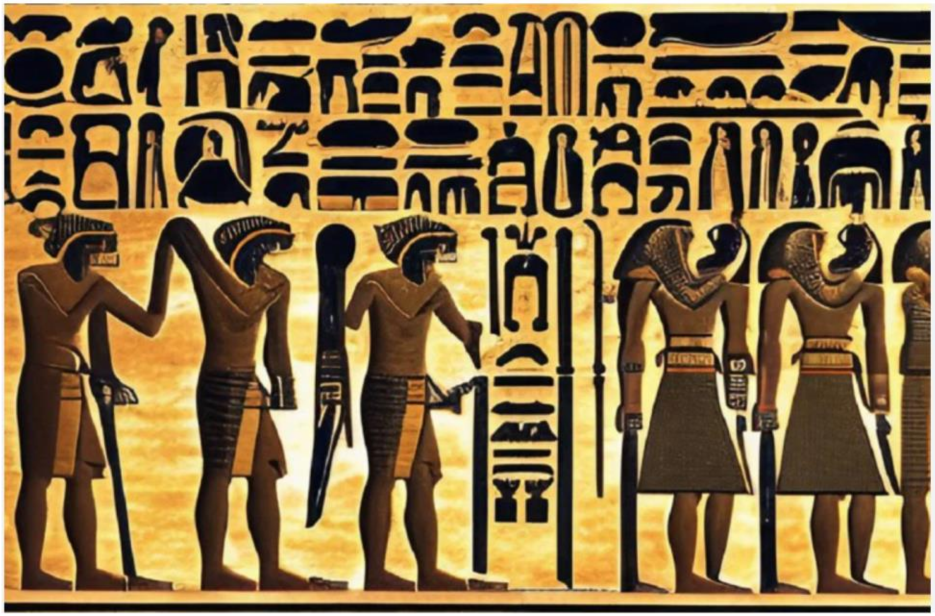
Cryptography

Unit-3

Cryptography

Ancient Cryptography



Ancient Cryptography

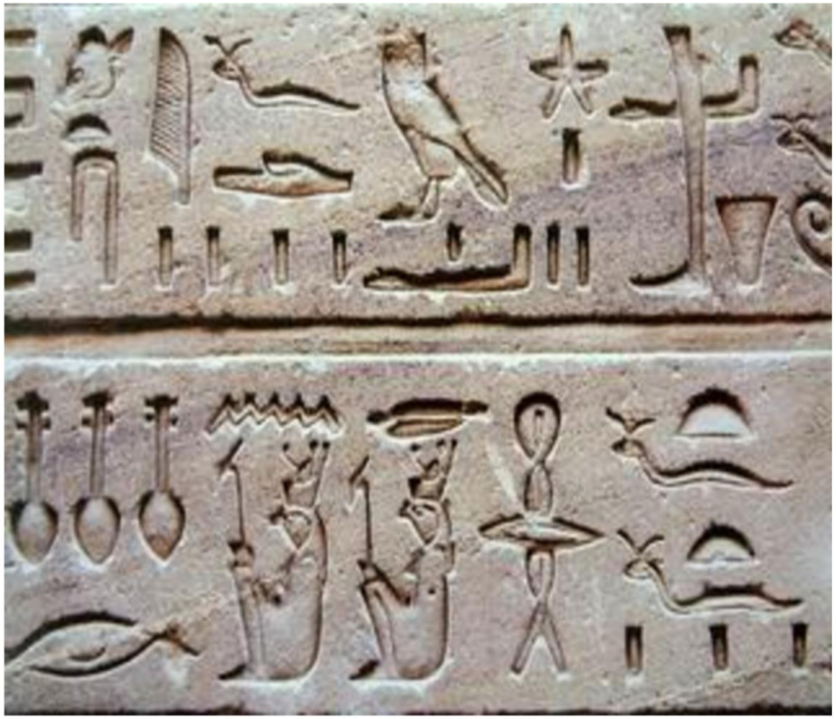
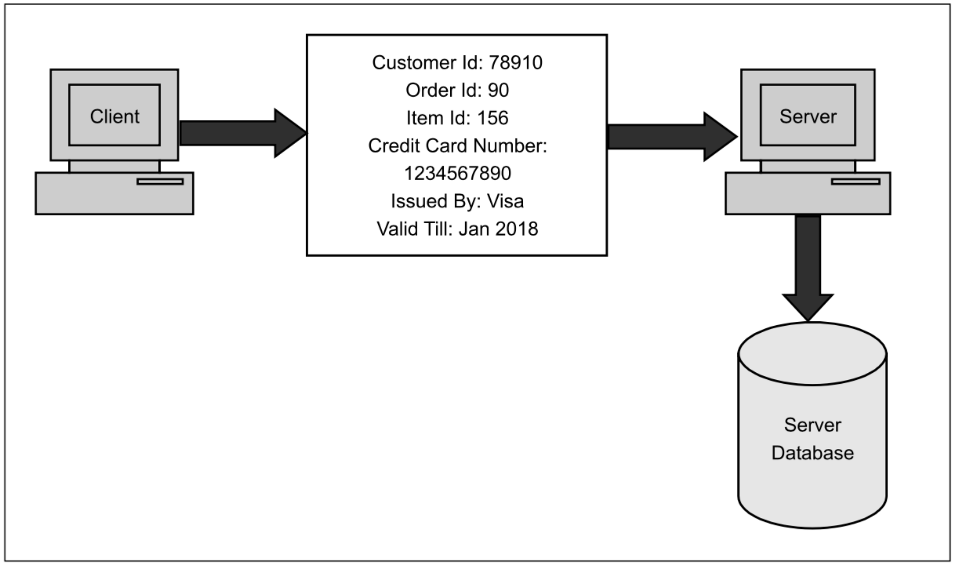


Fig. 2. Symbols used by Ancient Egypt

Need for Security

- Earlier computers had no or a best very little security
- This continued for number of years.
- Need for Security realized when financial and personal data needs security and privacy.



Need for Security

- Provide a user identification and password to every user, and use that information to authenticate a user.
- Encode information stored in the databases in some fashion, so that it is not visible to users who do not have the right permission.

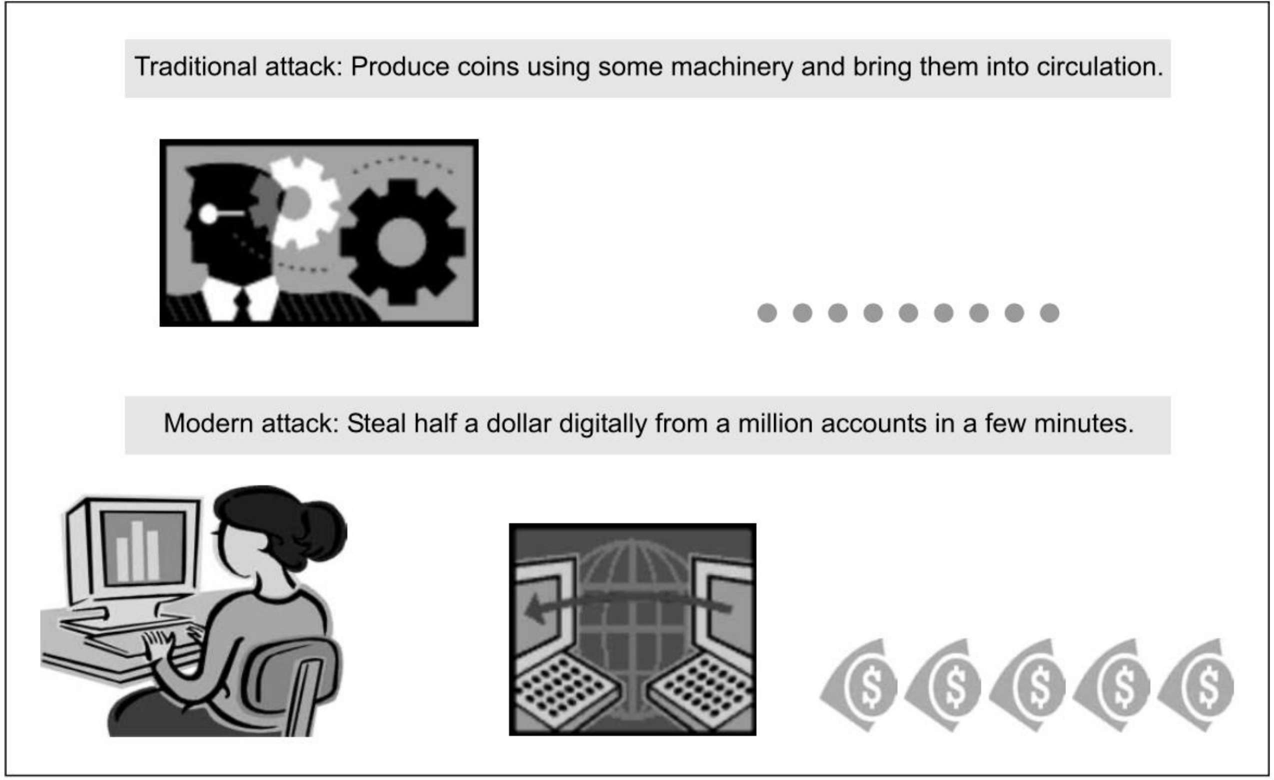
What are the various Security Holes?

Security Holes

- Intruder can capture the credit card details
- Merchant database can be hacked
- Example: Russian attacker (maxim) managed to intrude merchant site and obtained 300,000 credit card members. Demanded protection money $100,000 from the merchant

Modern Nature of Attacks

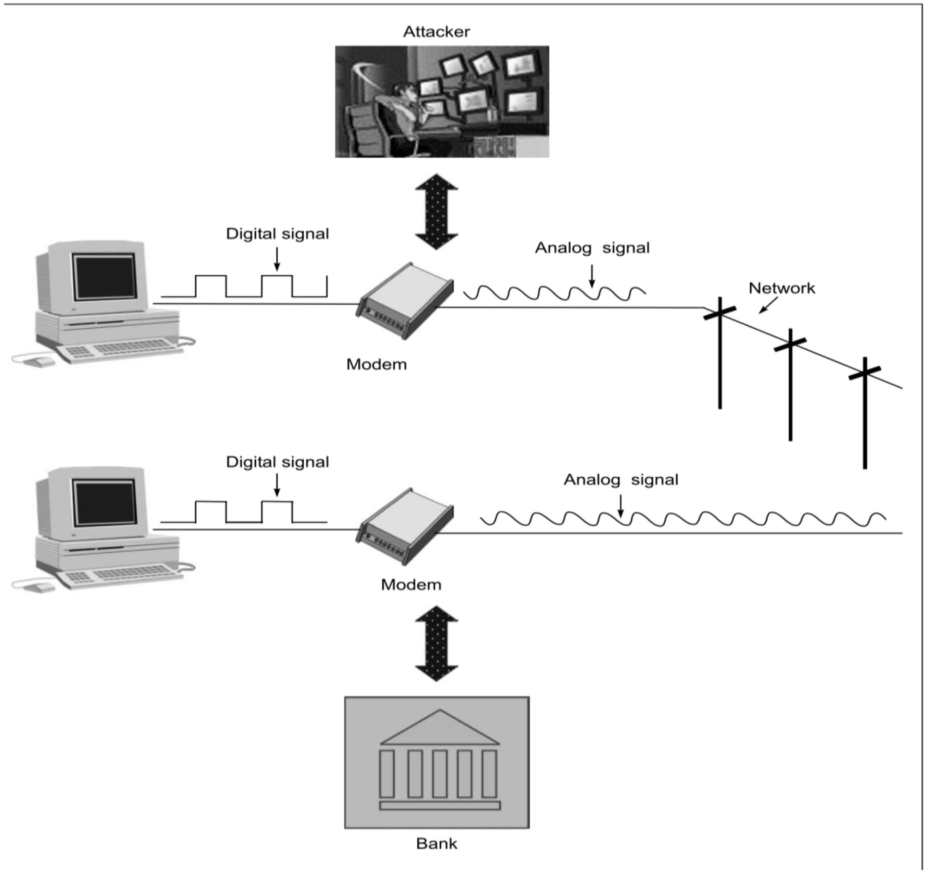
Automating attacks



Traditional attack: Produce coins using some machinery and bring them into circulation.

Modern attack: Steal half a dollar digitally from a million accounts in a few minutes.

Distance does not matter

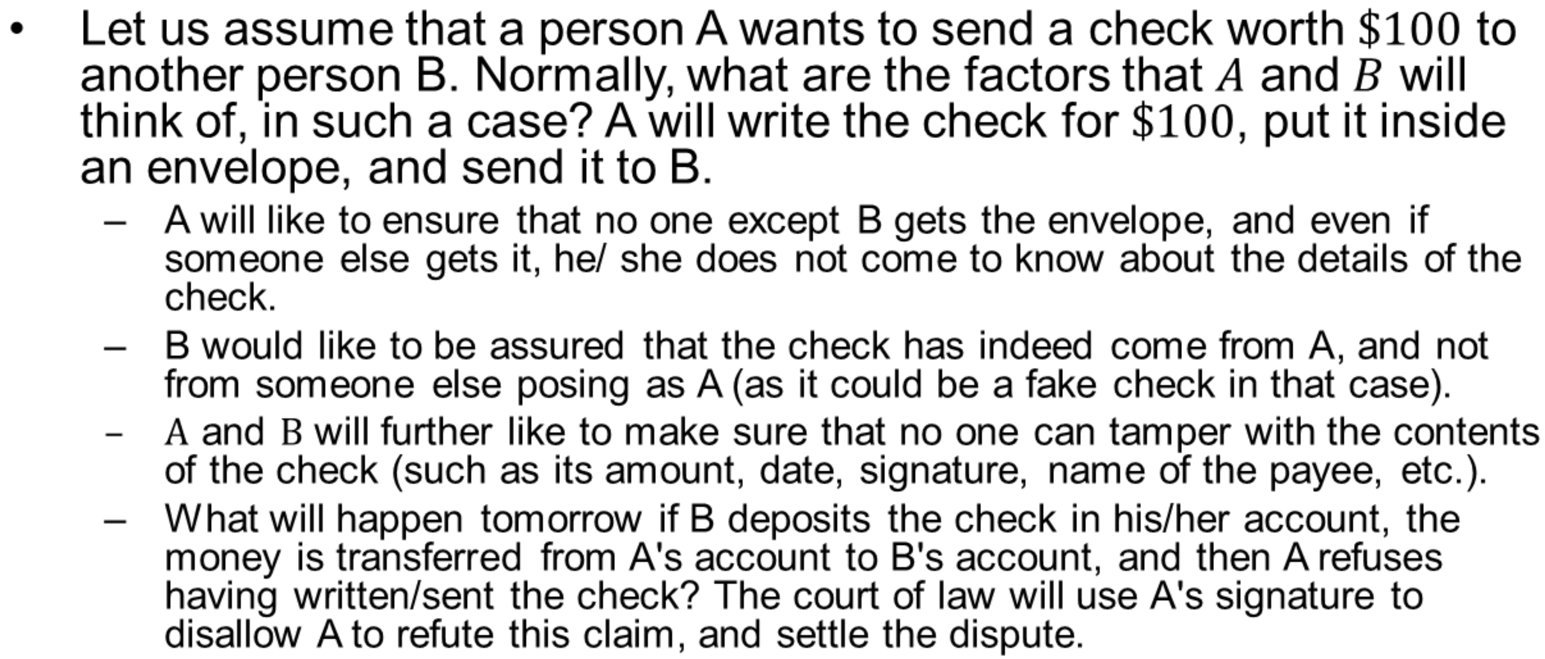


Principle of Security

- Let us assume that a person A wants to send a check worth $\$100$ to another person B. Normally, what are the factors that A and B will think of, in such a case? A will write the check for $\$100$, put it inside an envelope, and send it to B.

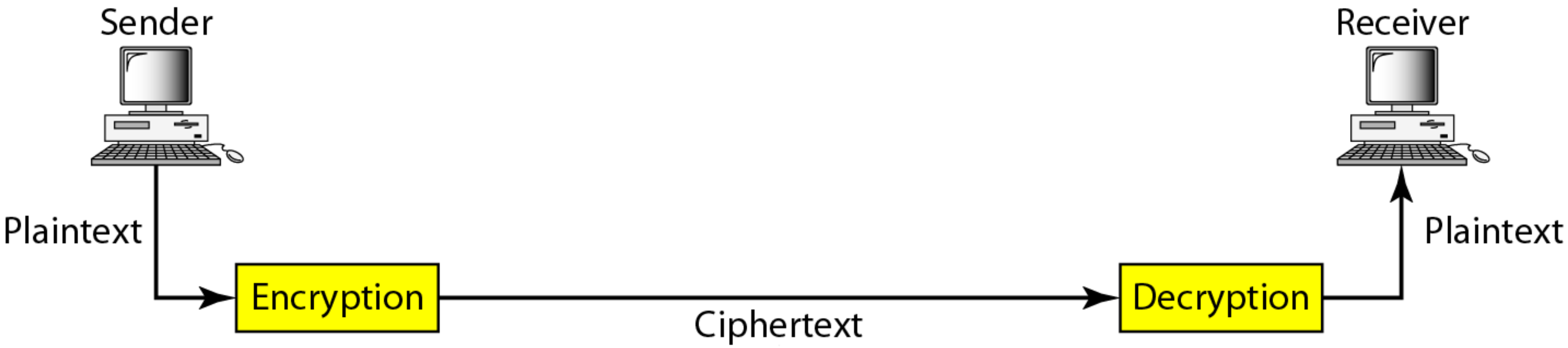
What can be the possible ways we can enforce security?

Principle of Security



- Let us assume that a person A wants to send a check worth $\$100$ to another person B. Normally, what are the factors that A and B will think of, in such a case? A will write the check for $\$100$, put it inside an envelope, and send it to B.
  - A will like to ensure that no one except B gets the envelope, and even if someone else gets it, he/ she does not come to know about the details of the check.
  - B would like to be assured that the check has indeed come from A, and not from someone else posing as A (as it could be a fake check in that case).
  - A and B will further like to make sure that no one can tamper with the contents of the check (such as its amount, date, signature, name of the payee, etc.).
  - What will happen tomorrow if B deposits the check in his/her account, the money is transferred from A's account to B's account, and then A refuses having written/sent the check? The court of law will use A's signature to disallow A to refute this claim, and settle the dispute.

Figure 30.1 *Cryptography components*



Basic Terminology

- **plaintext** - the original message
- **ciphertext** - the coded message
- **cipher** - algorithm for transforming plaintext to ciphertext
- **key** - info used in cipher known only to sender/receiver
- **encipher (encrypt)** - converting plaintext to ciphertext
- **decipher (decrypt)** - recovering ciphertext from plaintext
- **cryptography** - study of encryption principles/methods
- **cryptanalysis (codebreaking)** - the study of principles/ methods of deciphering ciphertext *without* knowing key
- **cryptology** - the field of both cryptography and cryptanalysis

Requirements

- two requirements for secure use of symmetric encryption:
  - a strong encryption algorithm
  - a secret key known only to sender / receiver

    $Y = \mathrm{E}_K(X)$

    $X = \mathrm{D}_K(Y)$
- assume encryption algorithm is known
- implies a secure channel to distribute key

Types of encryption

- Symmetric Encryption
- Asymmetric Encryption

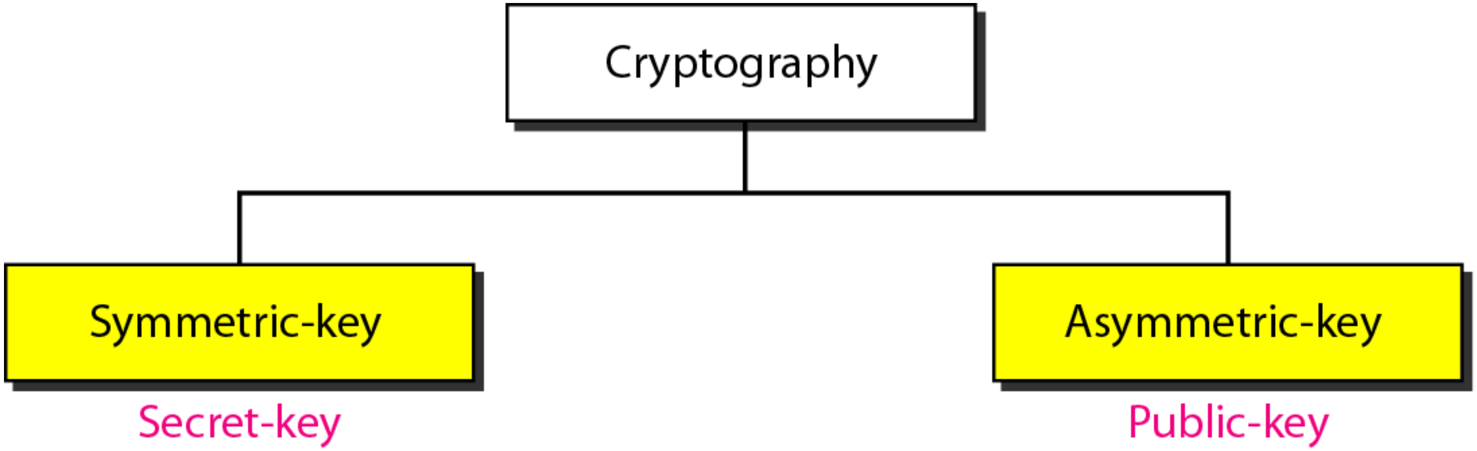
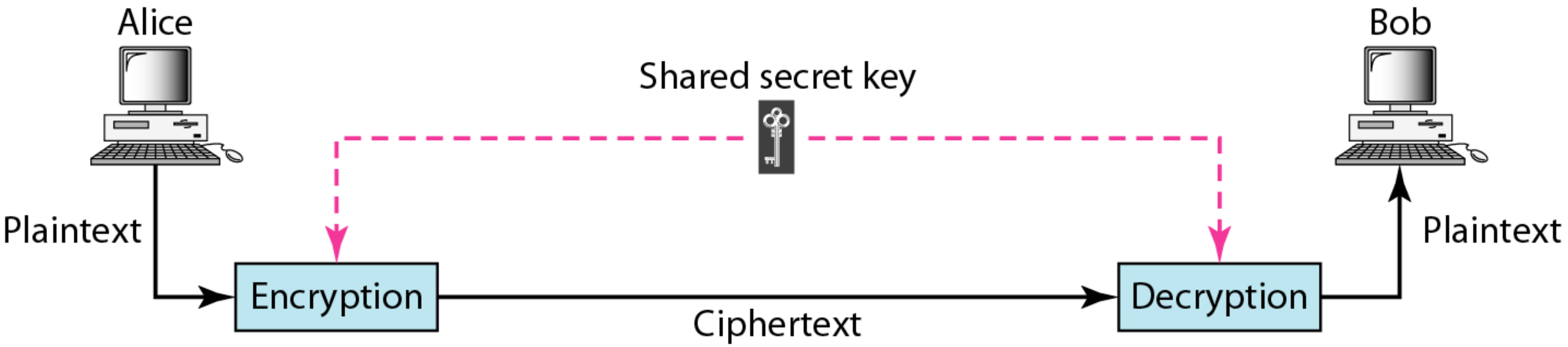


Figure 30.3 *Symmetric-key cryptography*



Symmetric Encryption

- A systems where by knowing the algorithm and the key, you can both encrypt and decrypt a message. This kind of encryption is known as symmetric encryption.
- The big advantage of this kind of encryption is that it is easy because it does not require complex math and much calculation to execute.
- On the other hand, it makes critical the key exchange moment and key management. In fact, the key has to be exchanged before the transmission can start between the parties, and it has to be done securely.

Symmetric Encryption

- As for the key management problem, since both parties know the same secret (in fact, this kind of cryptography is also called shared secret), if you have multiple people that have to communicate with each other, you will need *n(n-1)/2* keys
- This means that in a group of 20 people, you’ll need 190 keys.

Symmetric Encryption

- The types of symmetric encryption that are used are as follows:
  - Stream cipher (ex, RC4) (WEP, WPA, SSL, TSL)
  - Block cipher (AES, DES, 3DES)

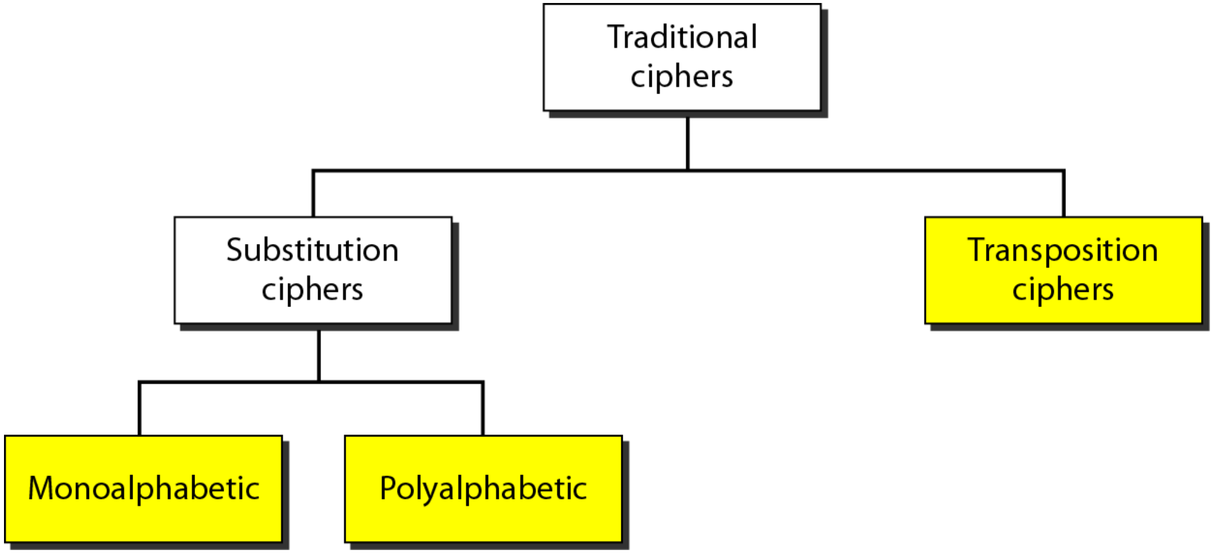
Note

In symmetric-key cryptography, the same key is used by the sender (for encryption) and the receiver (for decryption). The key is shared.

Cryptography

- can characterize by:
  - type of encryption operations used
    - substitution / transposition / product
  - number of keys used
    - single-key or private / two-key or public
  - way in which plaintext is processed
    - block / stream

Figure 30.7 *Traditional ciphers*



Monoalphabetic Ciphers

We first discuss a group of substitution ciphers called the **monoalphabetic ciphers.** In monoalphabetic substitution, a character (or a symbol) in the plaintext is always changed to the same character (or symbol) in the ciphertext regardless of its position in the text. For example, if the algorithm says that letter A in the plaintext is changed to letter D, every letter A is changed to letter D. In other words, the relationship between letters in the plaintext and the ciphertext is one-to-one.

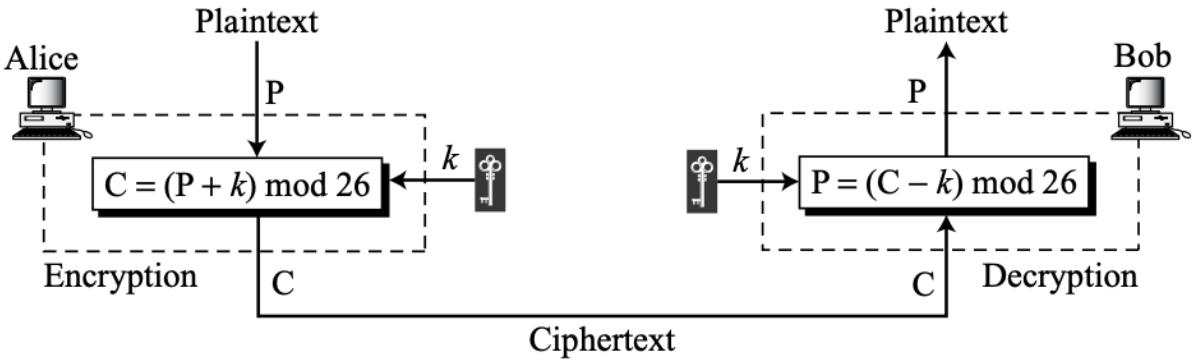
In monoalphabetic substitution, the relationship between a symbol in the plaintext to a symbol in the ciphertext is always one-to-one.

Figure 3.8 *Representation of plaintext and ciphertext characters in* $\boldsymbol{Z}_{26}$

Plaintext →	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
Ciphertext →	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
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Additive Cipher/Shift Cipher/Caesar Cipher

Figure 3.9 *Additive cipher*



Use the additive cipher with key = 15 to encrypt the message “hello”.

Use the additive cipher with key = 15 to encrypt the message “hello”.

Plaintext: h → 07	Encryption: (07 + 15) mod 26	Ciphertext: 22 → W
Plaintext: e → 04	Encryption: (04 + 15) mod 26	Ciphertext: 19 → T
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: o → 14	Encryption: (14 + 15) mod 26	Ciphertext: 03 → D

The result is “WTAAD”. Note that the cipher is monoalphabetic because two instances of the same plaintext character (l’s) are encrypted as the same character (A).

Use the additive cipher with key = 15 to decrypt the message “WTAAD”.

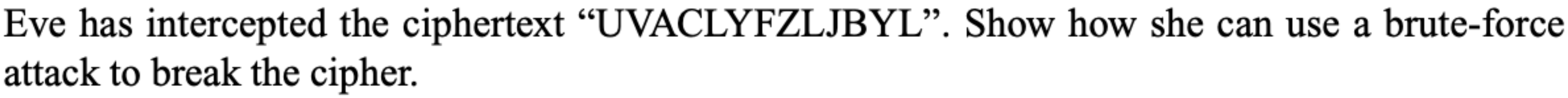
We apply the decryption algorithm to the plaintext character by character:

Ciphertext: W → 22	Decryption: (22 – 15) mod 26	Plaintext: 07 → h
Ciphertext: T → 19	Decryption: (19 – 15) mod 26	Plaintext: 04 → e
Ciphertext: A → 00	Decryption: (00 – 15) mod 26	Plaintext: 11 → l
Ciphertext: A → 00	Decryption: (00 – 15) mod 26	Plaintext: 11 → l
Ciphertext: D → 03	Decryption: (03 – 15) mod 26	Plaintext: 14 → o

Note

The shift cipher is sometimes referred to as the Caesar cipher. (monoalphabetic)

CryptAnalysis
BruteForce Attack



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

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

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

Crypt-Analysis
Statistical Attack

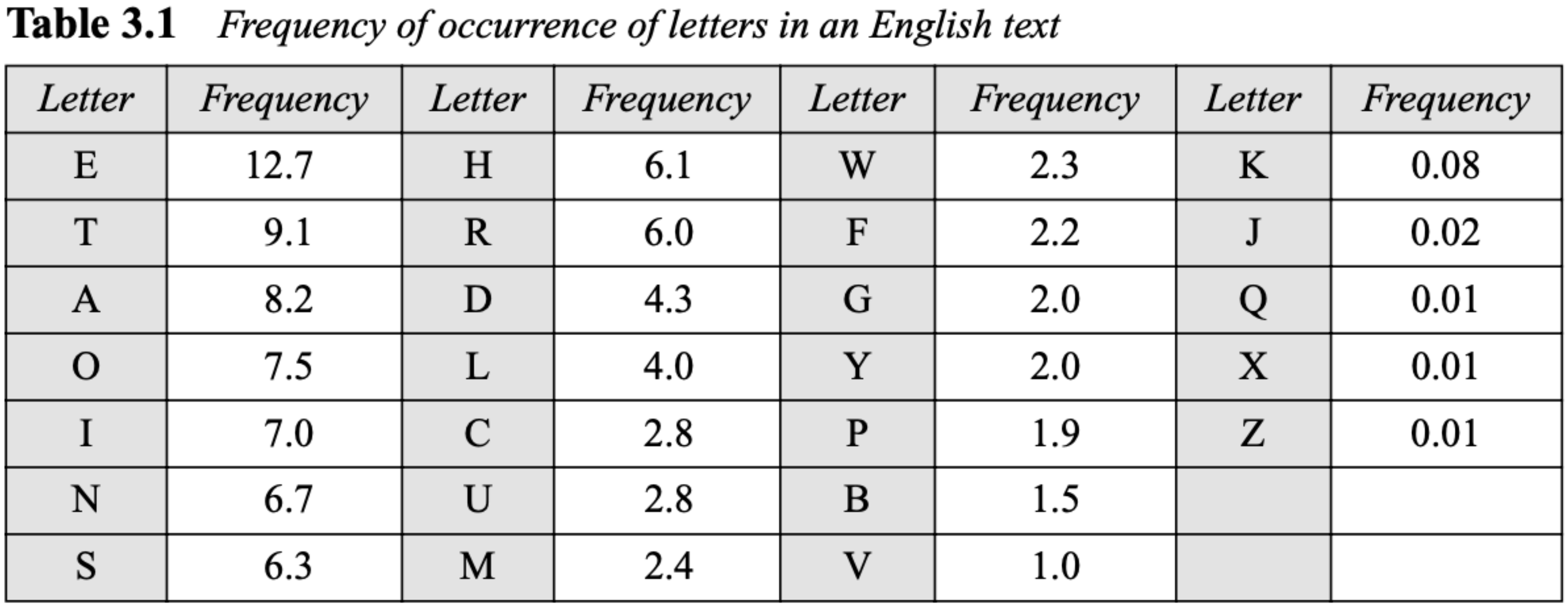


Table 3.1 *Frequency of occurrence of letters in an English text*

Letter	*Frequency*	*Letter*	*Frequency*	*Letter*	*Frequency*	*Letter*	*Frequency*
E	12.7	H	6.1	W	2.3	K	0.08
T	9.1	R	6.0	F	2.2	J	0.02
A	8.2	D	4.3	G	2.0	Q	0.01
O	7.5	L	4.0	Y	2.0	X	0.01
I	7.0	C	2.8	P	1.9	Z	0.01
N	6.7	U	2.8	B	1.5		
S	6.3	M	2.4	V	1.0		

Eve has intercepted the following ciphertext. Using a statistical attack, find the plaintext.

XLILSYWIMWRSAJSVWEPIJSVJSYVQMPPMSRHSPPEVWMXMWASVX-LQSVILYVVCFIJSVIXLIWIPPIVVIGIMZIWQSVISJJIVW

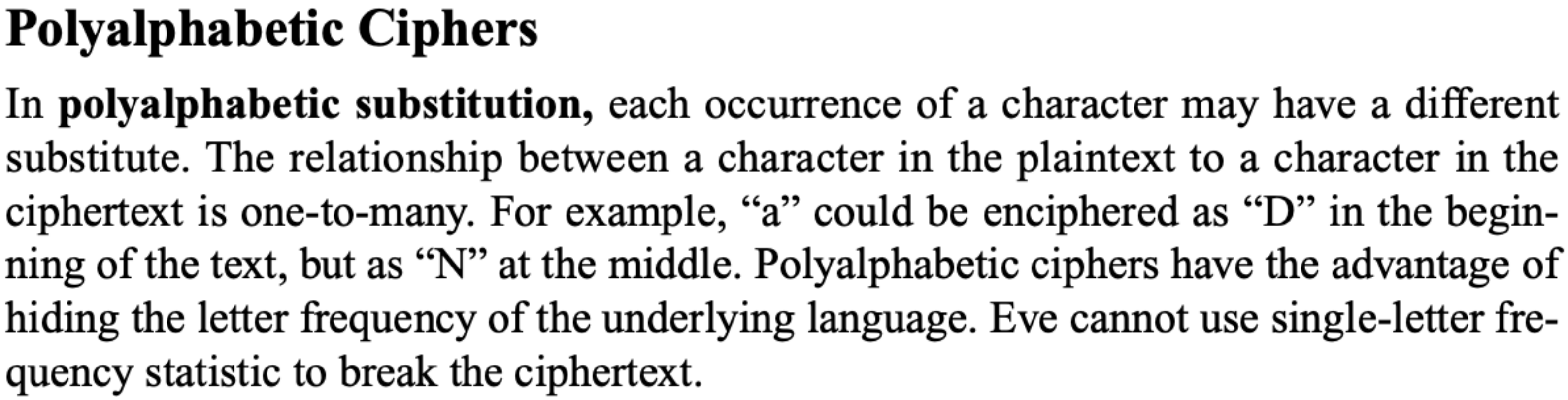
Eve has intercepted the following ciphertext. Using a statistical attack, find the plaintext.

XLILSYWIMWRSAJSVWEPIJSVJSYVQMPPMSRHSPPEVWMXMWASVX-LQSVILYVVCFIJSVIXLIWIPPIVVIGIMZIWQSVISJJIVW

When Eve tabulates the frequency of letters in this ciphertext, she gets: I =14, V =13, S =12, and so on. The most common character is I with 14 occurrences. This shows that character I in the ciphertext probably corresponds to the character e in plaintext. This means key = 4. Eve deciphers the text to get

the house is now for sale for four million dollars it is worth more hurry before the seller receives more offers

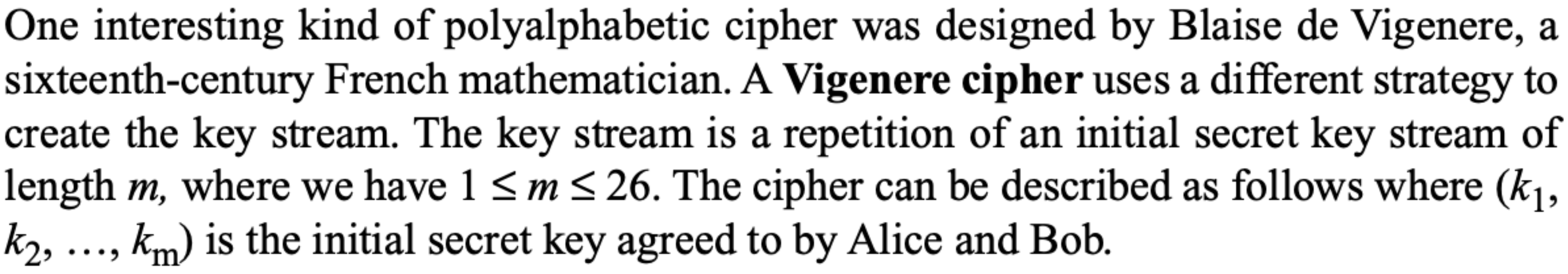
Polyalphabetic Ciphers



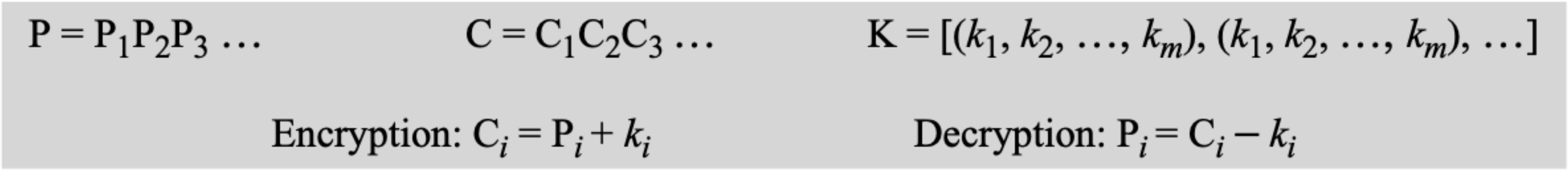
Polyalphabetic Ciphers

In **polyalphabetic substitution,** each occurrence of a character may have a different substitute. The relationship between a character in the plaintext to a character in the ciphertext is one-to-many. For example, “a” could be enciphered as “D” in the beginning of the text, but as “N” at the middle. Polyalphabetic ciphers have the advantage of hiding the letter frequency of the underlying language. Eve cannot use single-letter frequency statistic to break the ciphertext.

Vigenère Cipher



One interesting kind of polyalphabetic cipher was designed by Blaise de Vigenere, a sixteenth-century French mathematician. A **Vigenere cipher** uses a different strategy to create the key stream. The key stream is a repetition of an initial secret key stream of length *m,* where we have $1 \leq m \leq 26$. The cipher can be described as follows where (k_1, k_2, …, k_m) is the initial secret key agreed to by Alice and Bob.



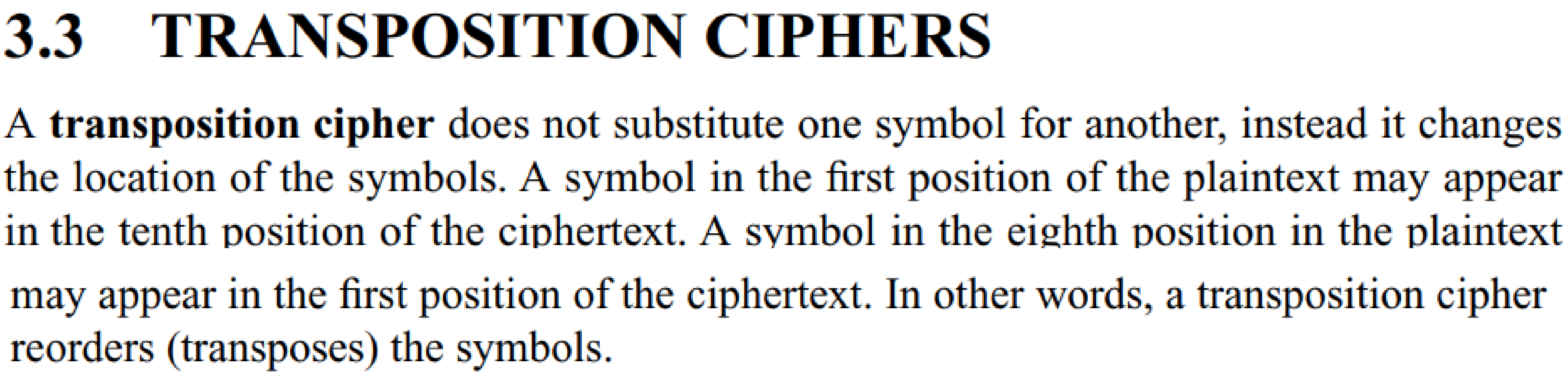
$P = P_1P_2P_3 \ldots$ $C = C_1C_2C_3 \ldots$ $K = [(k_1, k_2, \ldots, k_m), (k_1, k_2, \ldots, k_m), \ldots]$

Encryption: $C_i = P_i + k_i$ Decryption: $P_i = C_i - k_i$

Let us see how we can encrypt the message “She is listening” using the 6-character keyword “*PASCAL*”. The initial key stream is (15, 0, 18, 2, 0, 11). The key stream is the repetition of this initial key stream (as many times as needed).

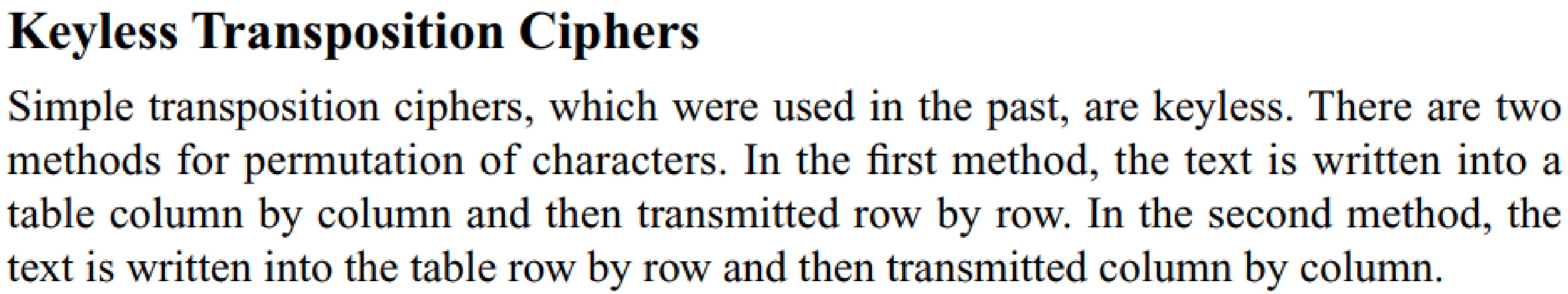
Plaintext:	s	h	e	i	s	l	i	s	t	e	n	i	n	g
P’s values:	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Key stream:	***15***	***00***	***18***	***02***	***00***	***11***	***15***	***00***	***18***	***02***	***00***	***11***	***15***	***00***
C’s values:	**07**	**07**	**22**	**10**	**18**	**22**	**23**	**18**	**11**	**6**	**13**	**19**	**02**	**06**
Ciphertext:	**H**	**H**	**W**	**K**	**S**	**W**	**X**	**S**	**L**	**G**	**N**	**T**	**C**	**G**

Transposition Ciphers



3.3 TRANSPOSITION CIPHERS

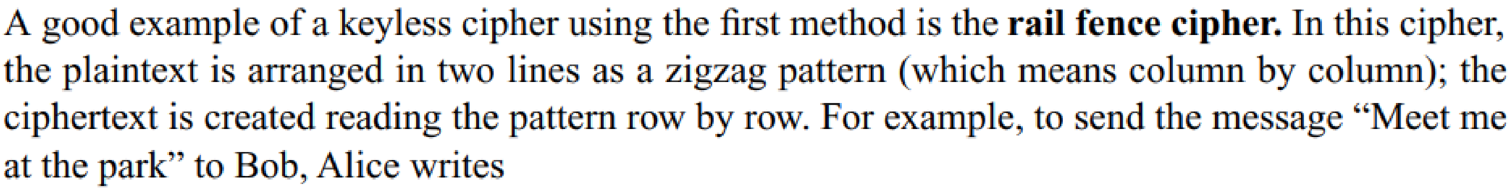
A **transposition cipher** does not substitute one symbol for another, instead it changes the location of the symbols. A symbol in the first position of the plaintext may appear in the tenth position of the ciphertext. A symbol in the eighth position in the plaintext may appear in the first position of the ciphertext. In other words, a transposition cipher reorders (transposes) the symbols.



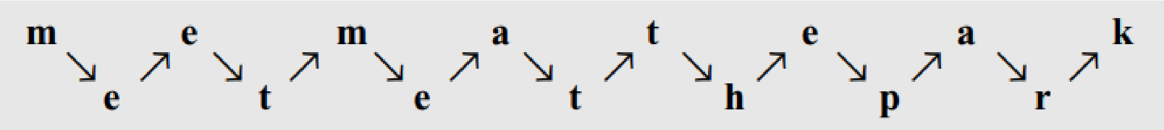
Keyless Transposition Ciphers

Simple transposition ciphers, which were used in the past, are keyless. There are two methods for permutation of characters. In the first method, the text is written into a table column by column and then transmitted row by row. In the second method, the text is written into the table row by row and then transmitted column by column.

Keyless Transposition Cipher



A good example of a keyless cipher using the first method is the **rail fence cipher.** In this cipher, the plaintext is arranged in two lines as a zigzag pattern (which means column by column); the ciphertext is created reading the pattern row by row. For example, to send the message “Meet me at the park” to Bob, Alice writes



ciphertext “MEMATEAKETETHPR”

Keyless Transposition Cipher

Alice and Bob can agree on the number of columns and use the second method. Alice writes the same plaintext, row by row, in a table of four columns.

m	**e**	**e**	**t**
m	**e**	**a**	**t**
t	**h**	**e**	**p**
a	**r**	**k**	

the ciphertext “MMTAEEHREAEKTTP”

Key Transposition Cipher

Alice needs to send the message “Enemy attacks tonight” to Bob. Alice and Bob have agreed to divide the text into groups of five characters and then permute the characters in each group. The following shows the grouping after adding a bogus character at the end to make the last group the same size as the others.

e n e m y	a t t a c	k s t o n	i g h t z

The key used for encryption and decryption is a permutation key, which shows how the character are permuted. For this message, assume that Alice and Bob used the following key:

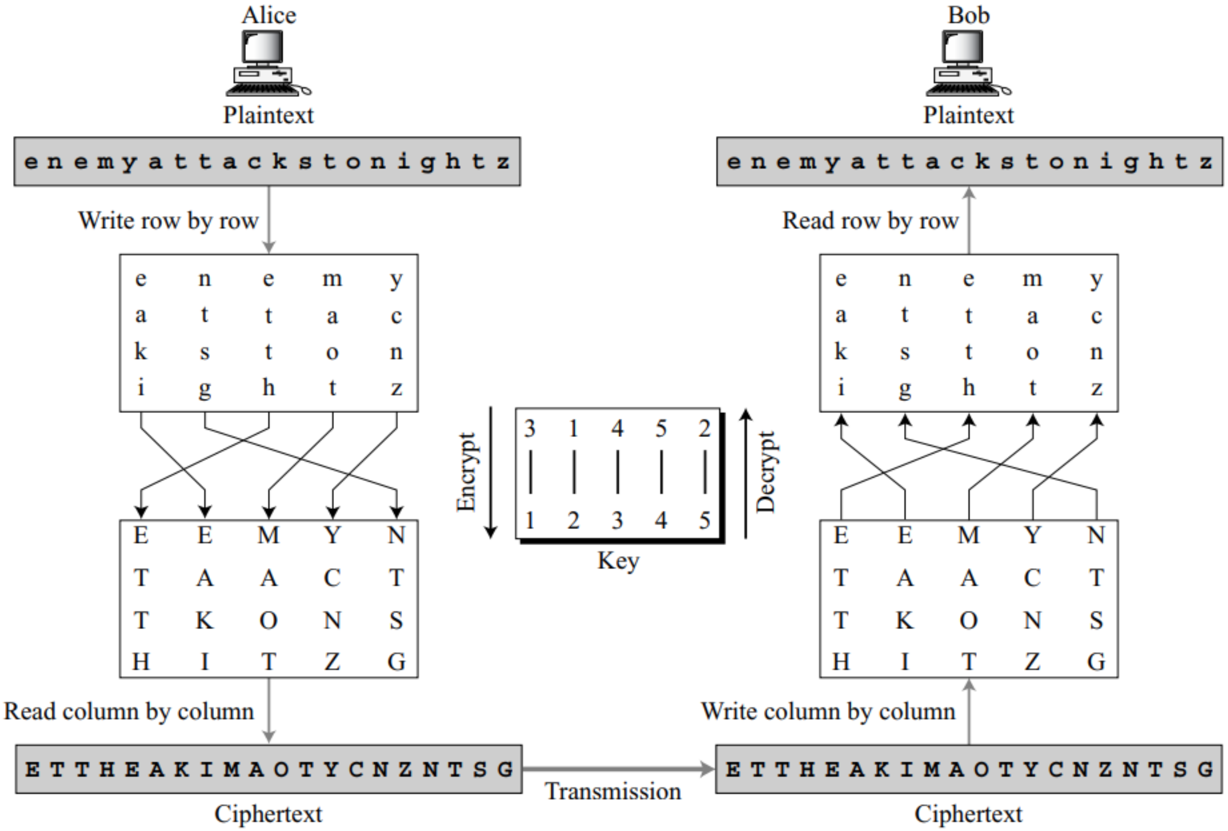
Encryption ↓	3	1	4	5	2	↑ Decryption
	1	2	3	4	5	

The third character in the plaintext block becomes the first character in the ciphertext block; the first character in the plaintext block becomes the second character in the ciphertext block; and so on. The permutation yields

E E M Y N	**T A A C T**	**T K O N S**	**H I T Z G**

Alice sends the ciphertext “EEMYNTAACTTKONSHITZG” to Bob. Bob divides the ciphertext into 5-character groups and, using the key in the reverse order, finds the plaintext.

Combination Transposition Cipher



Alice

Plaintext

enemyattackstonightz

Write row by row

e	n	e	m	y
a	t	t	a	c
k	s	t	o	n
i	g	h	t	z

E	E	M	Y	N
T	A	A	C	T
T	K	O	N	S
H	I	T	Z	G

Read column by column

ETTHEAKIMAOTYCNZNTSG

Ciphertext

Encrypt

3	1	4	5	2
1	2	3	4	5

Key

Decrypt

Transmission

Bob

Plaintext

enemyattackstonightz

Read row by row

e	n	e	m	y
a	t	t	a	c
k	s	t	o	n
i	g	h	t	z

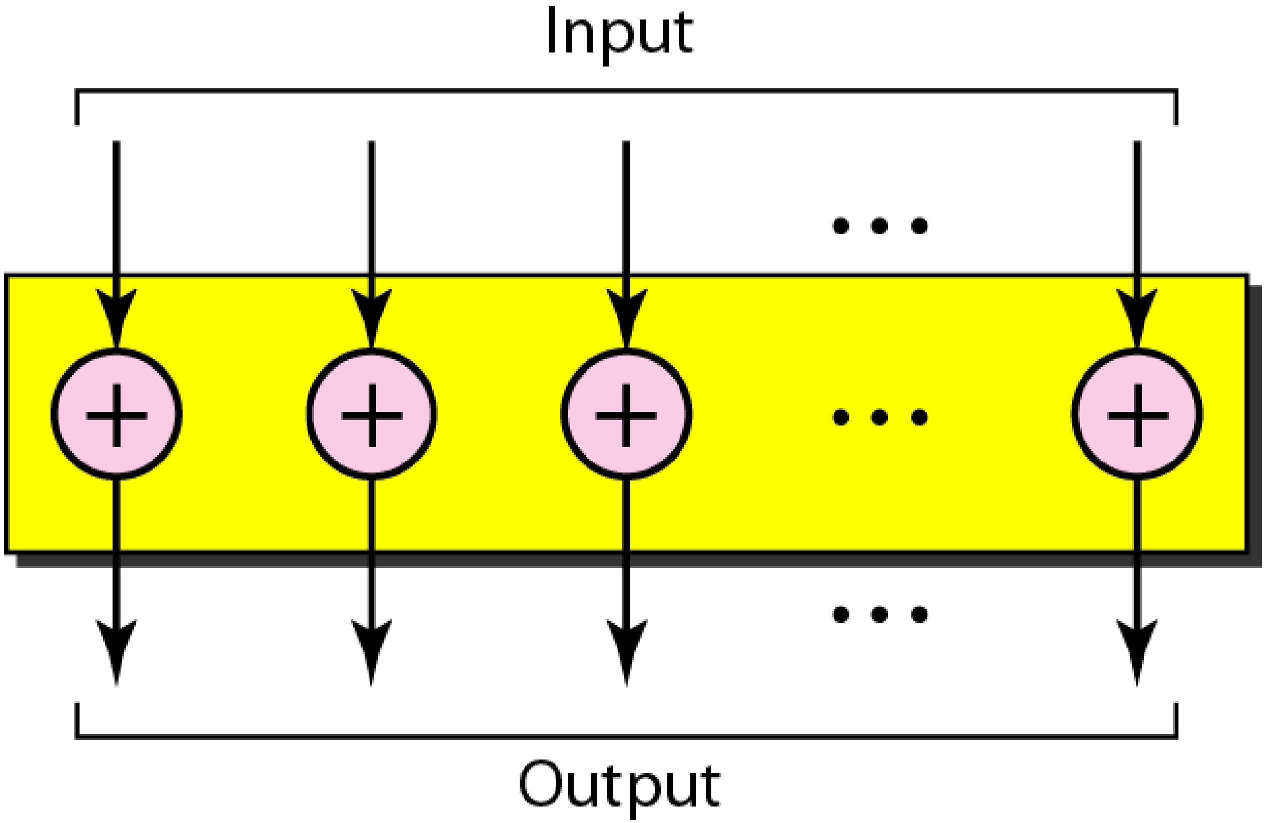
E	E	M	Y	N
T	A	A	C	T
T	K	O	N	S
H	I	T	Z	G

Write column by column

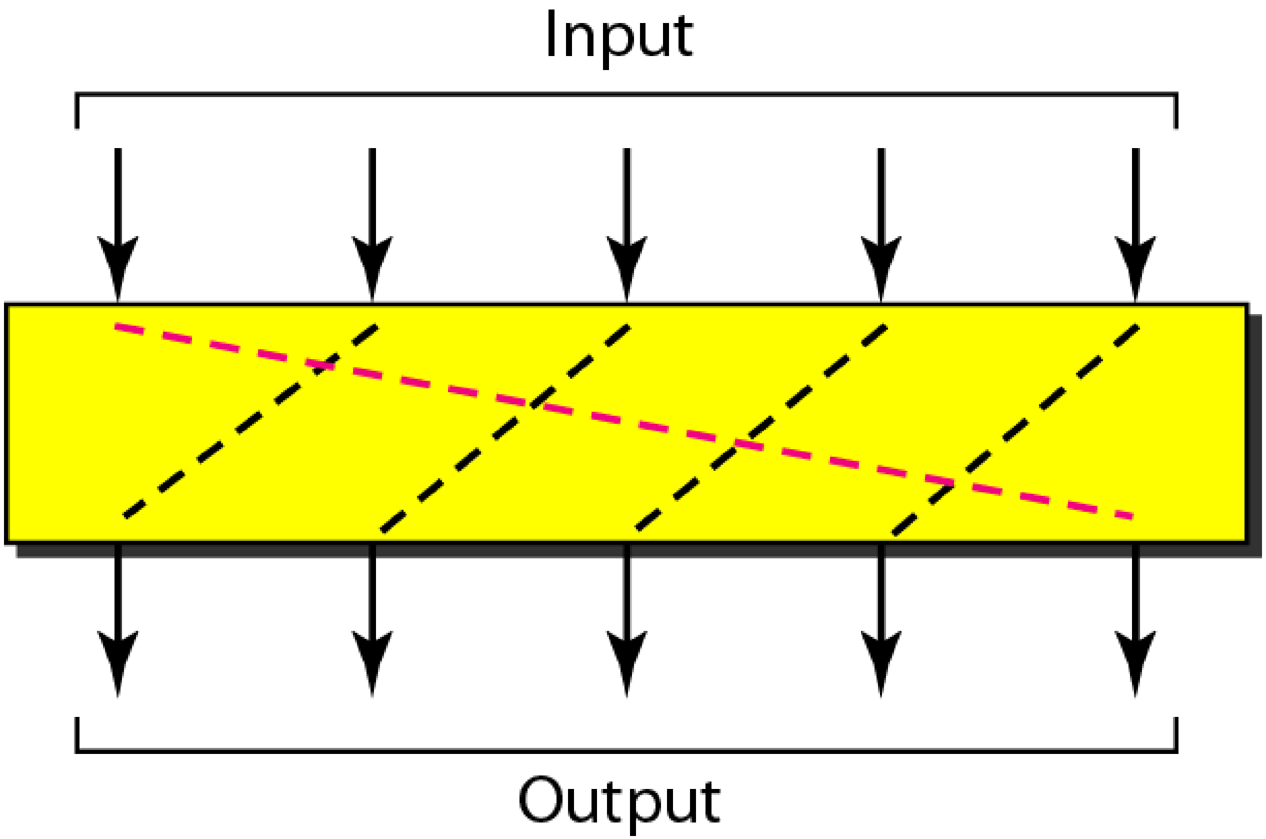
ETTHEAKIMAOTYCNZNTSG

Ciphertext

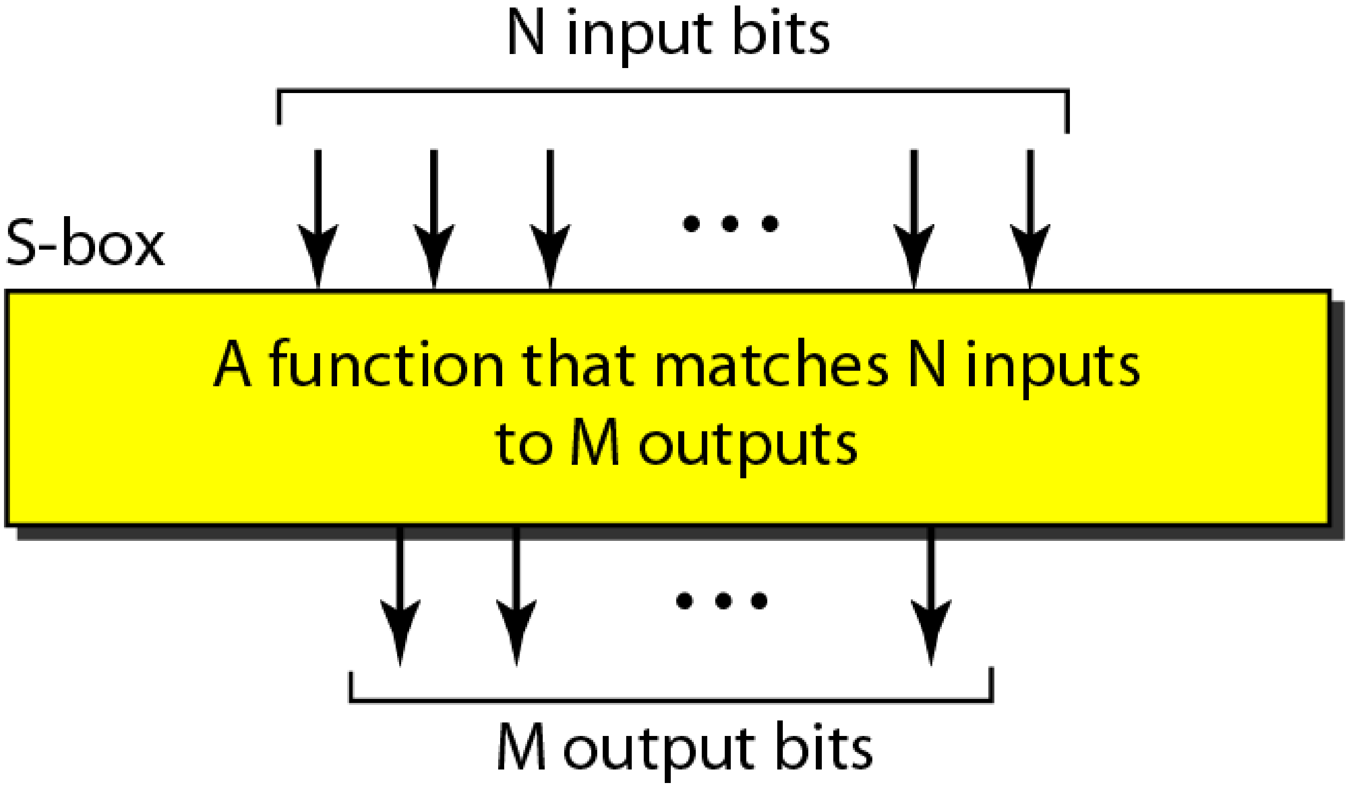
XOR cipher



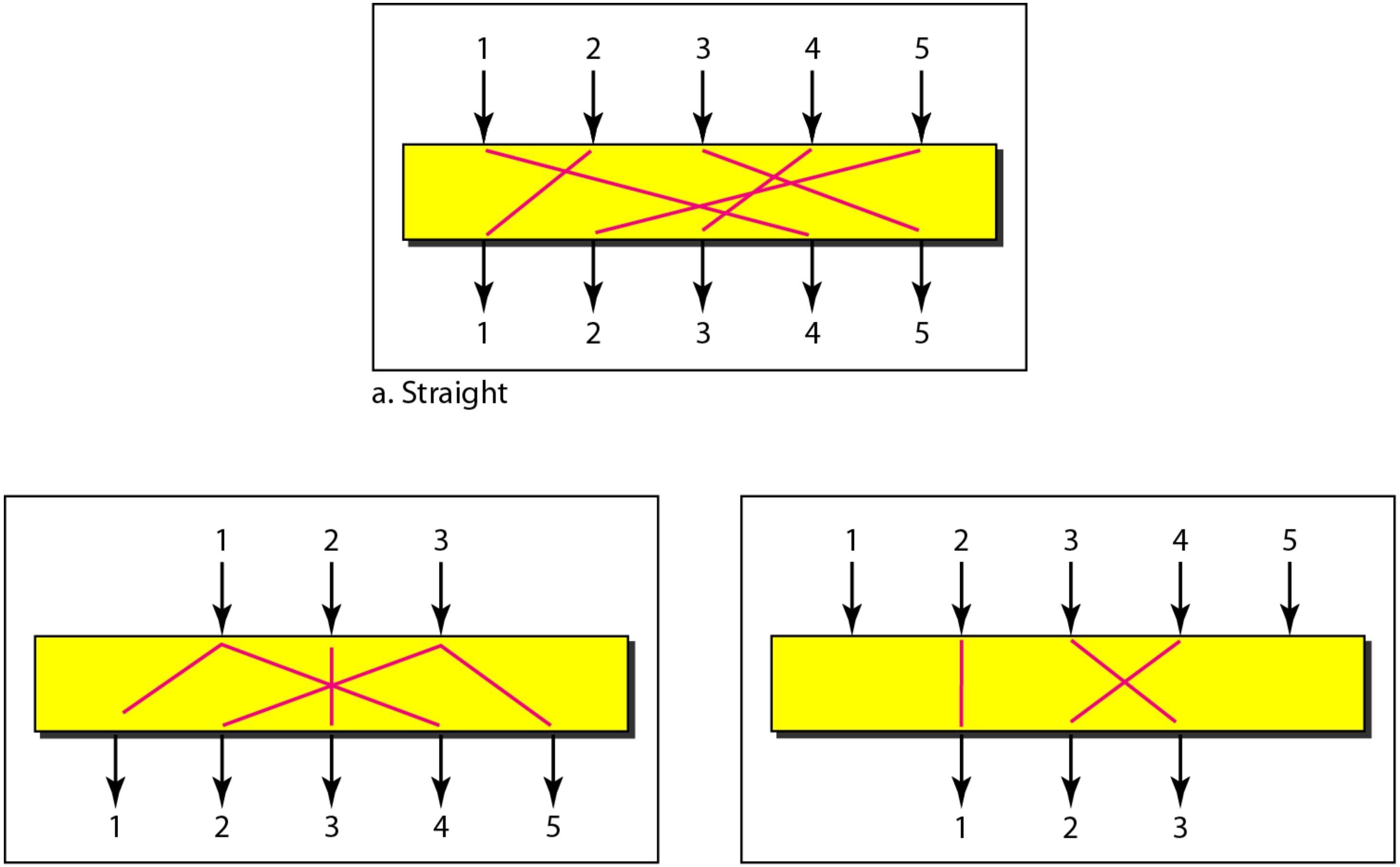
Rotation cipher



S-box (substitution box)



P-boxes (permutation box): straight, expansion, and compression

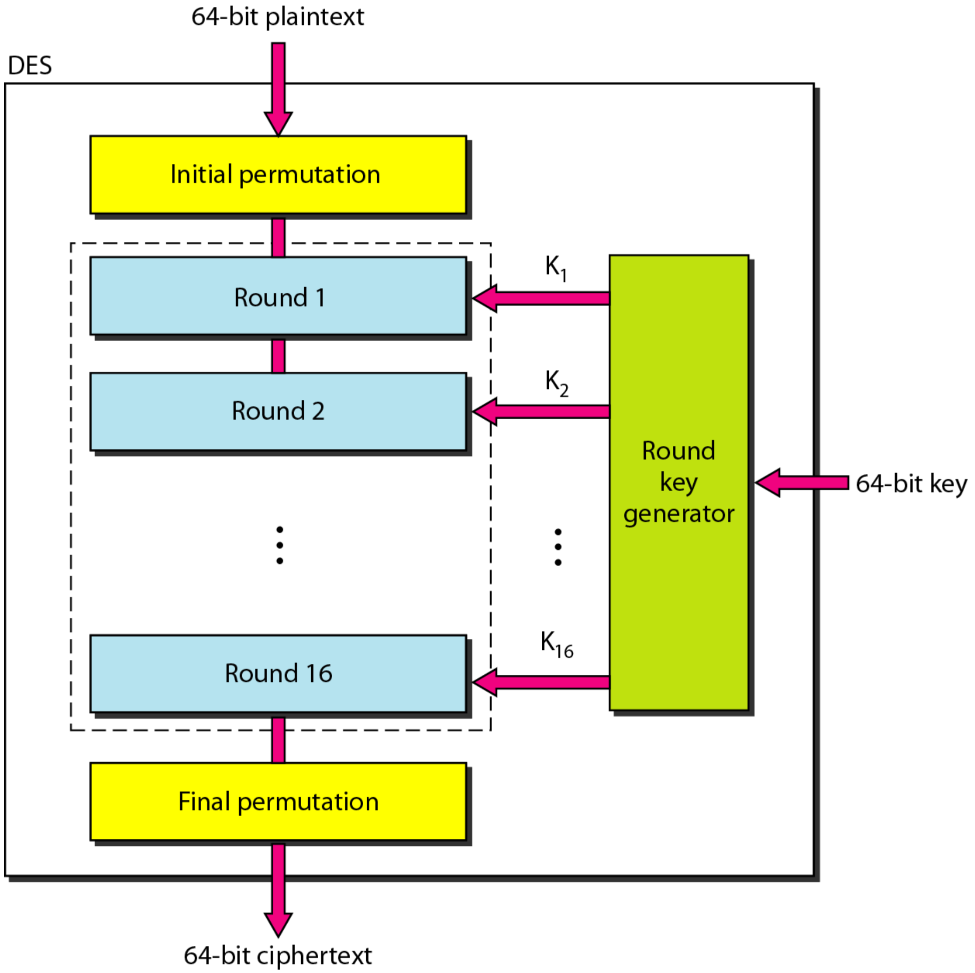


a. Straight

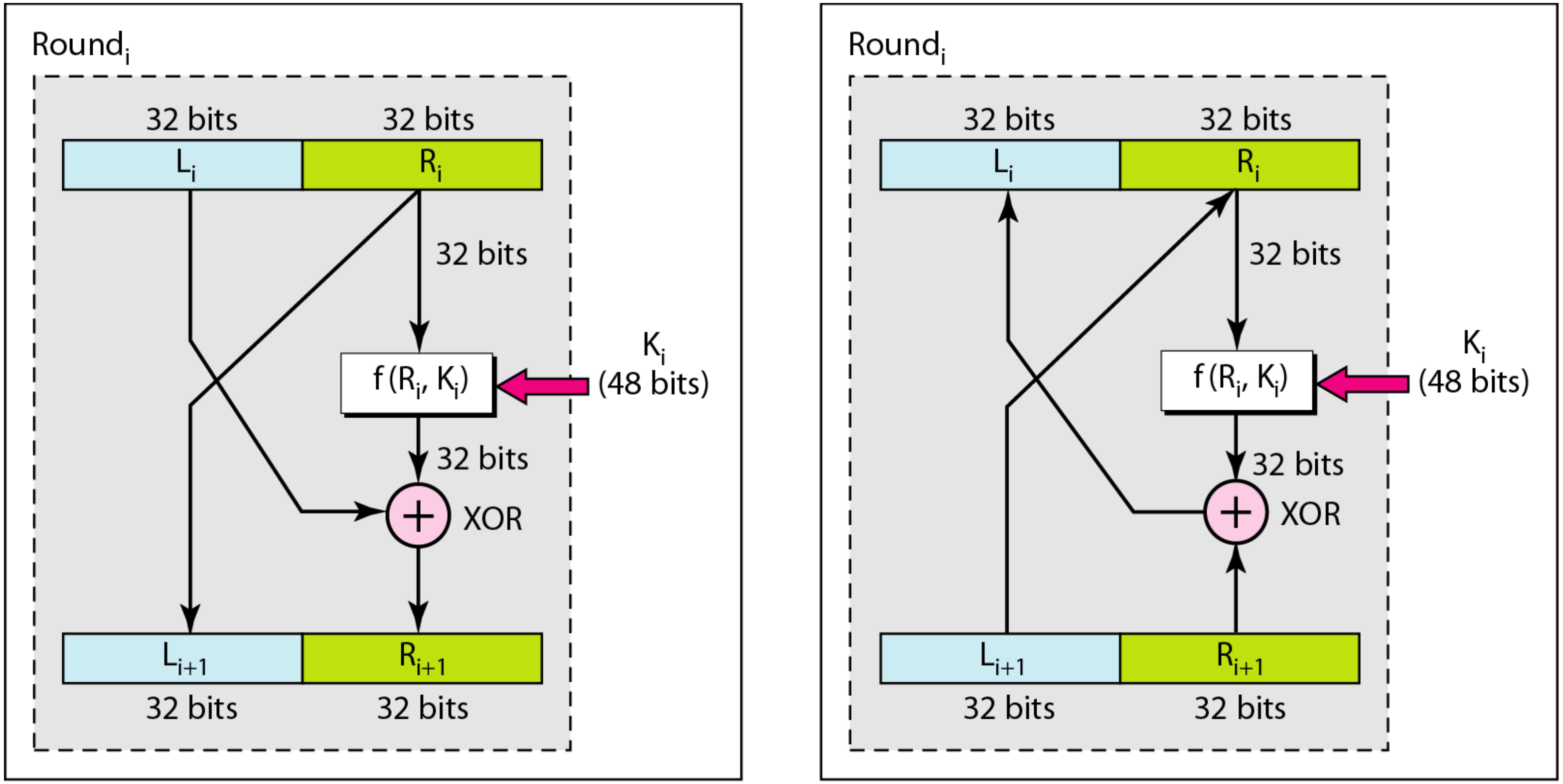
b. Expansion

c. Compression

DES (Data Encryption Standard)



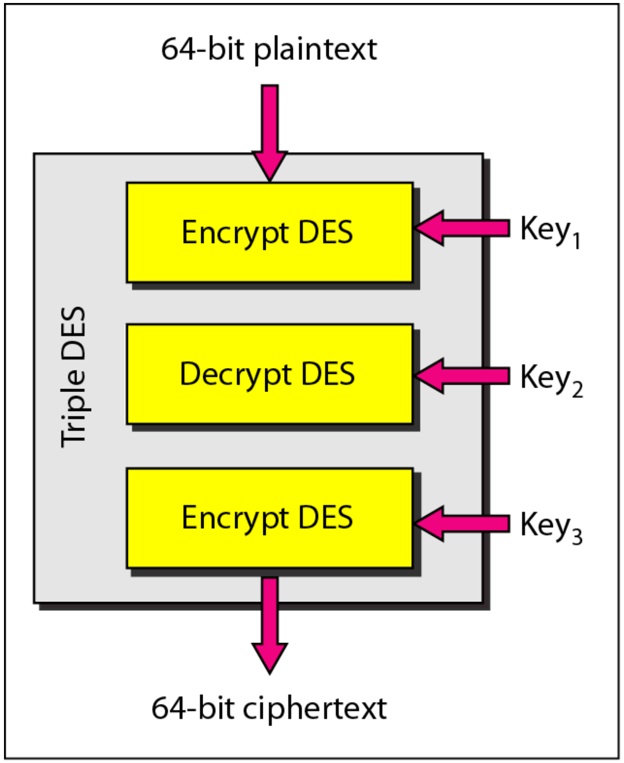
One round in DES ciphers



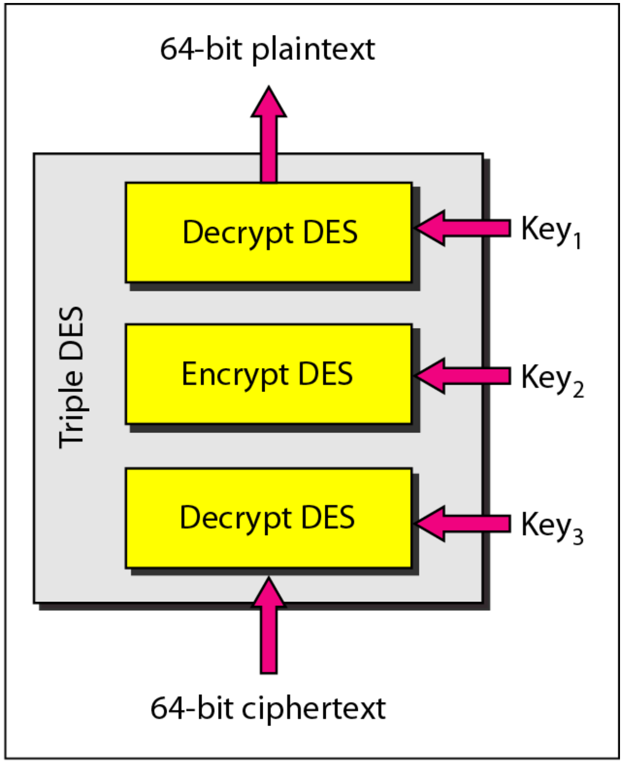
a. Encryption round

b. Decryption round

Triple DES (to resolve the short key issue for DES)



a. Encryption Triple DES



b. Decryption Triple DES

Asymmetric Encryption

- Asymmetric encryption has some core differences from symmetric encryption.
- The first that you can immediately notice is that in asymmetric encryption there are two keys: one public key to encrypt and a private key to decrypt.
- From this concept, one of the names of asymmetric encryption is derived: public key encryption.
- This approach does simplify greatly the key exchange and key management. For the key management, you only need a pair of keys (private/public) for each person. So if you have 20 people that have to communicate between themselves, you'll only need 20 pairs of keys.

Figure 30.4 *Asymmetric-key cryptography*

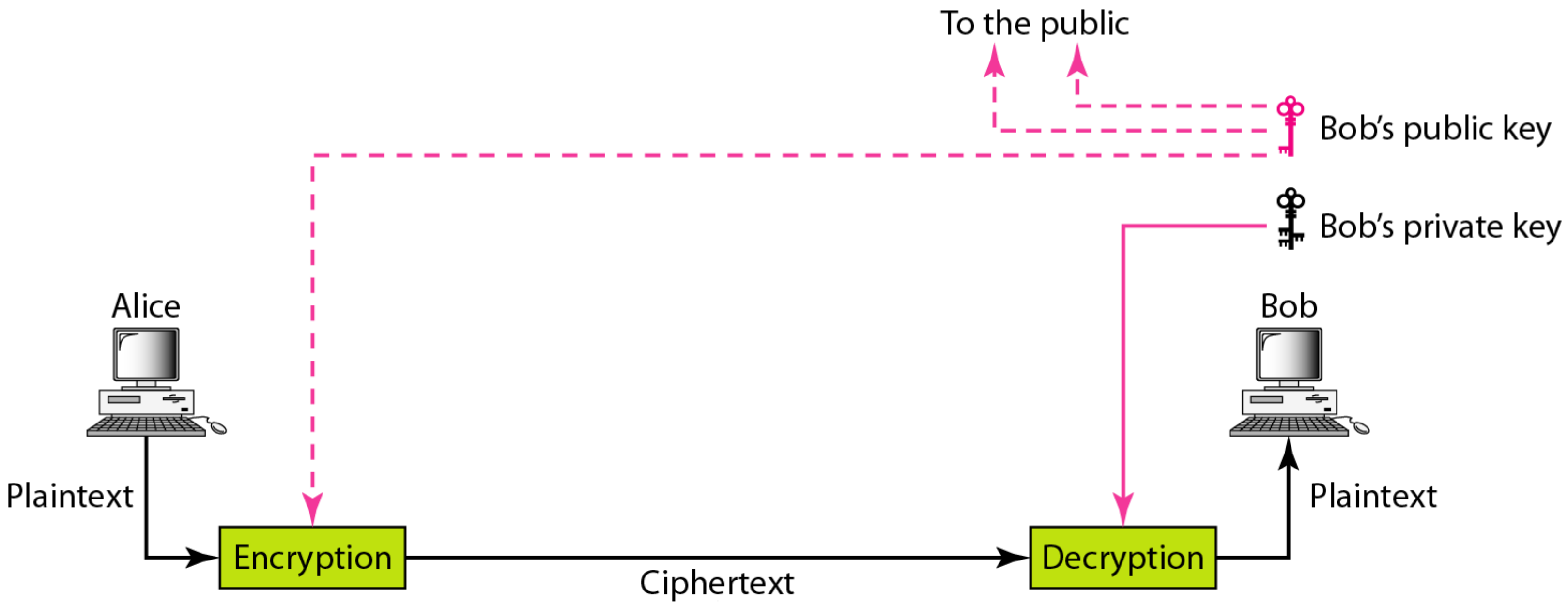
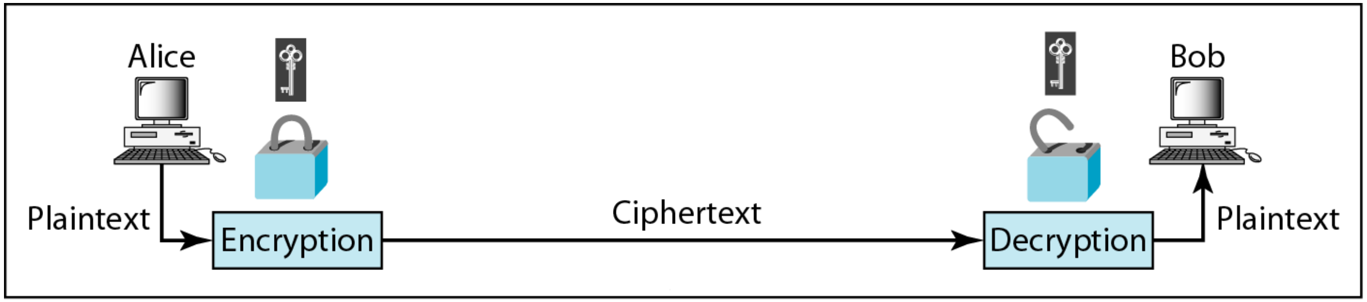
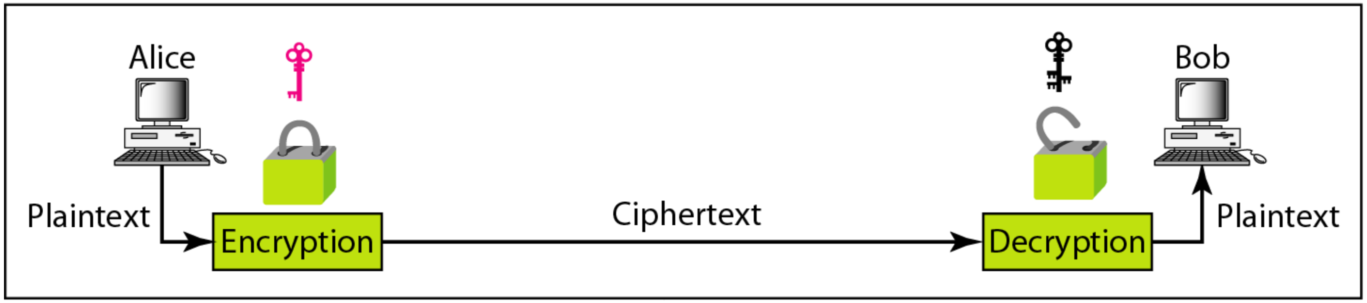


Figure 30.6 *Comparison between two categories of cryptography*



a. Symmetric-key cryptography



b. Asymmetric-key cryptography

RSA: Choosing keys

1. Choose two large prime numbers *p, q.* (e.g., 1024 bits each)

2. Compute $n = pq$, $\Phi = (p-1)(q-1)$

3. Choose e (with $e<n$) that has no common factors with Φ. (e, Φ are "relatively prime").

4. Choose d such that $ed-1$ is exactly divisible by Φ. (in other words: $ed \bmod \Phi = 1$).

5. *Public* key is $(\underbrace{n,e}_{K_B^+})$. *Private* key is $(\underbrace{n,d}_{K_B^-})$.

RSA: Encryption, decryption

0. Given (n,e) and (n,d) as computed above

1. To encrypt bit pattern, m, compute

$$c = m^e \bmod n$$

(i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute

$$m = c^d \bmod n$$

(i.e., remainder when c^d is divided by n)

Magic happens!

$$\boxed{m = (\underbrace{m^e \bmod n}_{c})^d \bmod n}$$

RSA example:

Bob chooses *p*=5, *q*=7. Then *n*=35, Φ =24.

e=5 (so *e*, Φ relatively prime).

d=29 (so *ed-1* exactly divisible by Φ).

encrypt:

letter	m	m^e	$c = m^e \bmod n$
l	12	1524832	17

decrypt:

c	c^d	$m = c^d \bmod n$	letter
17	481968572106750915091411825223071697	12	l

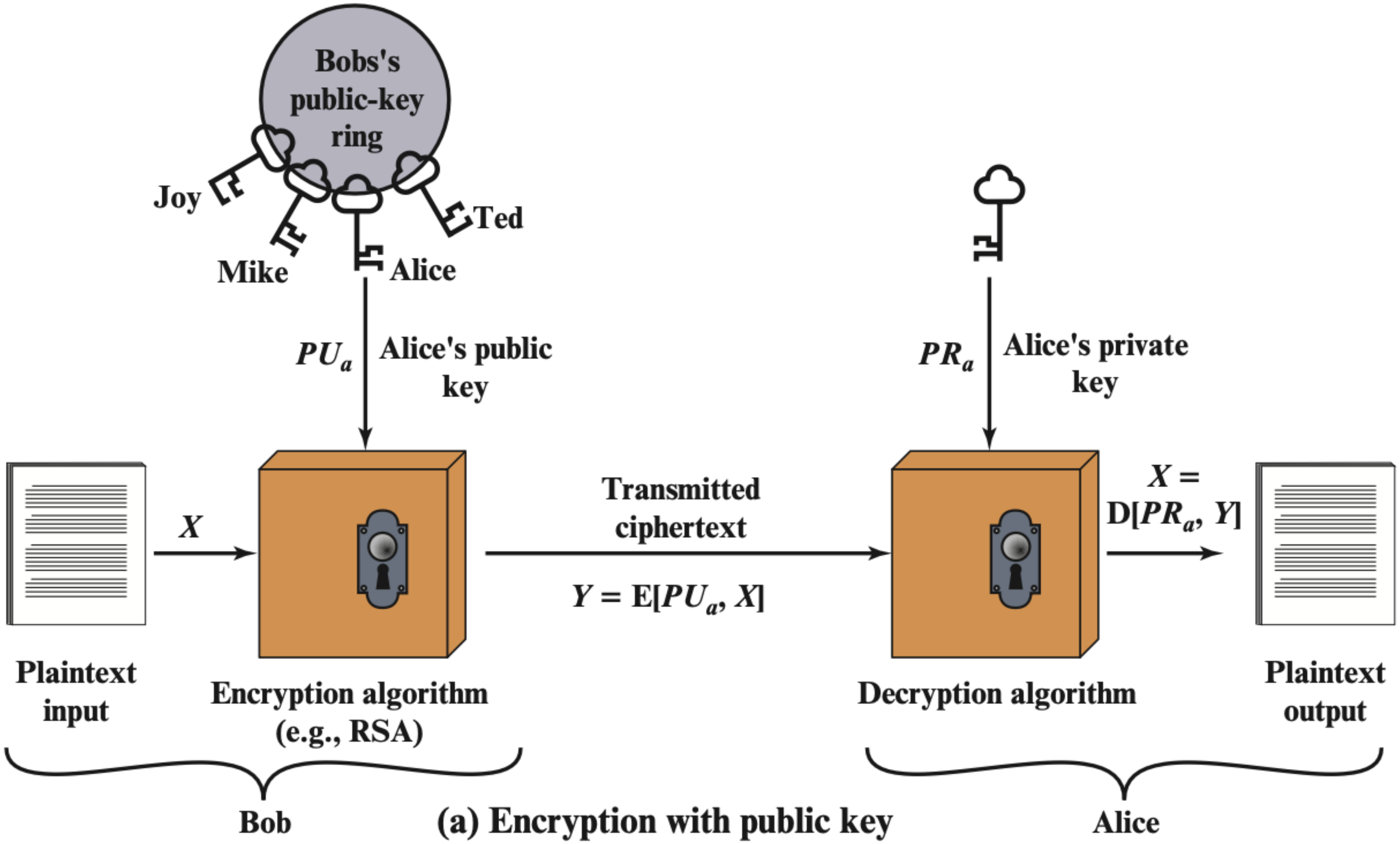
Computational very extensive

Note

In RSA, *e* and *n* are announced to the public; *d* and Φ are kept secret.

Public cryptography is very computational expensive.

Public Key Cryptography for Encryption/Decryption



(a) Encryption with public key

Public Key Cryptography for Digital Signatures

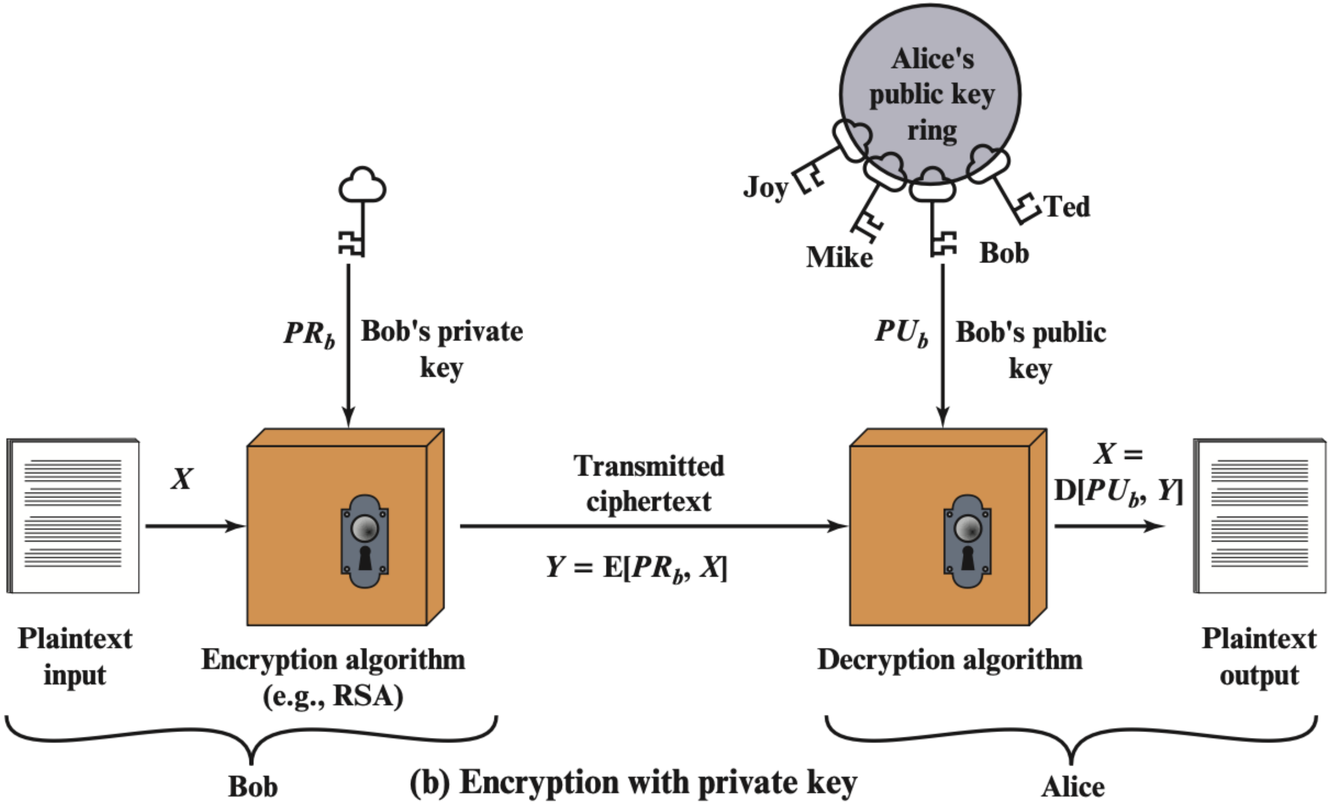


Figure 9.1 Public-Key Cryptography

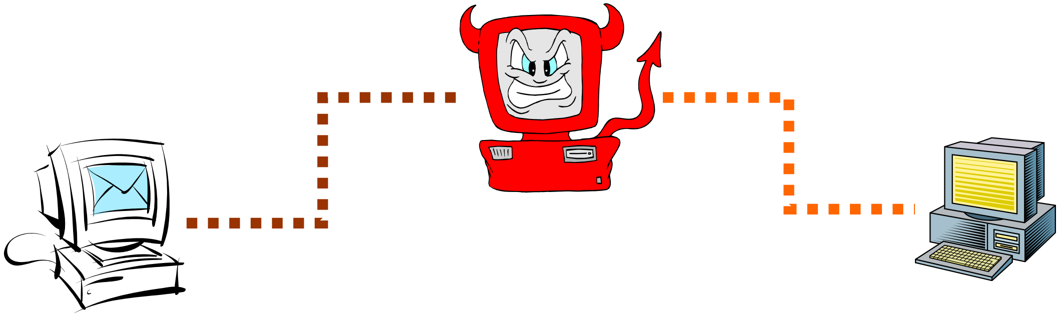
Demo: https://learn.pkiindia.in/asymmetric.html

comparison

Domain	Symmetric	Asymmetric
Able to grant	Confidential	Confidential, Offering Integrity, Authentication, and Non-repudiation
Needed keys	A single shared key	A public Key and A private Key
Key Exchange	Complex	Simple
Scalability	Not scalable, keys increase exponentially	Scalable
Key Size	small	Big
Implementation Speed	Fast	Slow
Best for	Bulk Data	Small amount of data, key exchange, digital envelopes, digital signatures and digital certificate

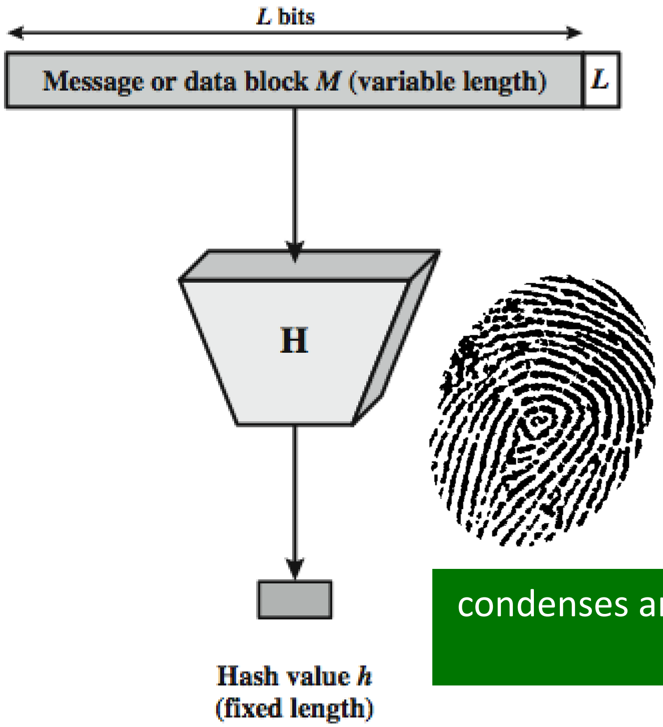
Hash Functions

Data Integrity and Source Authentication



- Encryption does not protect data from modification by another party.
- Need a way to ensure that data arrives at destination in its original form as sent by the sender and it is coming from an authenticated source.

Hash Function



- The hash value represents concisely the longer message
  - may called the *message digest*

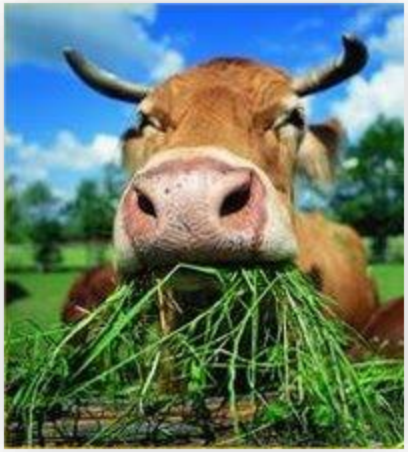
- A message digest is as a ``digital fingerprint'' of the original document

condenses arbitrary message to fixed size

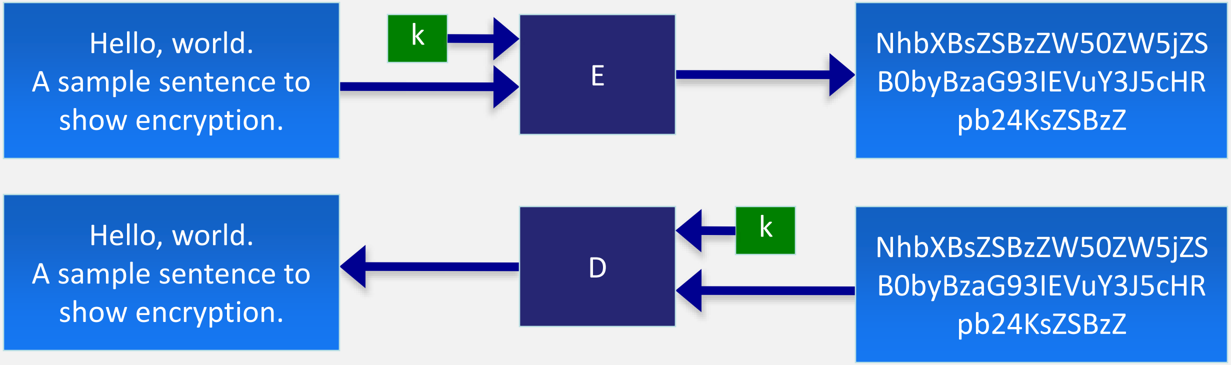
$h = H(M)$

Chewing functions

- Hashing function as “chewing” or “digest” function



Hashing V.S. Encryption



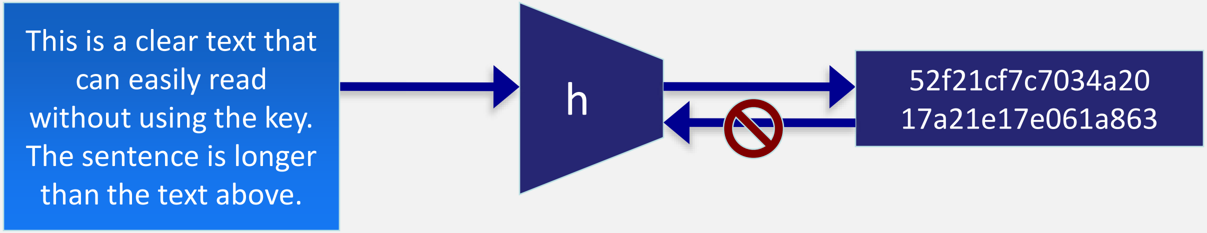
Hello, world.
A sample sentence to
show encryption.

NhbXBsZSBzZW50ZW5jZS
B0byBzaG93IEVuY3J5cHR
pb24KsZSBzZ

Hello, world.
A sample sentence to
show encryption.

NhbXBsZSBzZW50ZW5jZS
B0byBzaG93IEVuY3J5cHR
pb24KsZSBzZ

- Encryption is two way, and requires a key to encrypt/decrypt

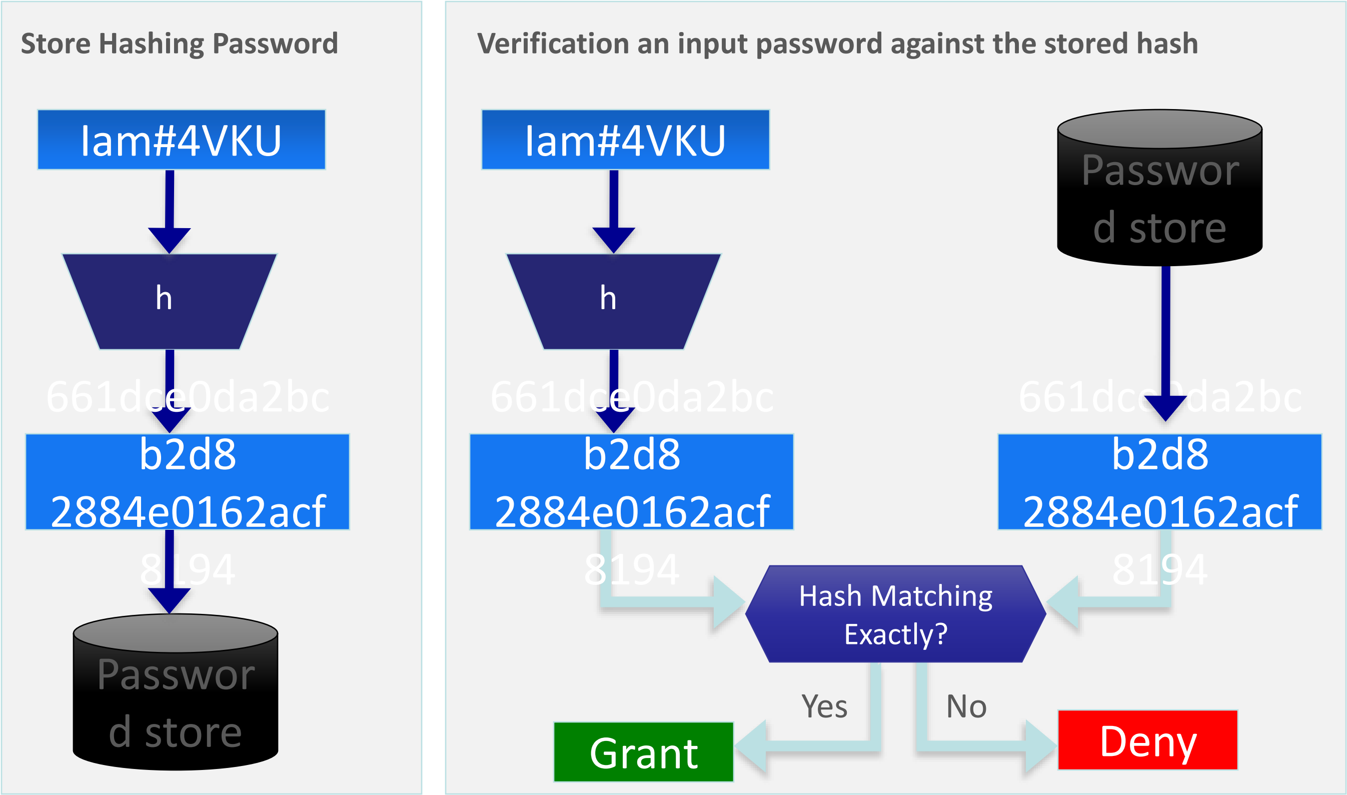


This is a clear text that
can easily read
without using the key.
The sentence is longer
than the text above.

52f21cf7c7034a20
17a21e17e061a863

– Hashing is one-way. There is no 'de-hashing’

Password Verification



Store Hashing Password

Iam#4VKU

h

661dce0da2bc
b2d8
2884e0162acf
8194

Password store

Verification an input password against the stored hash

Iam#4VKU

h

661dce0da2bc
b2d8
2884e0162acf
8194

Password store

661dce0da2bc
b2d8
2884e0162acf
8194

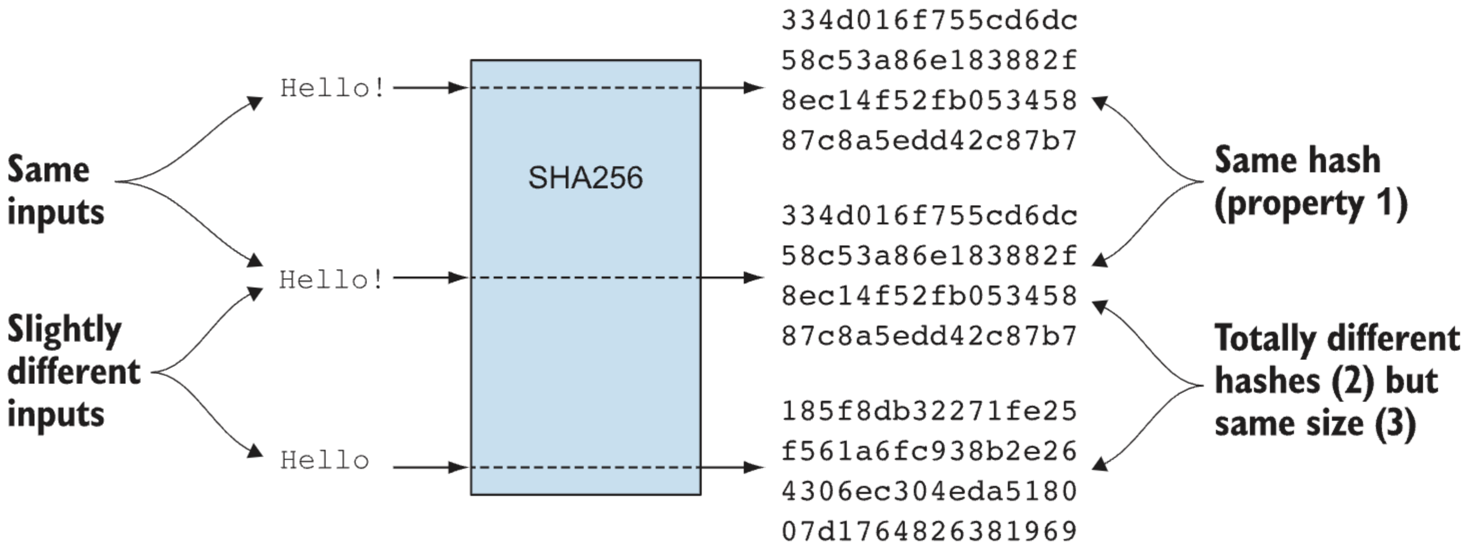
Hash Matching Exactly?

Yes

No

Grant

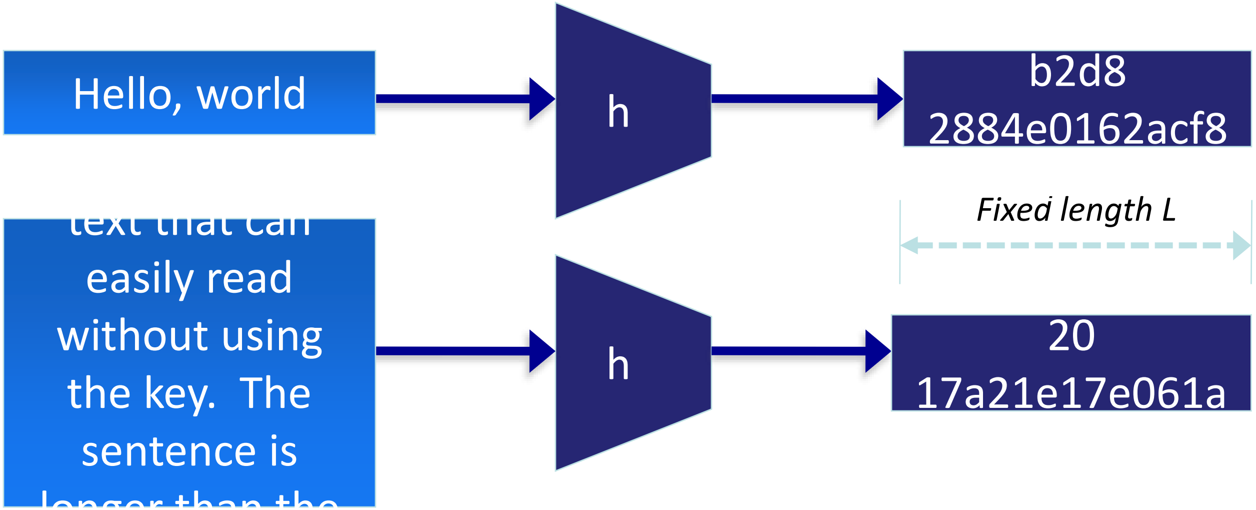
Deny


Same inputs
Slightly different inputs
Hello!
Hello!
Hello
SHA256
334d016f755cd6dc
58c53a86e183882f
8ec14f52fb053458
87c8a5edd42c87b7
334d016f755cd6dc
58c53a86e183882f
8ec14f52fb053458
87c8a5edd42c87b7
185f8db32271fe25
f561a6fc938b2e26
4306ec304eda5180
07d1764826381969
Same hash (property 1)
Totally different hashes (2) but same size (3)

Hash Function Properties

- Arbitrary-length message to fixed-length digest
- Preimage resistant (**One-way property**)
- Second preimage resistant (**Weak collision resistant**)
- Collision resistant (**Strong collision resistance**)

Properties : Fixed length

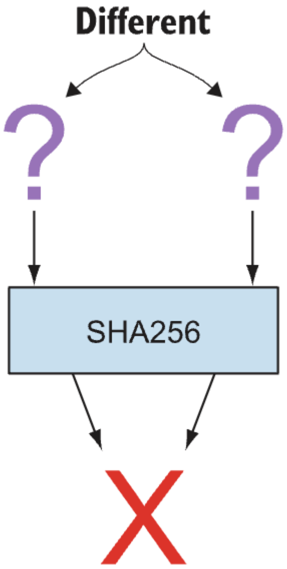


- Arbitrary-length message to fixed-length digest

Demo: https://www.fileformat.info/tool/hash.htm

Collision resistance

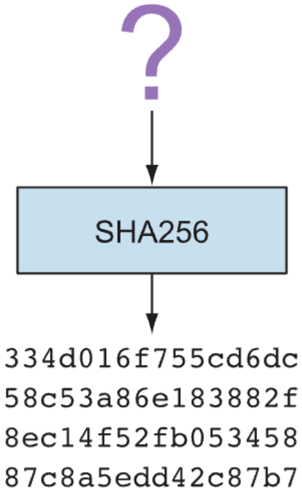
You have only the cryptographic hash function at hand. It’s hard to find two *different* inputs that *result in the same hash*.



Collision resistance

Pre-image resistance

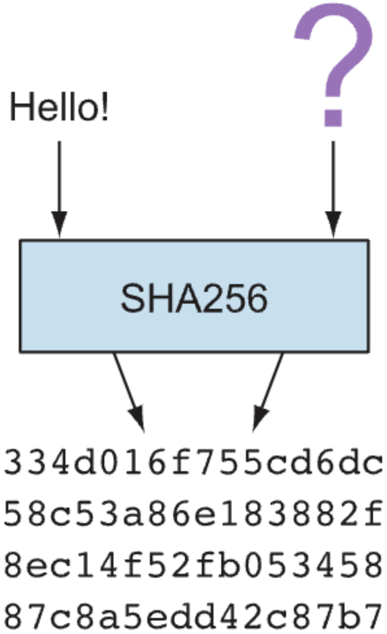
You have the hash function and a hash. It’s hard to find *a pre-image of that hash*.



Pre-image resistance

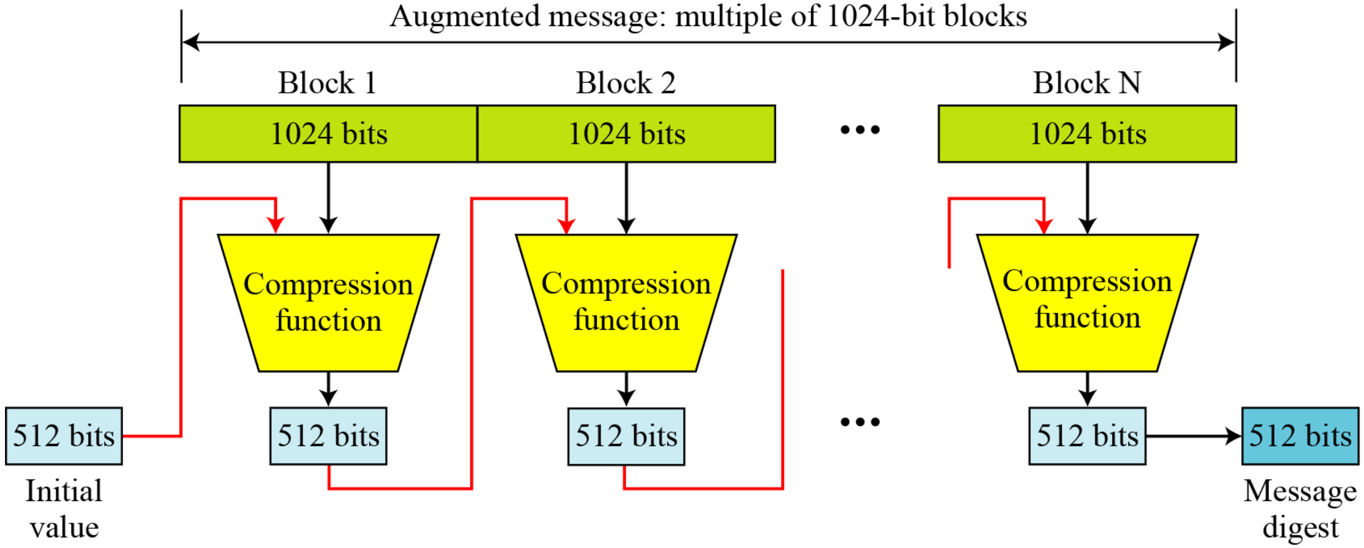
Second-pre-image resistance

You have the hash function and a pre-image (and thus the hash of that pre-image). It's hard to find *another pre-image with the same hash*.

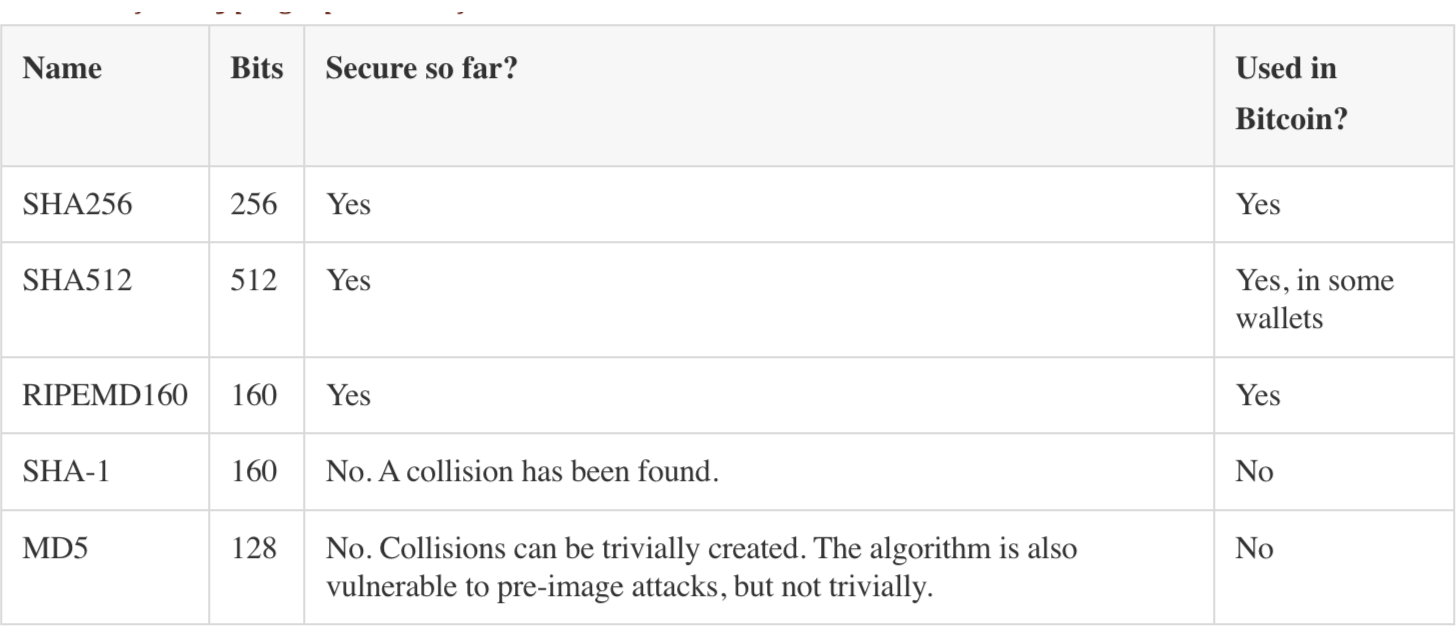


Second-pre-image resistance

SHA-512 Overview



Some well-known hash functions



Name	Bits	Secure so far?	Used in Bitcoin?
SHA256	256	Yes	Yes
SHA512	512	Yes	Yes, in some wallets
RIPEMD160	160	Yes	Yes
SHA-1	160	No. A collision has been found.	No
MD5	128	No. Collisions can be trivially created. The algorithm is also vulnerable to pre-image attacks, but not trivially.	No

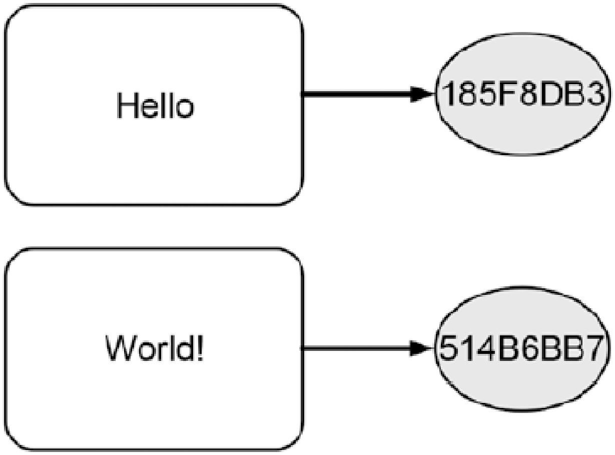
Patterns of Hashing Data

- Independent hashing
- Repeated hashing
- Combined hashing
- Sequential hashing
- Hierarchical hashing

Courtesy: Blockchain Basics: A Non-Technical Introduction in 25 Steps by Daniel Drescher

Types of Hashing

- Independent hashing



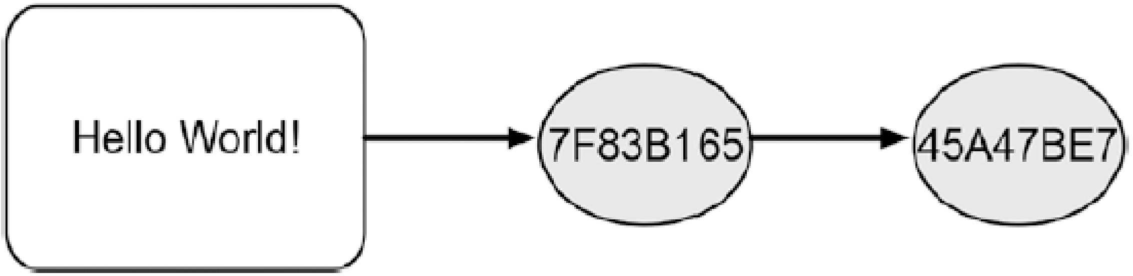
Hello

185F8DB3

World!

514B6BB7

- Repeated hashing



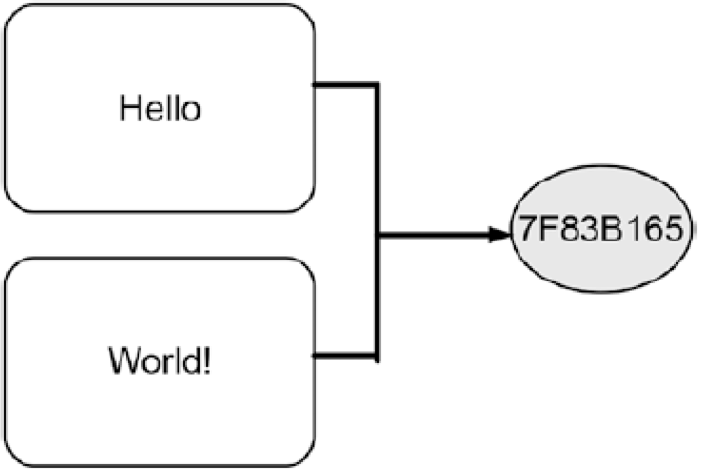
Hello World!

7F83B165

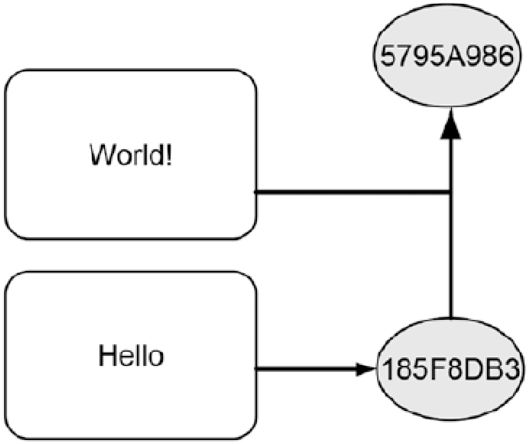
45A47BE7

Types of Hashing

- Combined hashing

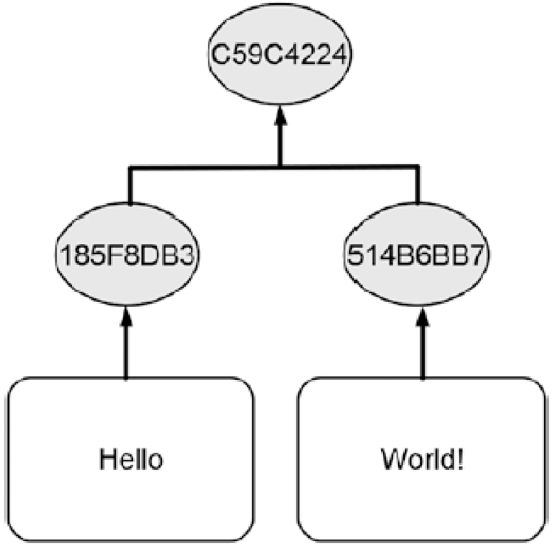


- Sequential hashing

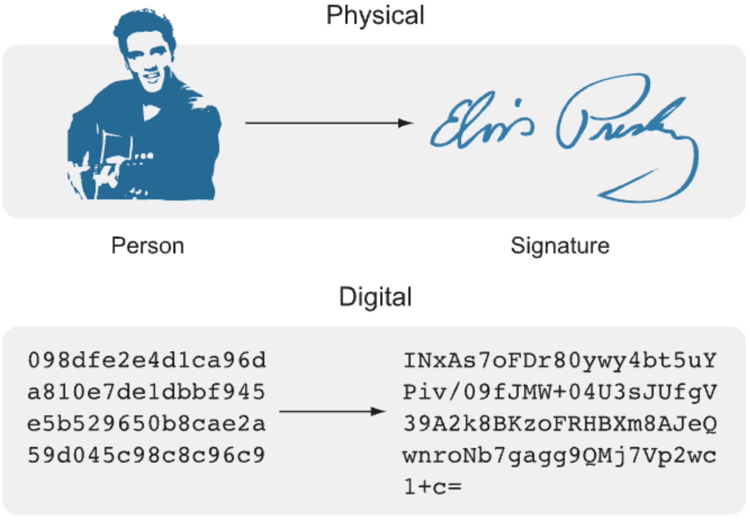


Types of Hashing

- Hierarchical hashing



Digital Signatures



Physical

Person

Signature

Digital

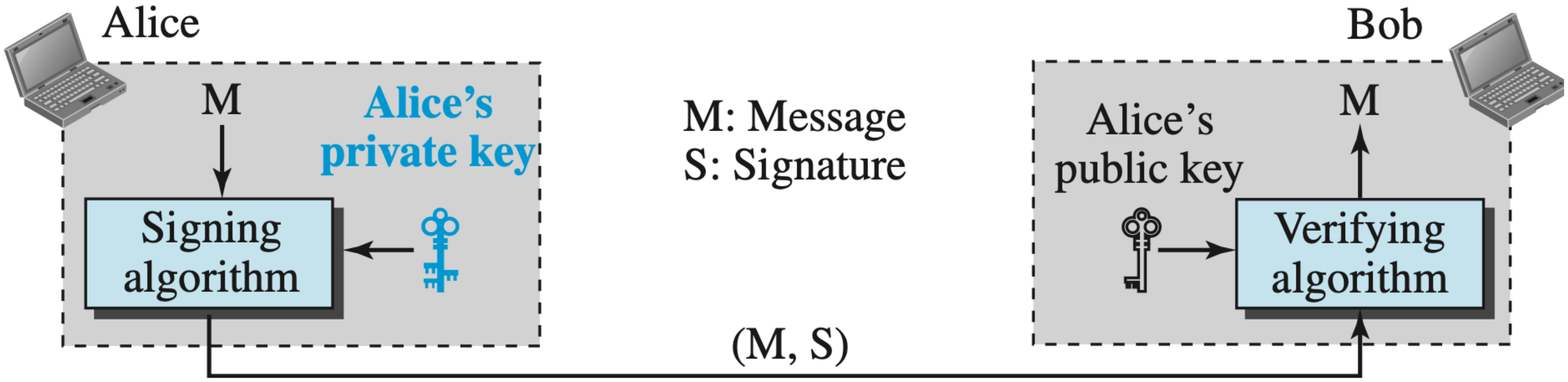
```
098dfe2e4d1ca96d
a810e7de1dbbf945
e5b529650b8cae2a
59d045c98c8c96c9
```

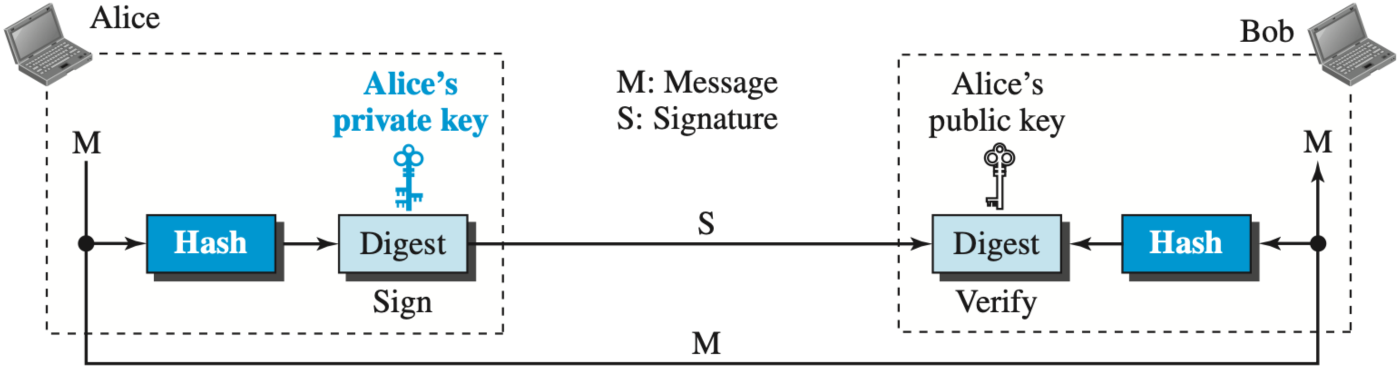
```
INxAs7oFDr80ywy4bt5uY
Piv/09fJMW+04U3sJUfgV
39A2k8BKzoFRHBXm8AJeQ
wnroNb7gagg9QMj7Vp2wc
1+c=
```

Private key

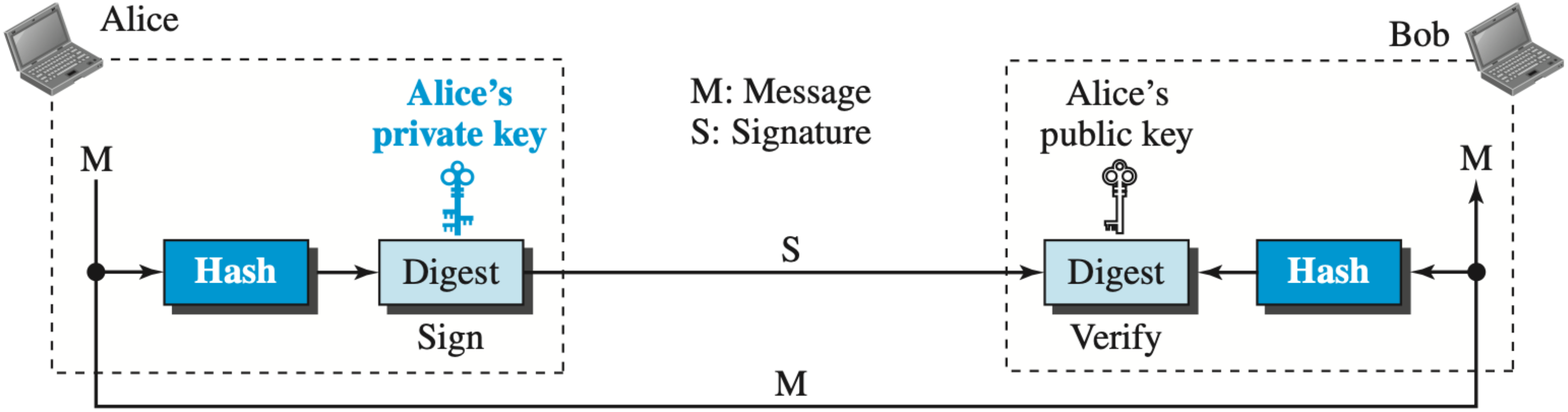
Digital signature

Digital Signature



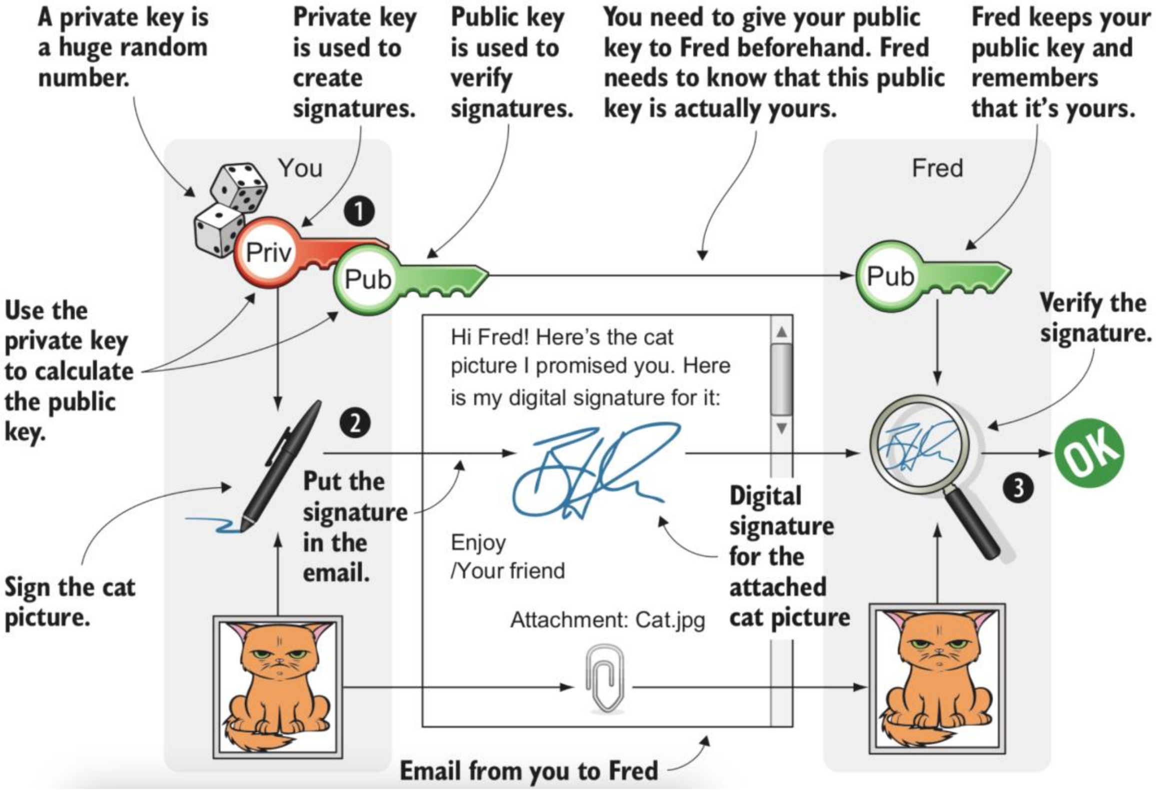


Digital Signature



- A digital signature needs a public-key system. The signer signs with her private key; the verifier verifies with the signer's public key.
- A cryptosystem uses the public and private keys of the receiver; a digital signature uses the private and public keys of the sender.

Digital Signatures



A private key is a huge random number.

Private key is used to create signatures.

Public key is used to verify signatures.

You need to give your public key to Fred beforehand. Fred needs to know that this public key is actually yours.

Fred keeps your public key and remembers that it's yours.

Use the private key to calculate the public key.

Verify the signature.

Hi Fred! Here's the cat picture I promised you. Here is my digital signature for it:

Enjoy
/Your friend

Attachment: Cat.jpg

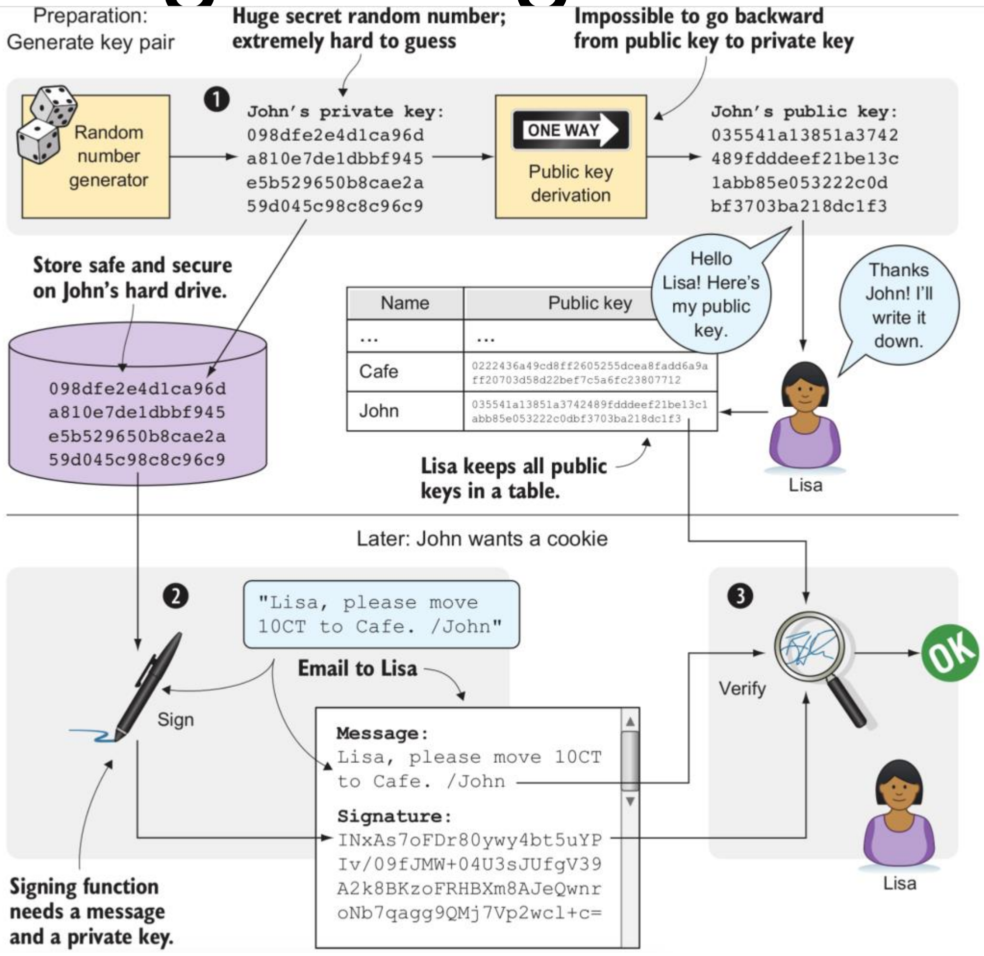
Put the signature in the email.

Digital signature for the attached cat picture

Sign the cat picture.

Email from you to Fred

Digital Signatures



Preparation:
Generate key pair

Huge secret random number; extremely hard to guess

Impossible to go backward from public key to private key

Store safe and secure on John's hard drive.

Name	Public key
...	...
Cafe	0222436a49cd8ff2605255dcea8fadd6a9a ff20703d58d22bef7c5a6fc23807712
John	035541a13851a3742489fdddeef21be13c1 abb85e053222c0dbf3703ba218dc1f3

Lisa keeps all public keys in a table.

Later: John wants a cookie

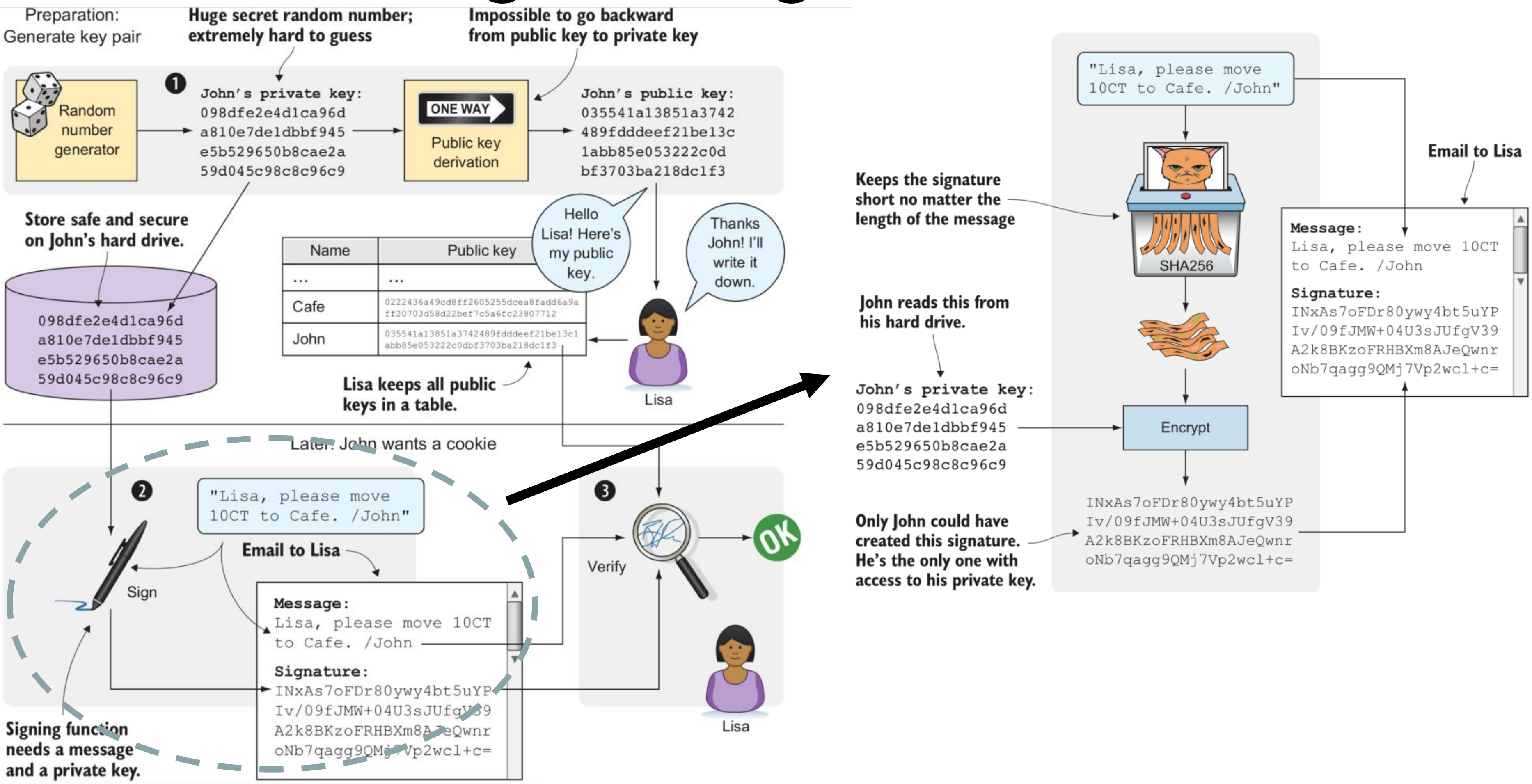
Email to Lisa

Message:
Lisa, please move 10CT to Cafe. /John

Signature:
INxAs7oFDr80ywy4bt5uYP
Iv/09fJMW+04U3sJUfgV39
A2k8BKzoFRHBXm8AJeQwnr
oNb7qagg9QMj7Vp2wcl+c=

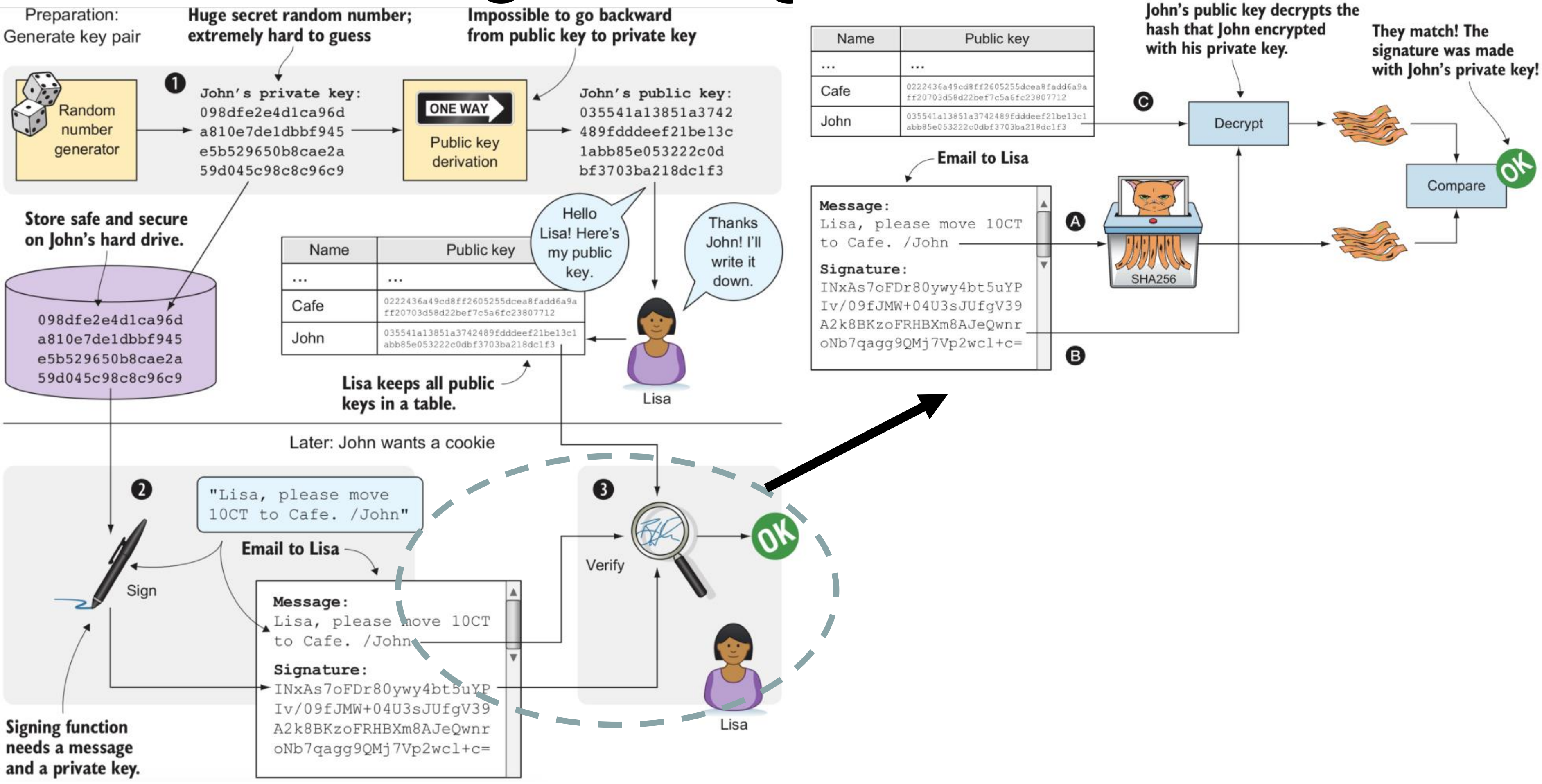
Signing function needs a message and a private key.

Digital Signatures



Name	Public key
...	...
Cafe	0222436a49cd8ff2605255dcea8fadd6a9a ff20703d58d22bef7c5a6fc23807712
John	035541a13851a3742489fdddeef21be13c1 abb85e053222c0dbf3703ba218dc1f3

Digital Signatures



Preparation:
Generate key pair
Huge secret random number; extremely hard to guess
Impossible to go backward from public key to private key
1
Random number generator
ONE WAY
Public key derivation
Store safe and secure on John's hard drive.
Hello Lisa! Here's my public key.
Thanks John! I'll write it down.
Lisa
Lisa keeps all public keys in a table.
Later: John wants a cookie
2
Sign
Email to Lisa
Signing function needs a message and a private key.
3
Verify
OK
Lisa
John's public key decrypts the hash that John encrypted with his private key.
They match! The signature was made with John's private key!
C
Decrypt
Compare
OK
Email to Lisa
A
SHA256
B

```
John's private key:
098dfe2e4d1ca96d
a810e7de1dbbf945
e5b529650b8cae2a
59d045c98c8c96c9
```

```
John's public key:
035541a13851a3742
489fdddeef21be13c
1abb85e053222c0d
bf3703ba218dc1f3
```

```
098dfe2e4d1ca96d
a810e7de1dbbf945
e5b529650b8cae2a
59d045c98c8c96c9
```

Name	Public key
...	...
Cafe	0222436a49cd8ff2605255dcea8fadd6a9a ff20703d58d22bef7c5a6fc23807712
John	035541a13851a3742489fdddeef21be13c1 abb85e053222c0dbf3703ba218dc1f3

```
"Lisa, please move
10CT to Cafe. /John"
```

```
Message:
Lisa, please move 10CT
to Cafe. /John

Signature:
INxAs7oFDr80ywy4bt5uYP
Iv/09fJMW+04U3sJUfgV39
A2k8BKzoFRHBXm8AJeQwnr
oNb7qagg9QMj7Vp2wcl+c=
```

Name	Public key
...	...
Cafe	0222436a49cd8ff2605255dcea8fadd6a9a ff20703d58d22bef7c5a6fc23807712
John	035541a13851a3742489fdddeef21be13c1 abb85e053222c0dbf3703ba218dc1f3

```
Message:
Lisa, please move 10CT
to Cafe. /John

Signature:
INxAs7oFDr80ywy4bt5uYP
Iv/09fJMW+04U3sJUfgV39
A2k8BKzoFRHBXm8AJeQwnr
oNb7qagg9QMj7Vp2wcl+c=
```

Demo

- https://learn.pkiindia.in/index.html