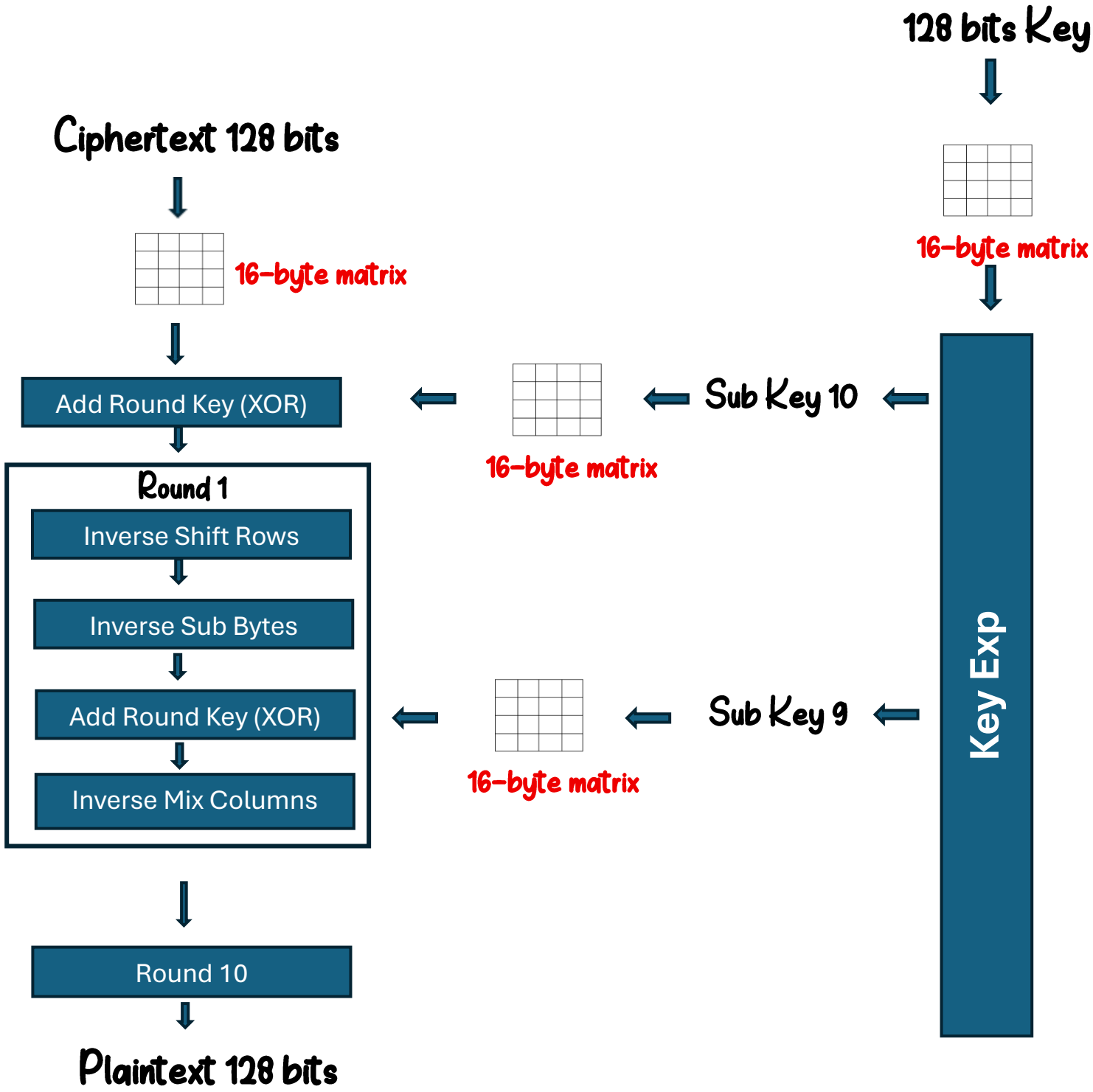
2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

=

State (Output)

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$

Decryption



All rounds include the Mix Columns step except the final round

Inverse Shift Rows:

AB	02	12	91	← No rotation
45	22	33	80	← Rotate 3 bytes
88	63	87	26	← Rotate 2 bytes
39	62	18	16	← Rotate 1 byte

Inverse Mix Columns:

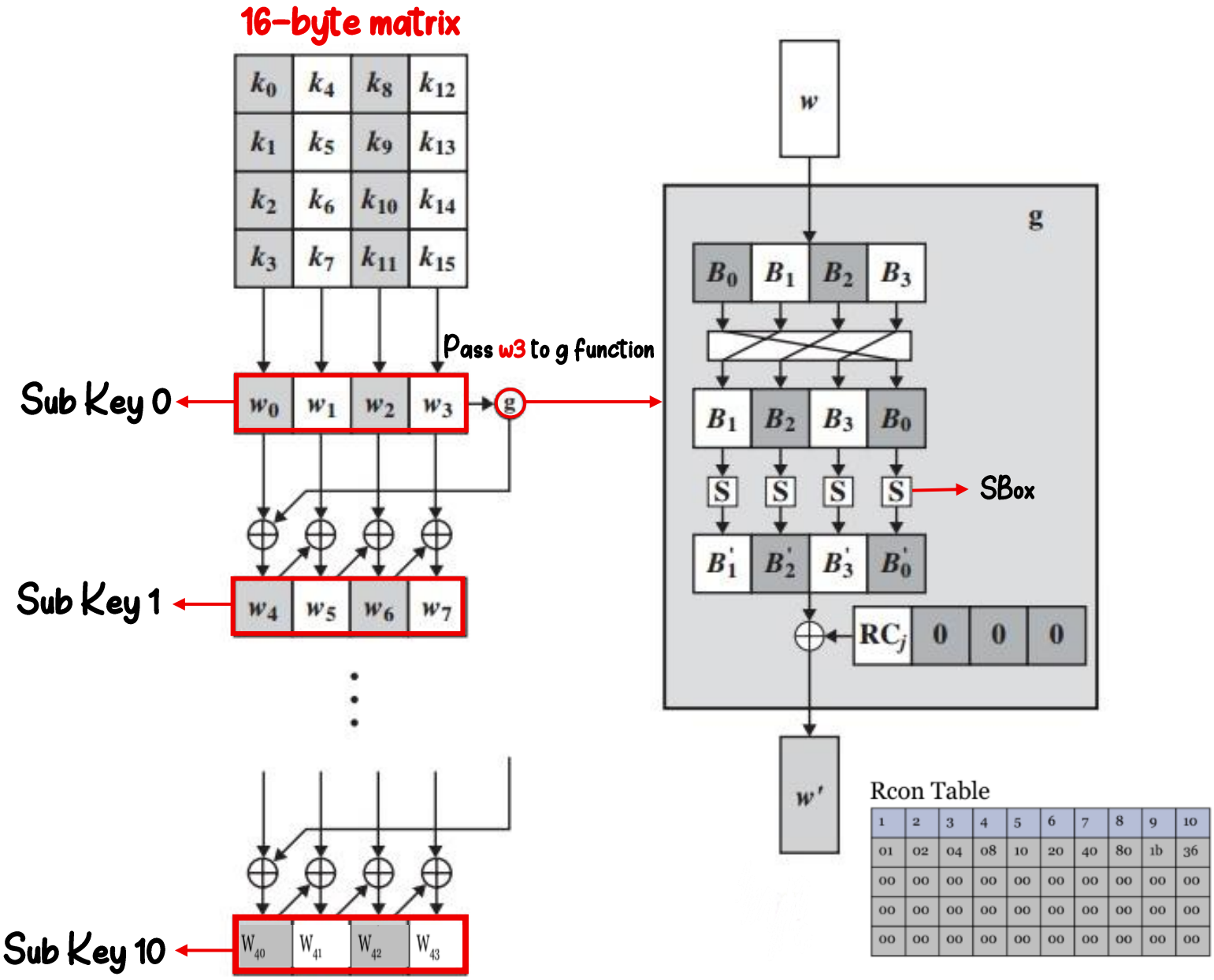
State (Input)					InvMixColumns Matrix					State (Output)			
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	x	14	11	13	9	=	$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$		9	14	11	13		$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$		13	9	14	11		$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$		11	13	9	14		$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$

Inverse SBox:

	y															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
x 8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Key Expansion

Key length (bits)	Number of subkeys
128	11
192	13
256	15

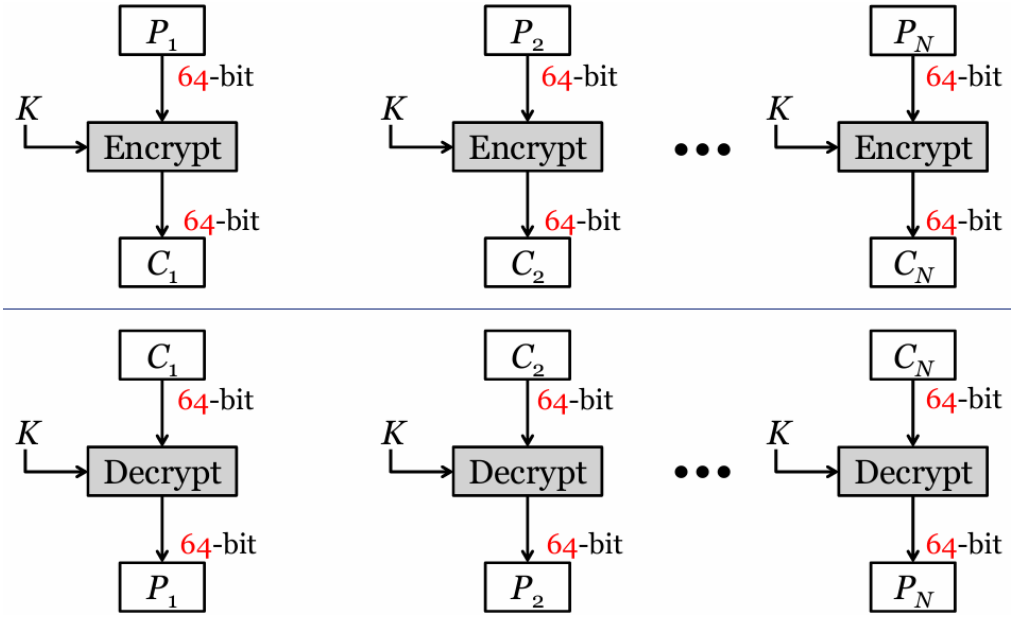


Rcon Table

1	2	3	4	5	6	7	8	9	10
01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

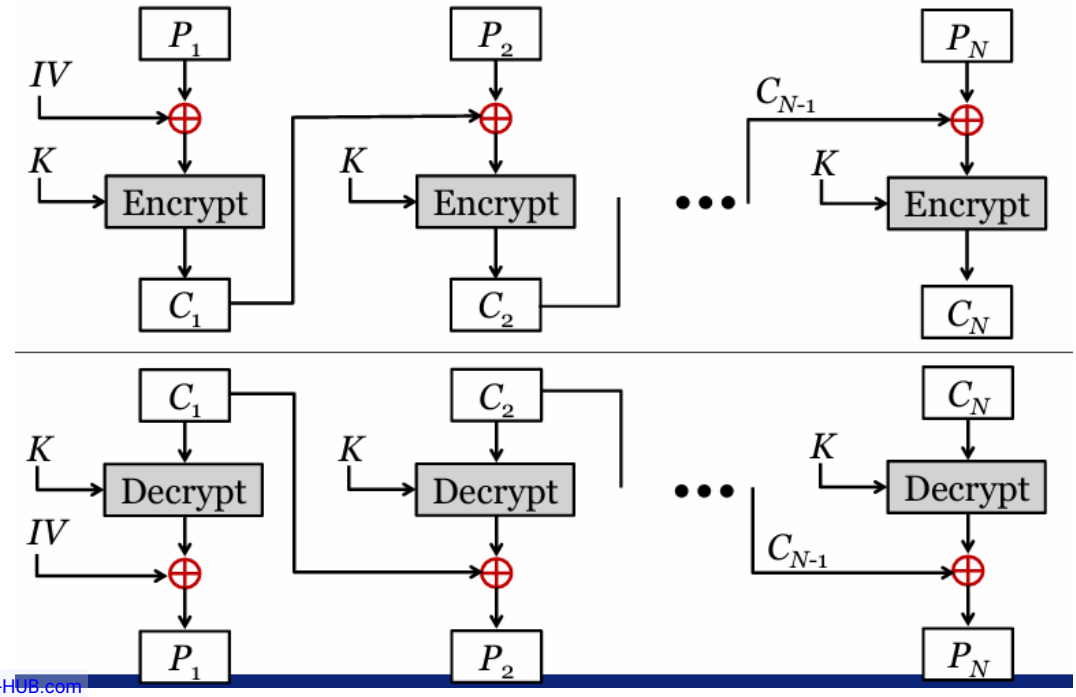
Block Cipher Modes of Operations

- ECB (Encryption & Decryption)

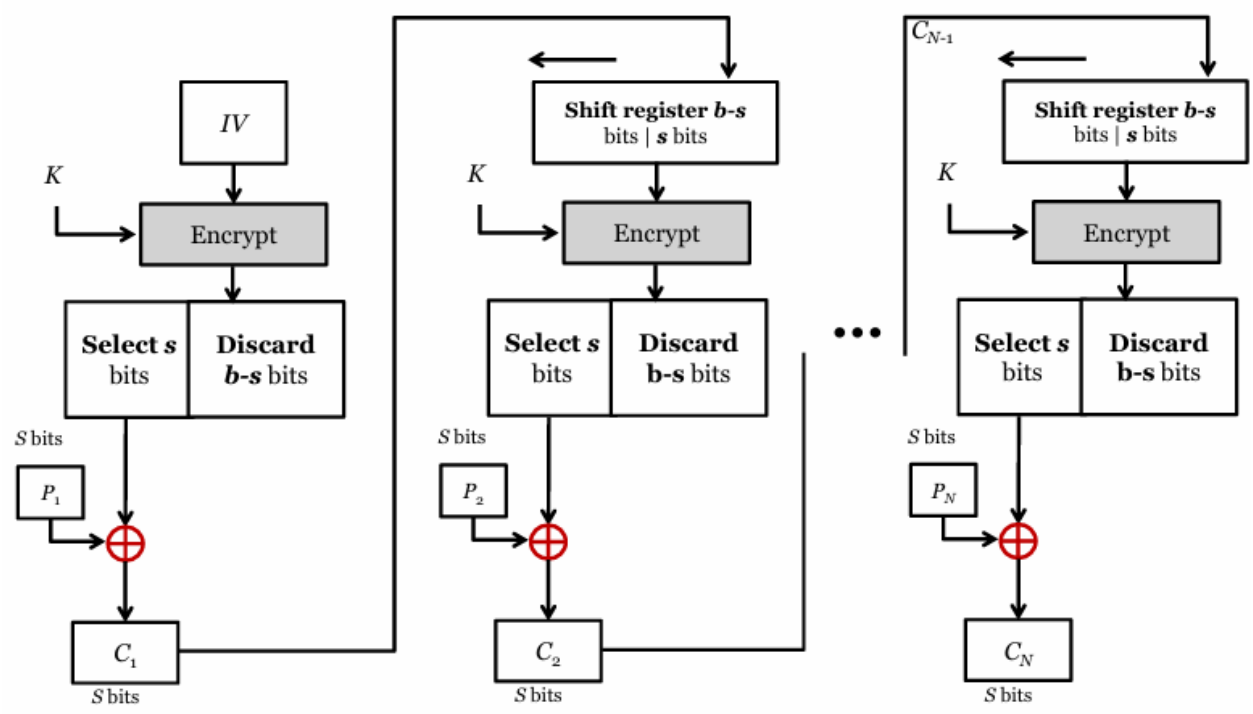


The encryption & decryption block can be implemented using AES, DES, or other block ciphers.

-CBC (Encryption & Decryption)

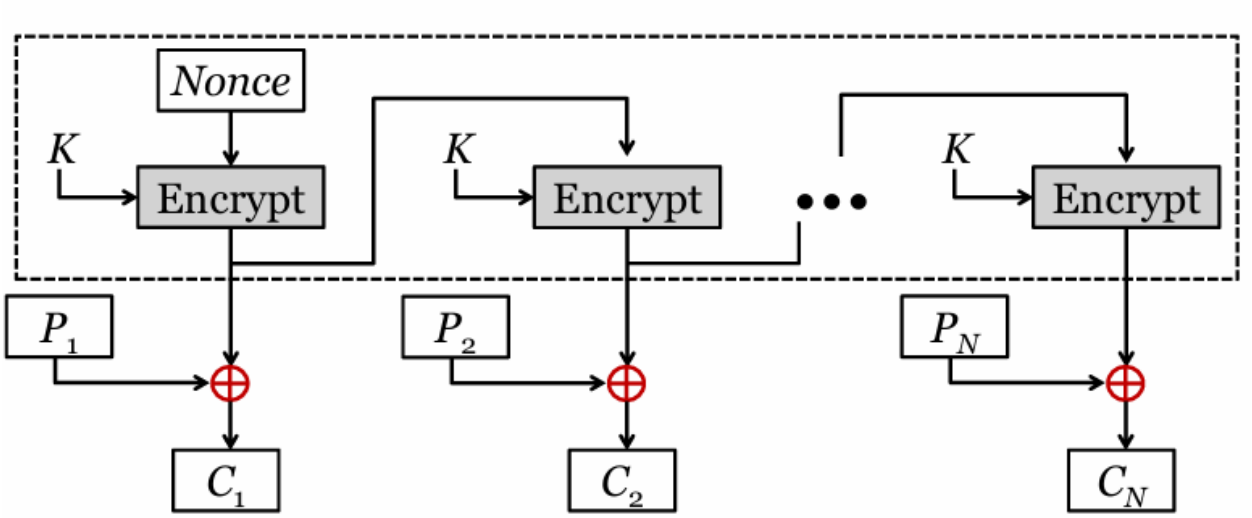


-CFB (Encryption & Decryption)



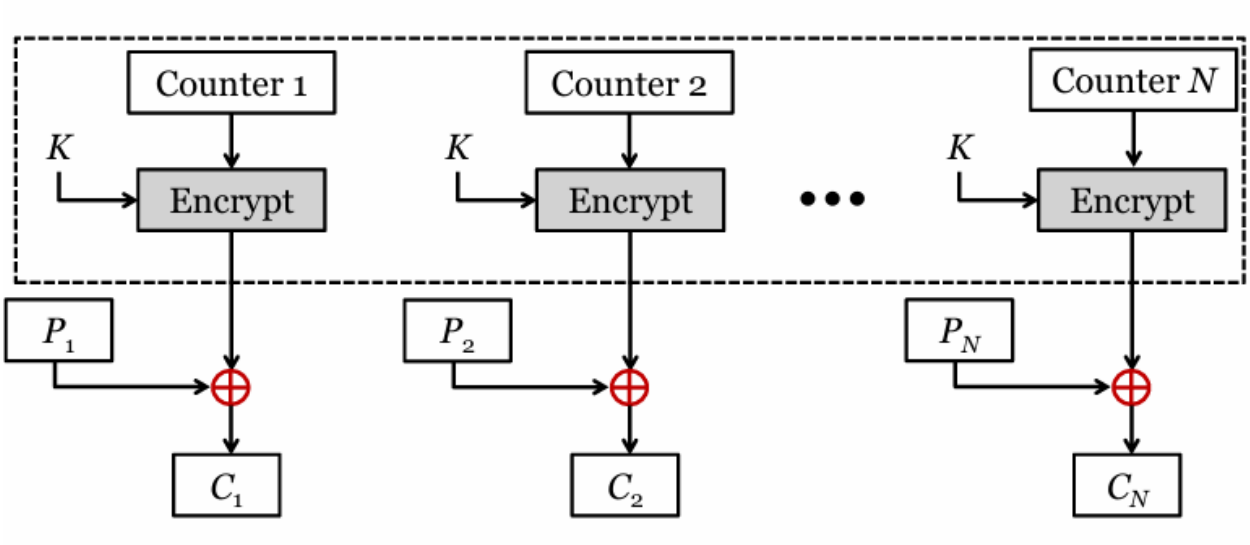
In CFB mode, encryption and decryption are the same because XOR works both ways

OFB (Encryption & Decryption)



CFB and OFB are similar, but in OFB the shift register takes the encryption output instead of the XOR result

CTR (Encryption & Decryption)



In CTR mode, encryption and decryption are the same because both use the encrypted counter XORed with the data.

Summary of all modes

Operation Mode	Description	Type of Result
ECB	Each n-bit block is encrypted independently with same key.	Block Cipher
CBC	Same as ECB, but each block is XORed with previous cipher text.	Block Cipher
CFB	Each s-bit block is XORed with s-bit key which is part of previous cipher text.	Stream Cipher
OFB	Same as CFB, but input to the encryption is preceding encryption output.	Stream Cipher
CTR	Same as OFB, but a counter is used instead of nonce.	Stream Cipher

RSA Algorithm

- RSA is a block cipher in which the Plaintext and Ciphertext are represented as integers between 0 and n-1 for some value of n.
- Large messages can be broken up into a number of blocks.
- Each block would then be represented by an integer.

Step-1: Generate Public key and Private key.

Step-2: Encrypt message using Public key.

Step-3: Decrypt message using Private key.

Step-1: Generate Public key and Private key

- Select two large prime numbers: p and q
- Calculate modulus : $n = p * q$
- Calculate Euler's totient function : $\varphi(n) = (p-1) * (q-1)$
- Select e such that e is relatively prime to $\varphi(n)$ and $1 < e < \varphi(n)$

Two numbers are relatively prime if they have no common factors other than 1 or $(\gcd(\varphi(n), e)=1)$.

- Determine d such that $d * e \equiv 1 \pmod{\varphi(n)}$
- Public key : PU = { e, n }
- Private key : PR = { d, n }

RSA is strong because it's very hard for an attacker to figure out the two prime numbers p and q that multiply to make n

Step-2 and 3: Encryption and Decryption

Step-2: Encryption using Public Key:

Plaintext: M

Ciphertext: $C = M^e \bmod n$

Step-3: Decryption using Private Key:

Ciphertext: C

Plaintext: $M = C^d \bmod n$

Example:

Step-1: Generate Public key and Private key

- Select two large prime numbers: p = 3 and q = 11
- Calculate modulus : n = p * q = 3 * 11 = 33
- Calculate Euler's totient function : $\varphi(n) = (p-1) * (q-1)$

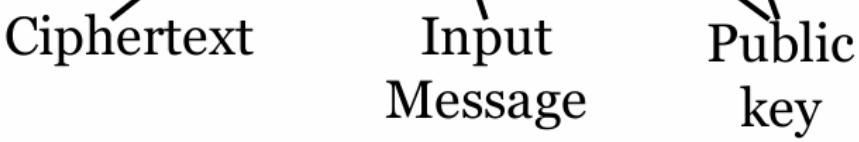
  $\varphi(n) = (3 - 1) * (11 - 1) = 20$
- Select e such that e is relatively prime to $\varphi(n)$ and $1 < e < \varphi(n)$
- We have several choices for e :3, 7, 11, 13, 17, 19 Let's take e = 7
- Determine d such that $d * e \equiv 1 \pmod{\varphi(n)}$
- $? * 7 \equiv 1 \pmod{20}$, $3 * 7 \equiv 1 \pmod{20}$
- $? * 7 \pmod{20} \equiv 1$, $3 * 7 \pmod{20} \equiv 1$
- Public key : PU = { e, n } , PU = { 7, 33 }
- Private key : PR = { d, n }, PR = { 3, 33 }

- This is equivalent to finding d which satisfies $de = 1 + j.\varphi(n)$ where j is any integer.
- We can rewrite this as $d = (1 + j.\ \varphi(n)) / e$

Step-2 : Encrypt Message

- Encryption using Public Key:

$$C = M^e \bmod n$$



PU = { e, n } , PU = { 7, 33 }

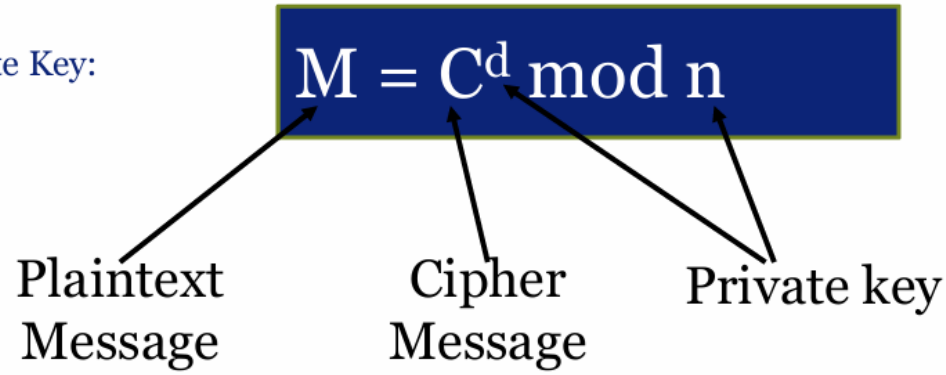
For message M = 14

$C = 14^7 \bmod 33$
$C = 20$

Step-3 : Decrypt Message

- Decryption using Private Key:

$$M = C^d \bmod n$$



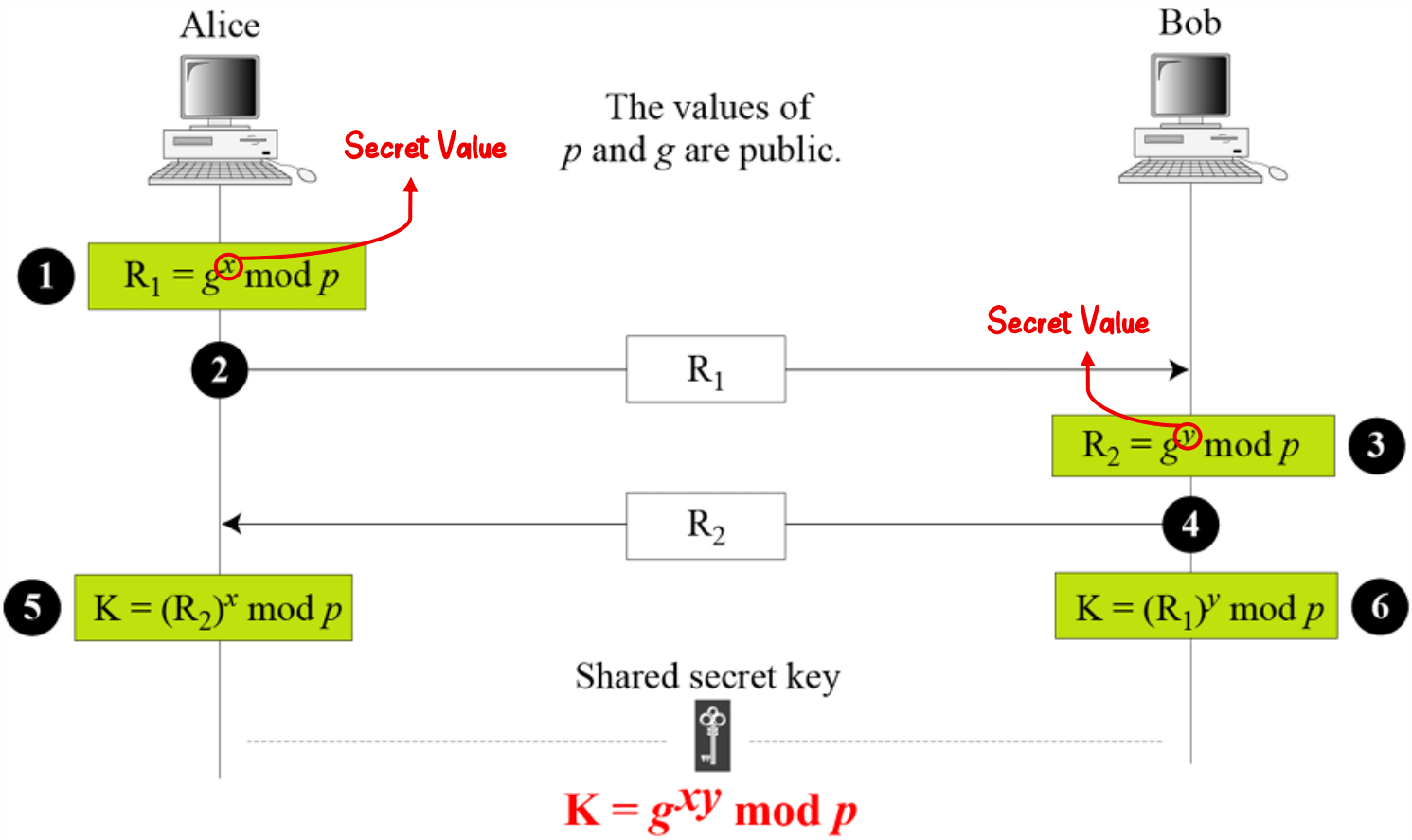
PR = { d, n } , PR = { 3, 33 }

For Ciphertext C = 20

$M = 20^3 \bmod 33$
$M = 14$

Key Management: Diffie-Hellman

- Before establishing a symmetric key, the two parties need to choose two numbers p and g



p : large prime number
g: is a primitive root modulo p

Elgamal Cryptosystem

Sender

Receiver

B = Temporary key that changes with each message

1< B < p-1

Encryption:

C1 = G^B mod P

C2 = M *E^B mod P

Encrypted message

(C1, C2)

Key Generation :

P = prime number

G = primitive root of p

A = private key (1<A<p-1)

E = G^A mod P

Public key (P , G , E)

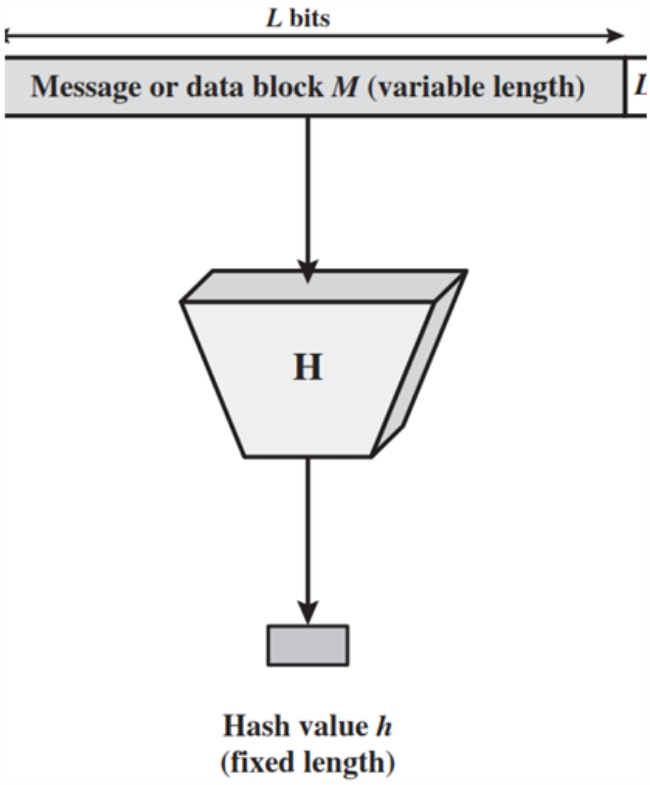
Decryption :

X = c1 ^ a mod p

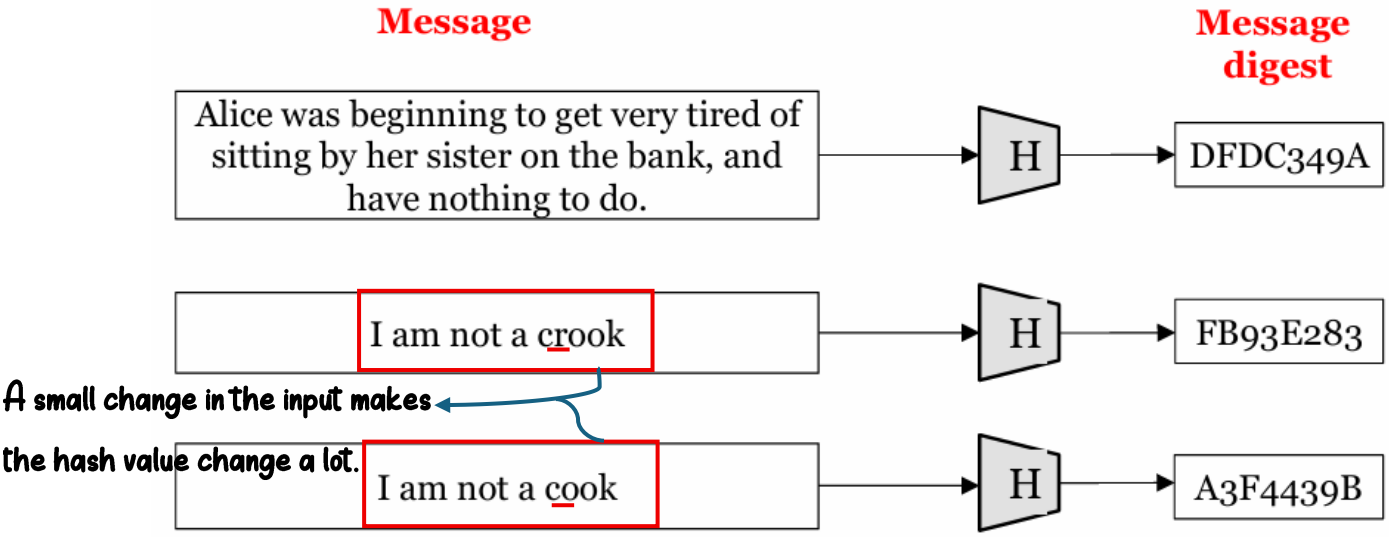
M = c2 * X^(p-2)

Hash Function

- A hash function **H** accepts a variable-length block of data **M** as input and produces a fixed-size hash value **h = H(M)**.
- A "good" hash function has the property that the results of applying a change to any bit or bits in M results, with high probability, in a change to the hash code.
- One way property
- Collision free



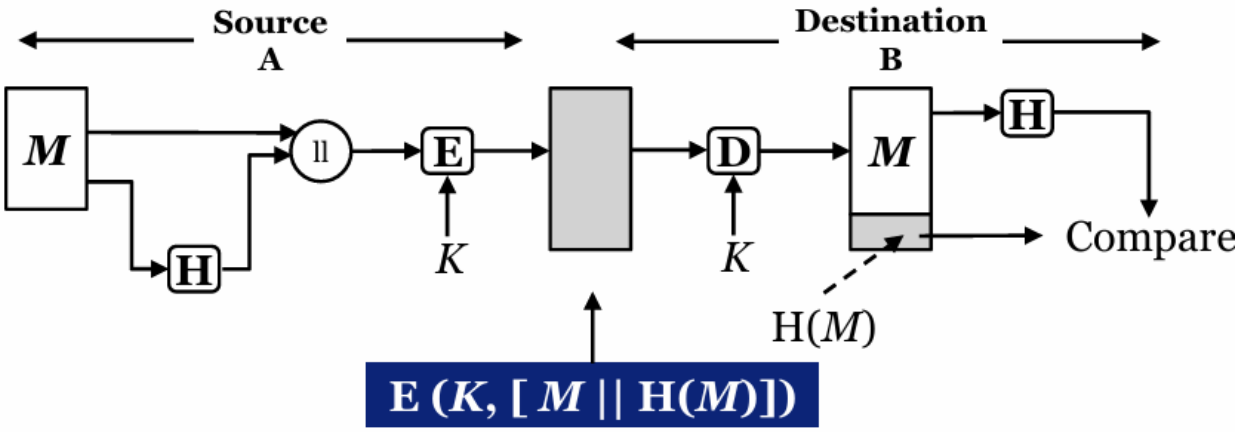
Input-Output behaviour of hash functions



1. Message Authentication

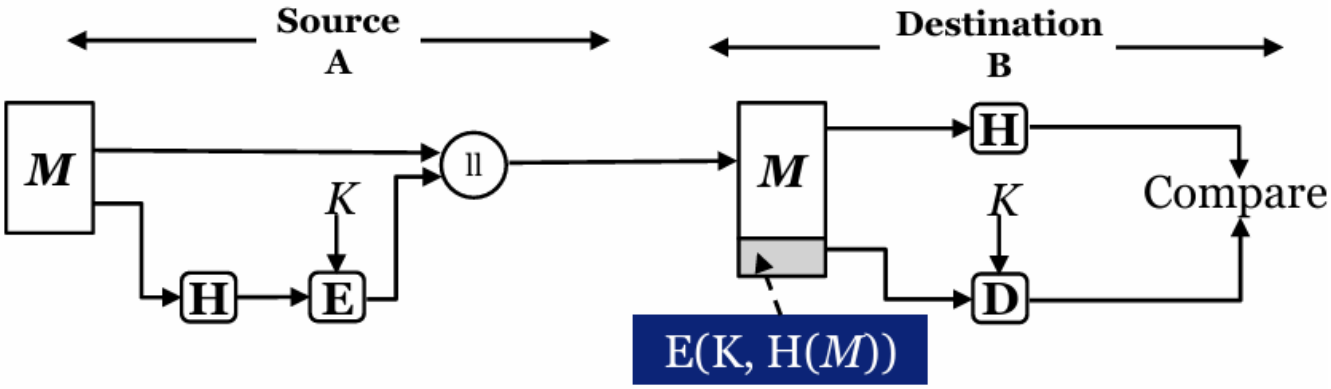
- Message authentication is a mechanism or service used to verify the integrity of a message.
- Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay).
- When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest.

Message authentication method - 1



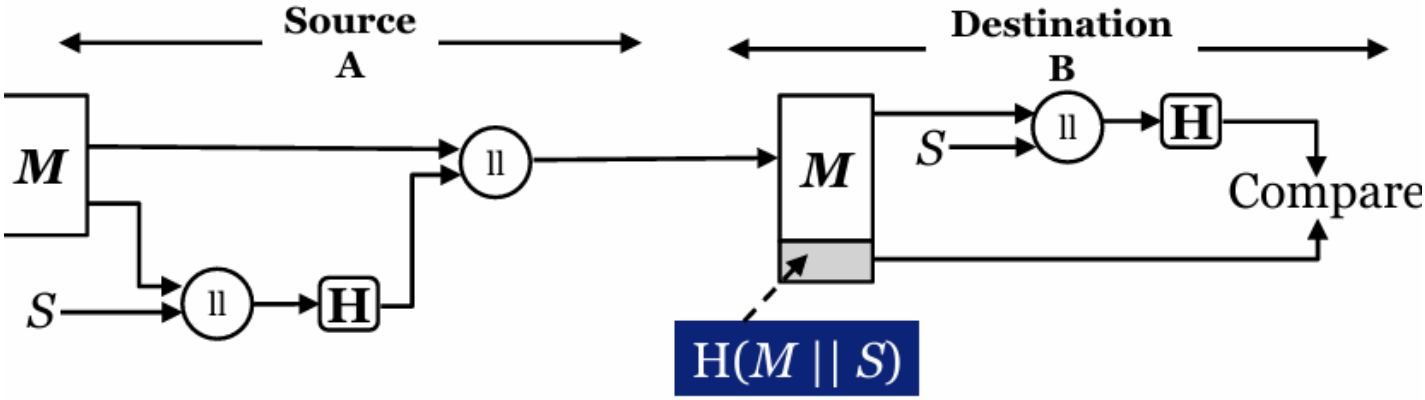
- Only A and B share the secret key, the message must have come from A and has not been altered.
- The hash code provides the structure required to achieve authentication.
- Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

Message authentication method - 2



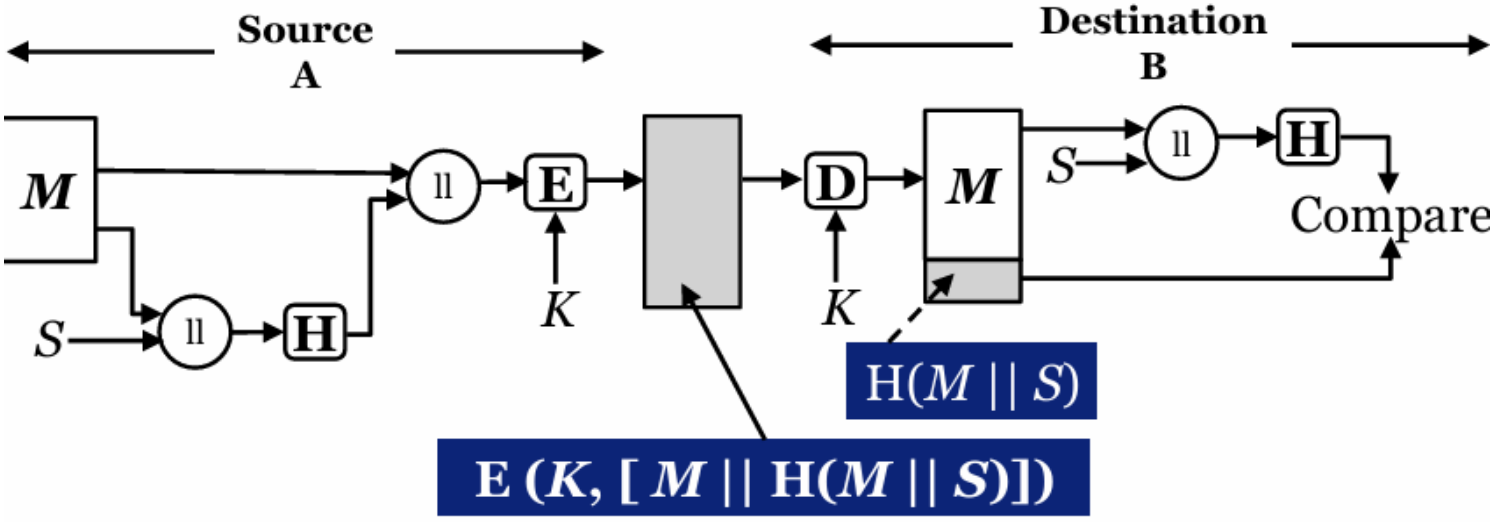
- Only the hash code is encrypted, using symmetric encryption.
- This reduces the processing burden for those applications that do not require confidentiality.

Message authentication method - 3



- It is possible to use a hash function but no encryption for message authentication.
- A and B share a common secret value S.
- A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses S, it can recompute the hash value to verify the message.
- An opponent cannot modify an intercepted message.

Message authentication method - 4



- Confidentiality can be added to the approach of method (3) by encrypting the entire message plus the hash code.

Digital Signature

COMPARISON:

looking at the differences between digital signatures and conventional signatures:

- Inclusion

A conventional signature is part of the document itself. In a digital signature, the signature is sent separately from the document.

- Verification Method

In a conventional signature, the recipient checks the signature on the document against the one on file. In a digital signature, the recipient receives both the message and its signature and uses a verification method to confirm the authenticity

- Relationship

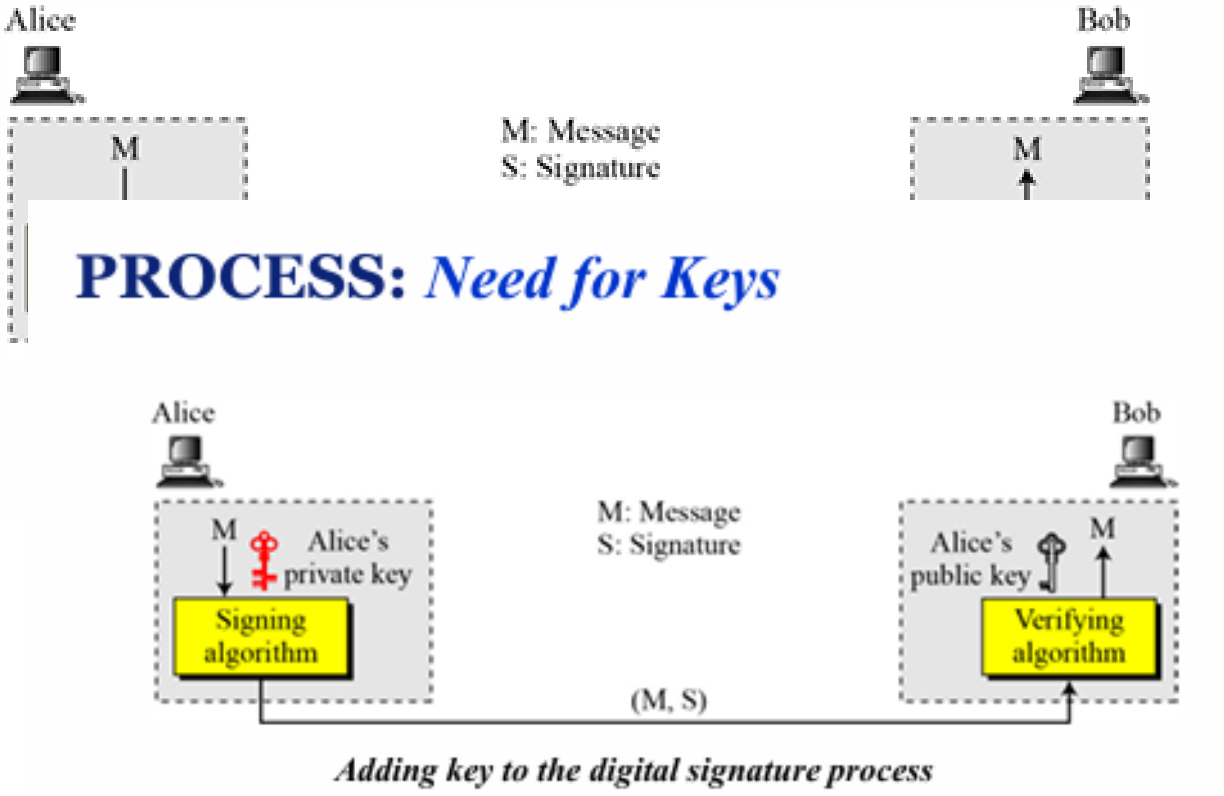
In a conventional signature, one signature can be used for many documents (one-to-many). In a digital signature, each signature is unique to a single message (one-to-one)

- Duplicity

In a conventional signature, a copy of a signed document can be distinguished from the original. In a digital signature, copies cannot be distinguished from the original unless a timestamp is included

PROCESS

The sender creates a message and uses a signing algorithm to generate a signature, then sends both the message and the signature to the receiver. The receiver uses a verification algorithm on the message and signature. If the result is true, the message is accepted; otherwise, it is rejected.



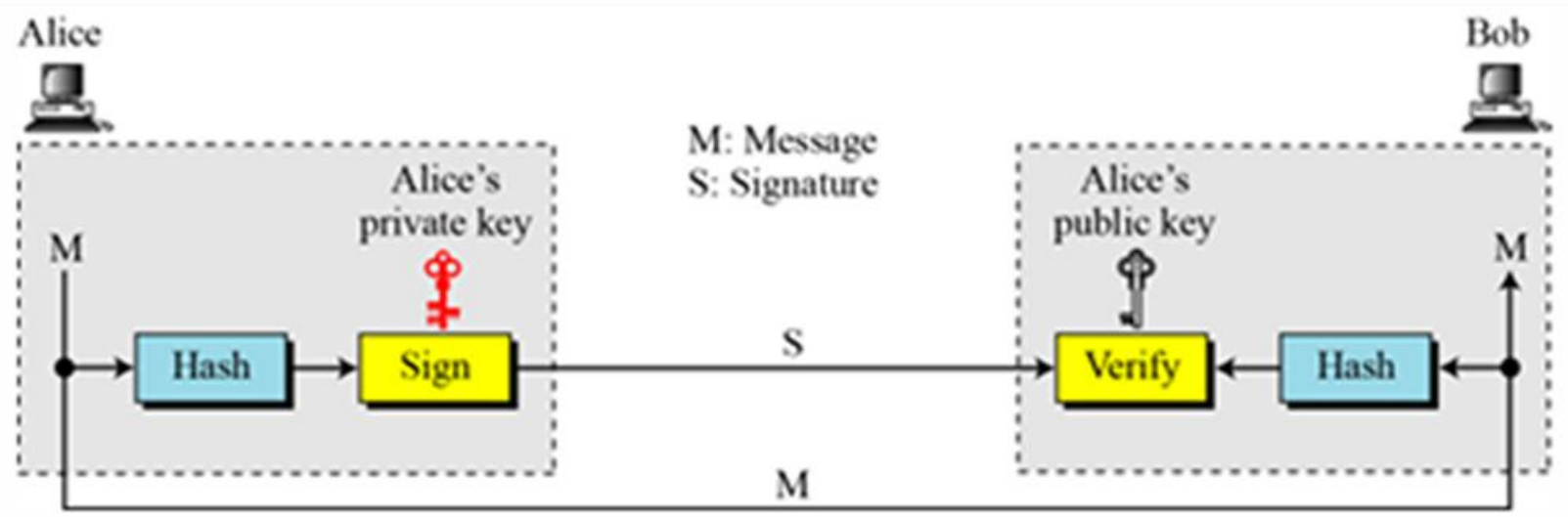
Adding key to the digital signature process

A digital signature needs a public-key system. The signer signs with her private key; the verifier verifies with the signer's public key.

Quick Note:

A cryptosystem uses the private and public keys of the receiver: a digital signature uses the private and public keys of the sender.

Signing the Digest



Signing the digest

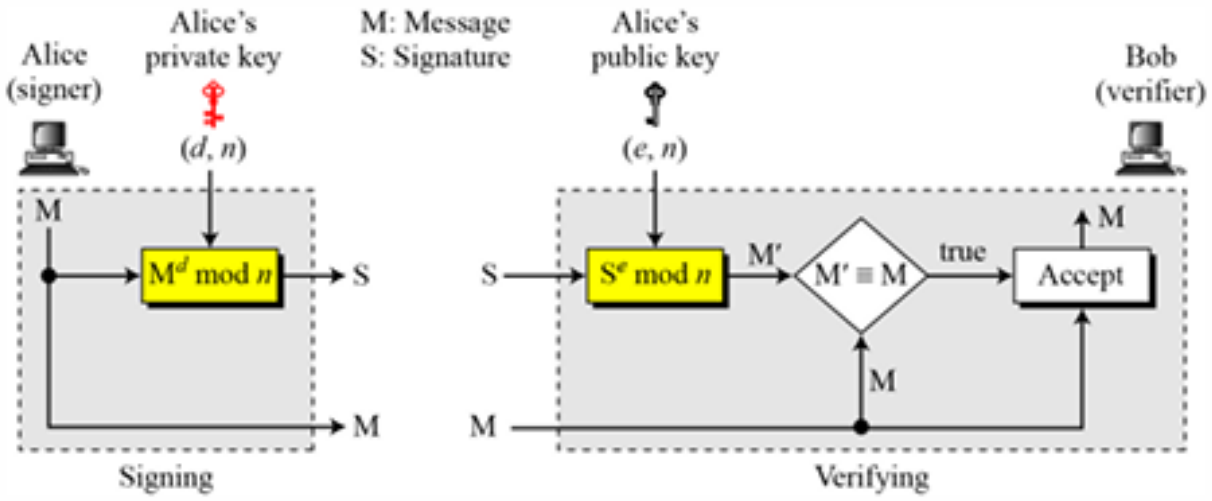
RSA Digital Signature Scheme

Key Generation

Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA

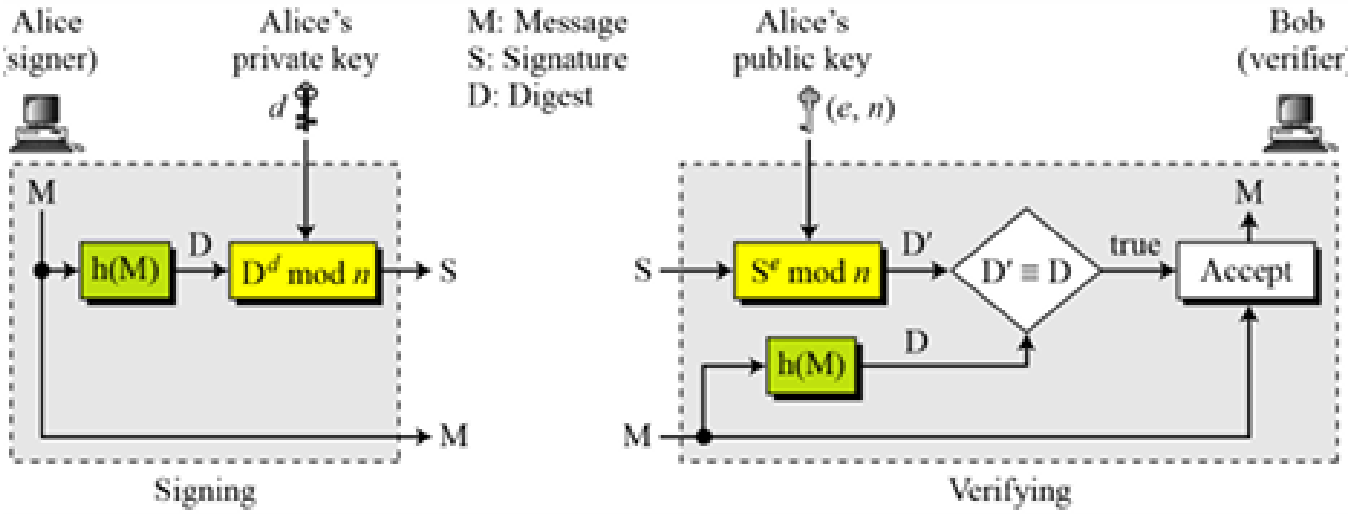
RSA Digital Signature Scheme

Signing and Verifying



RSA Digital Signature Scheme

RSA Signature on the Message Digest



SERVICES:

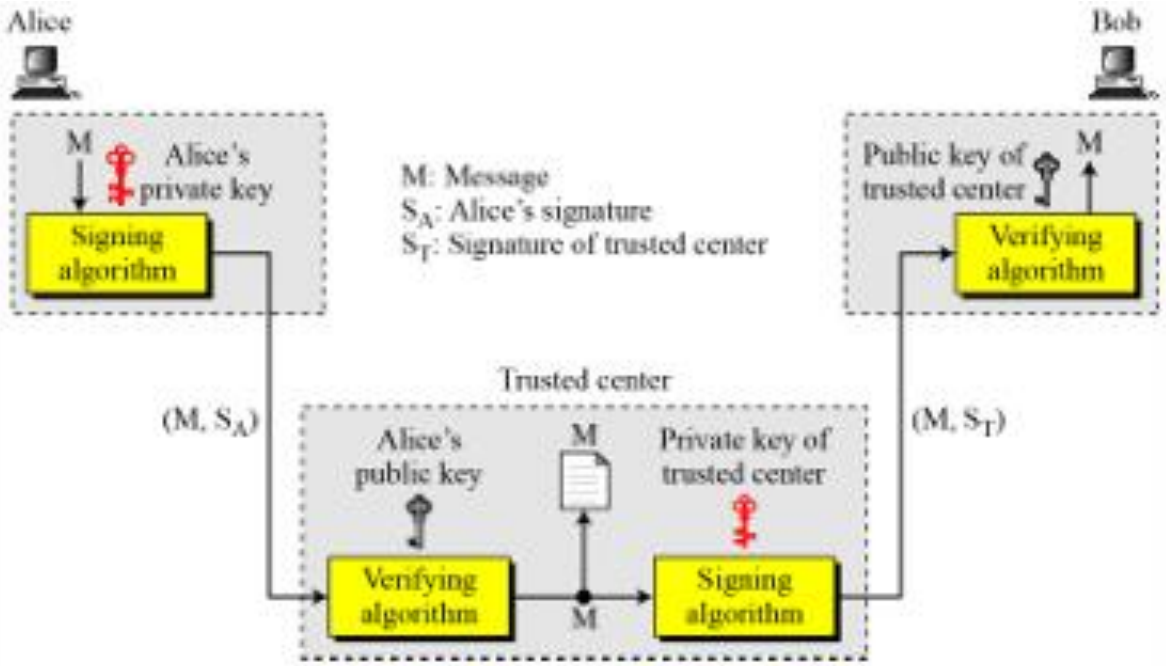
- Message Authentication

A secure digital signature, like a secure conventional signature, can ensure the authenticity of a message

- Message Integrity

The message's integrity is protected because any change to the message will result in a different signature

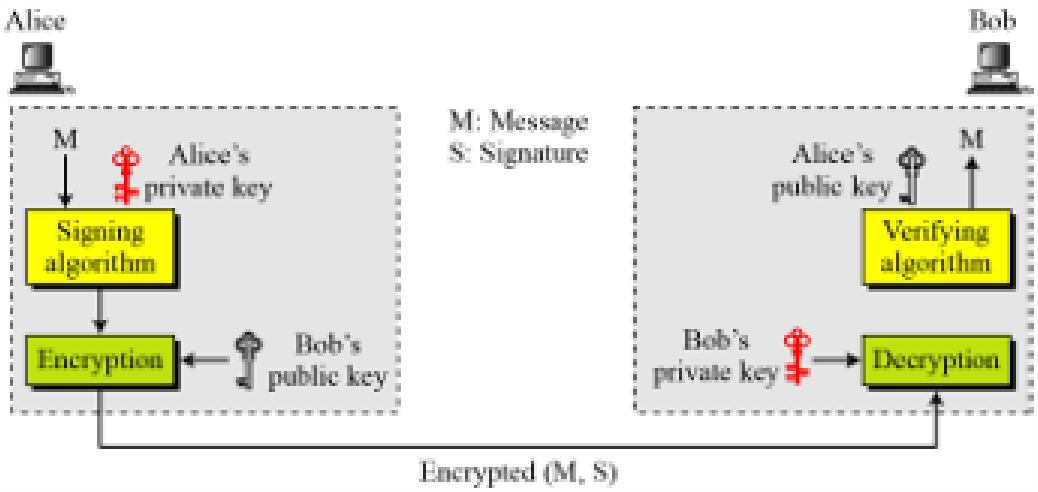
- Nonrepudiation



Using a trusted center for nonrepudiation

Nonrepudiation can be provided using a trusted party.

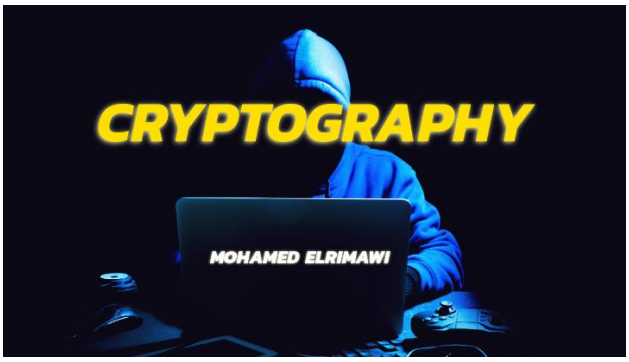
- Confidentiality



Adding confidentiality to a digital signature scheme

A digital signature does not provide privacy. If there is a need for privacy, another layer of encryption/decryption must be applied.

Cryptography Playlist:



https://youtube.com/playlist?list=PLmAUN-bKzQpjZVeyFf4_GvS77COhoMsf0&si=faOmYp_fyz7uielU4

Hill Cipher

Encryption:

https://www.youtube.com/watch?v=ELRmxLkRnyY&list=PL2JGmfJf7X9EOIFr5NI36IkdWcWYXm1Gr&index=14

Decryption:

https://www.youtube.com/watch?v=ZjxKREIc3jc&list=PL2JGmFJF7XgEOIFr5NI36IkdWcWYXm1Gr&index=12