Chapter 8: Introduction to Cryptography

Information Security

Nguyễn Đăng Quang

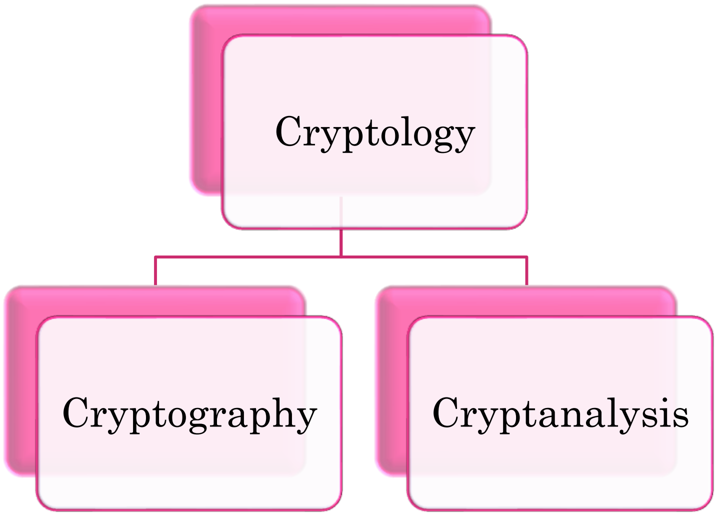
Fall 2017

Goals

- Explain the security services that cryptography provides,
- Explain the security primitives can be used to provide security services
- Describe the components of a Cryptosystem,
- Identify the Symmetric vs. Asymmetric key cryptosystem and security services that they provide.
- Explain the operations of traditional ciphers
- Explain the features of Cryptographic hashing algorithm.
- Explain the meaning of message authentication code and the benefits.
- Explain the process and meaning of digital signatures.

Cryptology

Cryptology is the study of techniques for secure communication and information protection. It encompasses both cryptography, which involves the creation and use of codes and ciphers, and cryptanalysis, which focuses on breaking codes and ciphers to gain unauthorized access to information.



Cryptology

- Cryptography: deals with the actual securing of digital data, concerns with the design of cryptosystems
- Cryptanalysis: involves the study of cryptographic mechanism with the intention to break them, study the breaking of cryptosystems

Security services of Cryptography

Security services refer to the measures and mechanisms employed to ensure the security goals.

1. Confidentiality (privacy/secrecy)
2. Integrity
3. Authentication
4. Non-repudiation

Cryptography Primitives

The tools and techniques in Cryptography that can be selectively used to provide a set of desired security services.

1. Encryption
2. Hash functions
3. Message Authentication codes (MAC)
4. Digital Signatures

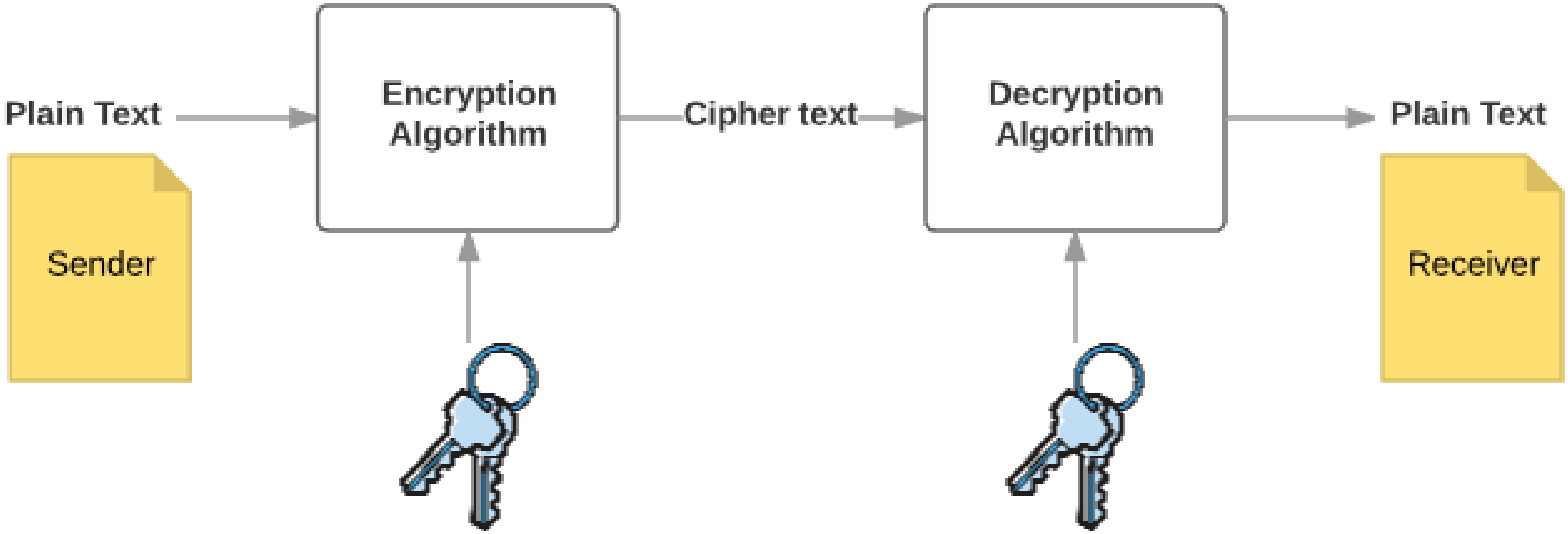
Cryptography primitives and Security services provides

Cryptography Primitives → Security Services ↓	Encryption	Hash functions	Message Authentication Code (MAC)	Digital Signatures
Confidentiality	Yes	No	No	No
Integrity	No	Sometimes	Yes	Yes
Authentication	No	No	Yes	Yes
Non-repudiation	No	No	Sometimes	Yes

Cryptosystem (Cipher system)

A cryptosystem is a system or framework that combines cryptographic algorithms and protocols to provide secure communication and information protection. It includes components such as encryption algorithms, decryption algorithms, key management mechanisms, and cryptographic protocols.

Cryptosystem (Cipher system)



- Symmetric Key Encryption: same keys are used for encrypting and decrypting (secret key cryptosystem). Primarily used for **privacy** and **confidentiality**

- Asymmetric Key Encryption: where different keys are used for encrypting and decrypting the information. Primarily used for **authentication**, **non-repudiation.**

Symmetric Key Encryption

Well-known examples: (DES), Triple-DES (3DES), AES.

Features:

- Both parties must share a common key prior to exchange of information,
- Keys are recommended to be changed regularly to prevent attack on the system.
- A robust mechanism needs to exist to exchange the key b/w the communicating parties. As keys are required to be changed regularly, this mechanism becomes expensive and cumbersome.

Symmetric Key Encryption

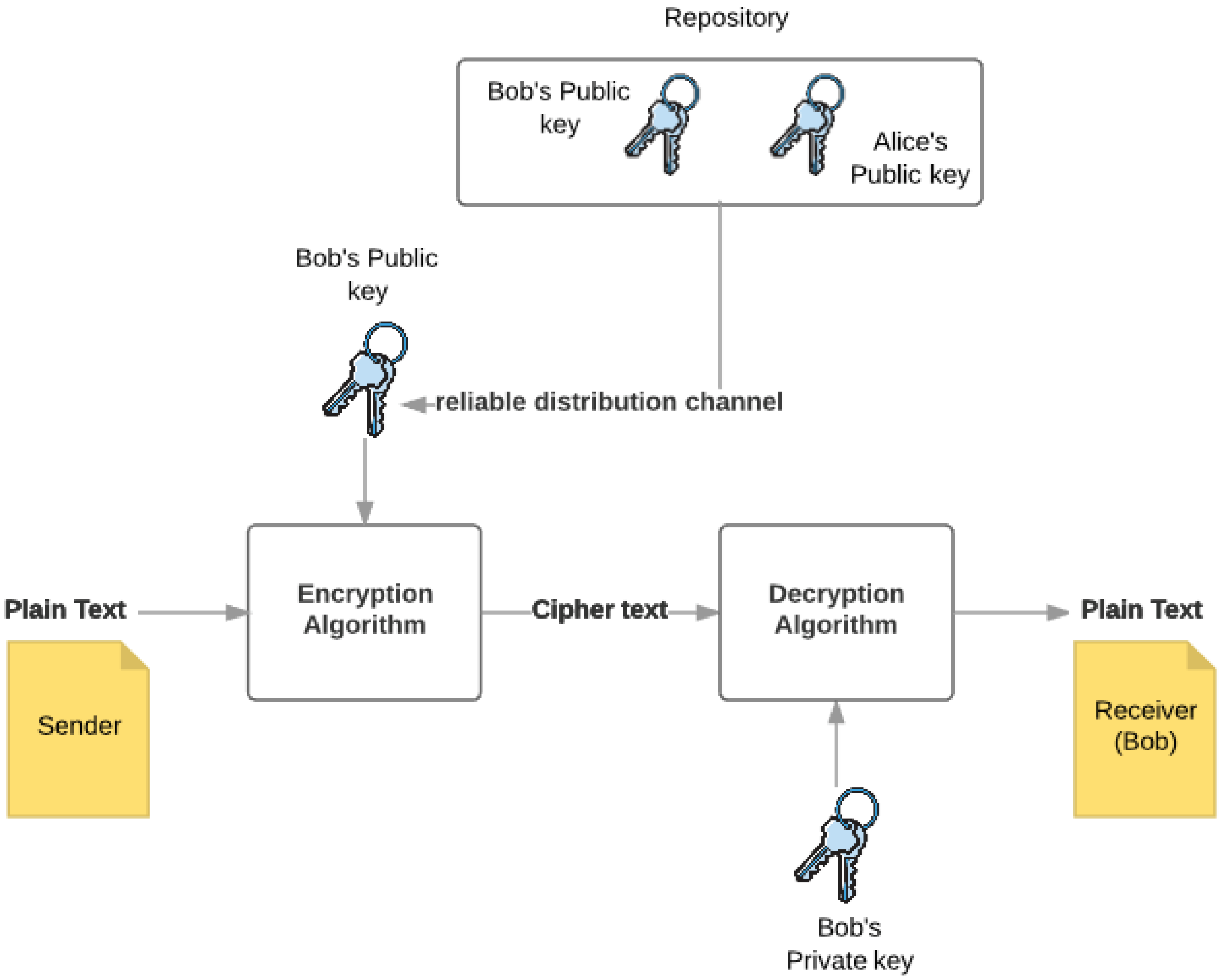
- In a group of **n** people, to enable two-party communication between any two persons, the number of keys required for group is **n × (n – 1)/2**.
- Length of Key (number of bits) is smaller ➔ process of encryption-decryption is faster than asymmetric key encryption.
- Processing power of computer system required to run symmetric algorithm is less.

Challenges

- **Key establishment** – Before any communication, both the sender and the receiver need to agree on a secret symmetric key. It requires a secure key establishment mechanism in place.
- **Trust Issue** – Since the sender and the receiver use the same symmetric key, there is an implicit requirement that the sender and the receiver 'trust' each other.

Asymmetric Key Encryption

Invented in the 20th century to overcome the necessity of pre-shared secret key between communicating parties.



Asymmetric Key Encryption

- Every user needs to have a pair of dissimilar but mathematically related keys, **private key** and **public key**.
- One key is used for encryption, the other is for decryption.
- It requires to put the public key in public repository and the private key as a well-guarded secret → **Public Key Encryption**.

Asymmetric Key Encryption

- Though public and private keys are related, it is computationally not feasible to find one from another. This is a strength of this scheme.
- When *Alice* needs to send data to *Bob*, he obtains the public key of *Bob* from repository, encrypts the data, and transmits.
- *Bob* uses his private key to extract the plaintext.
- Length of Keys (number of bits) in this encryption is large →the process of encryption-decryption is slower than symmetric key encryption.

Challenge

- Public-key cryptosystems have one significant challenge – the user needs to trust that the public key that he is using in communications with a person really is the public key of that person and has not been spoofed by a malicious third party.
- This is usually accomplished through a Public Key Infrastructure (PKI) consisting a trusted third party.

Stream vs Block Ciphers

Two encryption/decryption algorithms belong to the family of symmetric key ciphers

- Block ciphers encrypt data in fixed-size blocks. The plaintext is divided into blocks, and each block is encrypted separately.
- Stream ciphers works bit-by-bit, using keystreams to generate ciphertext for arbitrary lengths of plain text messages.

Stream vs Block Ciphers

Bit conversion

Stream ciphers convert plaintext to ciphertext 1 byte at a time.

Block ciphers transform plaintext 1 block (64/128/256 bits) at a time.

Speed

Faster

Slower

Translation principle

Stream ciphers utilize only the confusion principle to transform data, ensuring data confidentiality

Block ciphers use data diffusion and confusion to encrypt plaintext → implement authenticated encryption for enhanced security.

Reversibility

Use an XOR operation on the plaintext to create ciphertext → easier to reverse

Encrypt more bits at a time, making the decryption comparatively complex

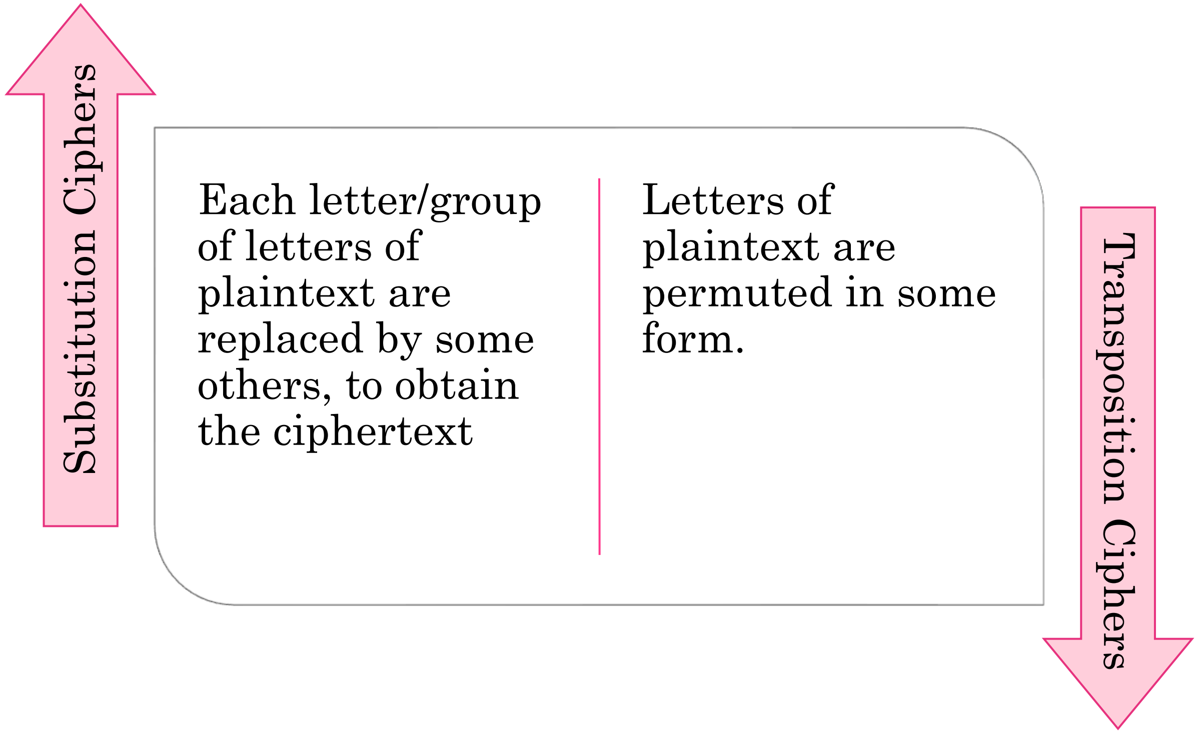
Attacking a cipher

There are two general approach:

1. Cryptanalysis
2. Brute force

Traditional Ciphers

- **Based on symmetric key encryption** scheme.
- The only security service these systems provide is confidentiality.



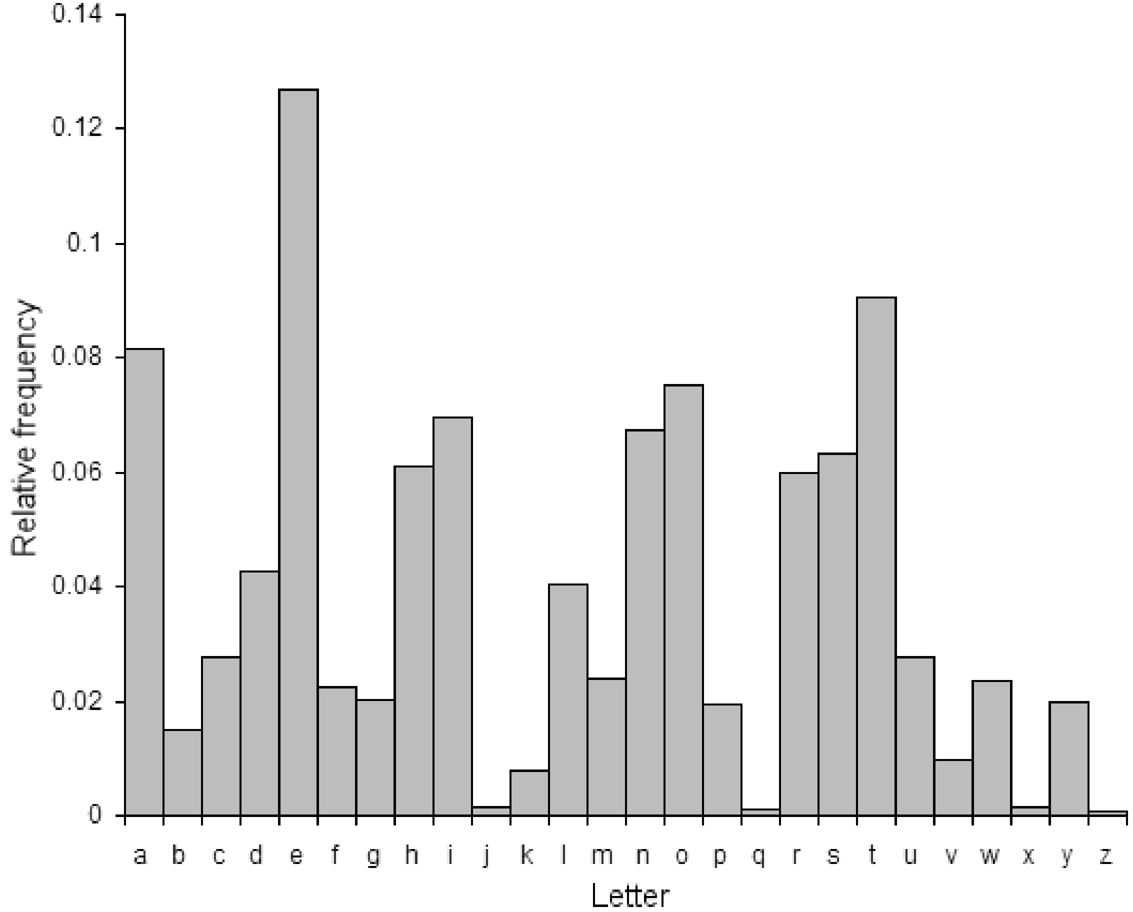
Each letter/group of letters of plaintext are replaced by some others, to obtain the ciphertext

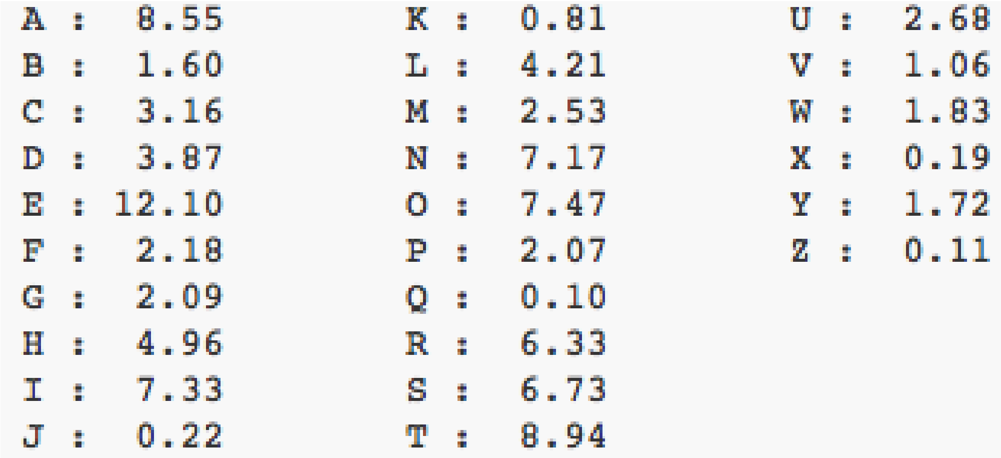
Letters of plaintext are permuted in some form.

Substitution Cipher

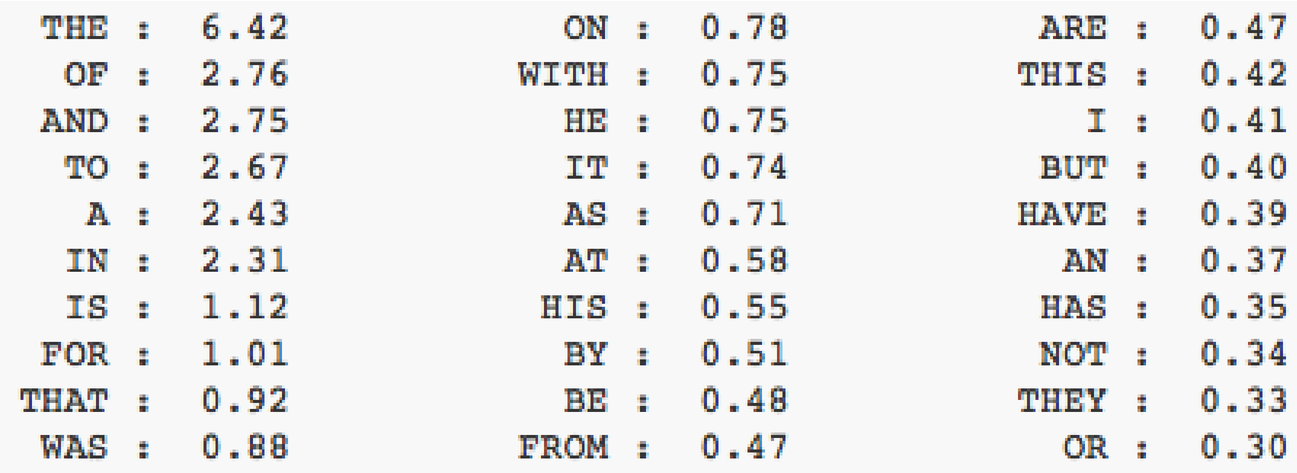
- All cryptographic algorithms involve substituting one thing for another.
- Caesar cipher: Taking each letter in the plaintext message and substituting the letter that is k letters later.
- For $k = 3$, the text “bob, i love you. alice” becomes “ere, l oryh brx. dolfh” in ciphertext.
- There are only 25 possible key values to break the code if one know that the Caesar cipher was being used.

English letters frequency





A	:	8.55	K	:	0.81	U	:	2.68
B	:	1.60	L	:	4.21	V	:	1.06
C	:	3.16	M	:	2.53	W	:	1.83
D	:	3.87	N	:	7.17	X	:	0.19
E	:	12.10	O	:	7.47	Y	:	1.72
F	:	2.18	P	:	2.07	Z	:	0.11
G	:	2.09	Q	:	0.10			
H	:	4.96	R	:	6.33			
I	:	7.33	S	:	6.73			
J	:	0.22	T	:	8.94			



THE	:	6.42	ON	:	0.78	ARE	:	0.47
OF	:	2.76	WITH	:	0.75	THIS	:	0.42
AND	:	2.75	HE	:	0.75	I	:	0.41
TO	:	2.67	IT	:	0.74	BUT	:	0.40
A	:	2.43	AS	:	0.71	HAVE	:	0.39
IN	:	2.31	AT	:	0.58	AN	:	0.37
IS	:	1.12	HIS	:	0.55	HAS	:	0.35
FOR	:	1.01	BY	:	0.51	NOT	:	0.34
THAT	:	0.92	BE	:	0.48	THEY	:	0.33
WAS	:	0.88	FROM	:	0.47	OR	:	0.30

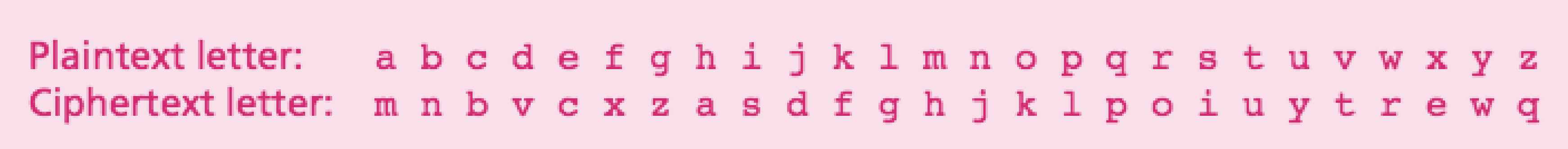
Try to break this cipher

Ciphertext: efgfoe uif fbtu xbmm pg uif dbtumf

Plaintext: ?

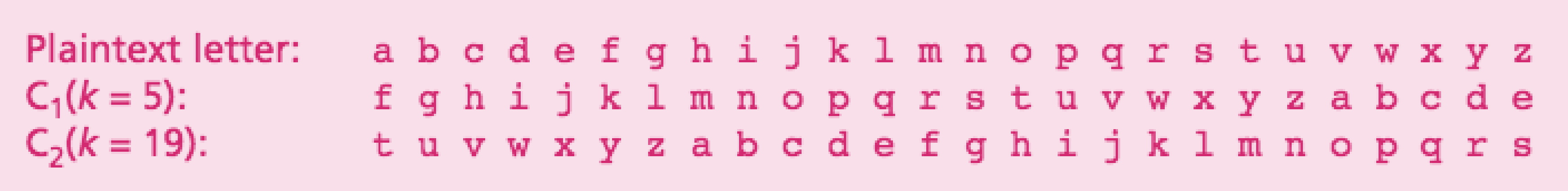
Substitution Cipher

- **Monoalphabetic**



Plaintext letter:	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 letter:	m	n	b	v	c	x	z	a	s	d	f	g	h	j	k	l	p	o	i	u	y	t	r	e	w	q

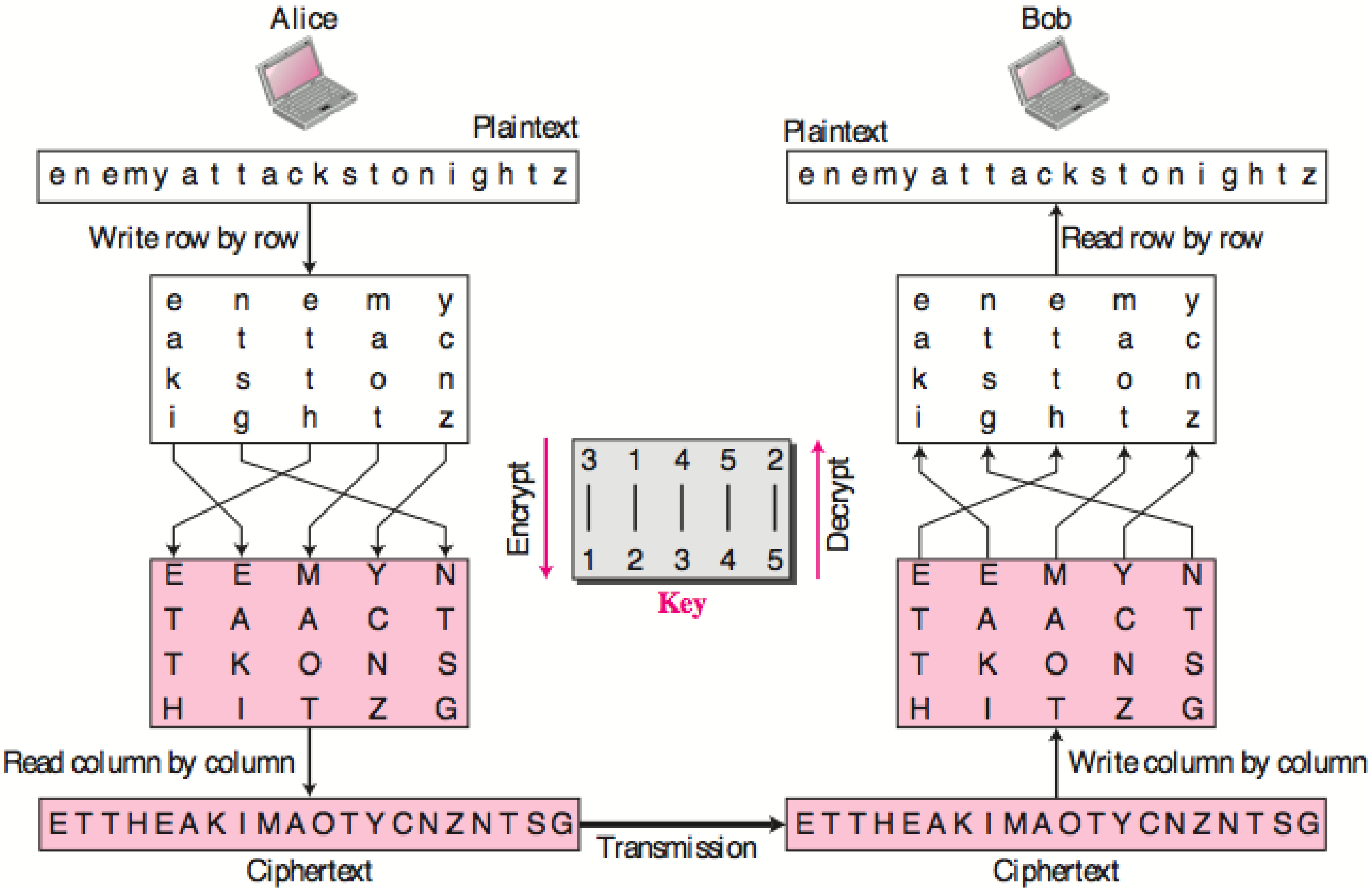
- **Polyalphabetic**



Plaintext letter:	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
$C_1(k = 5)$:	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
$C_2(k = 19)$:	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

- These two Caesar ciphers, C1 and C2, can be used in the repeating pattern C1, C2, C2, C1, C2.

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

Encrypt

3	1	4	5	2
1	2	3	4	5

Key

Decrypt

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

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

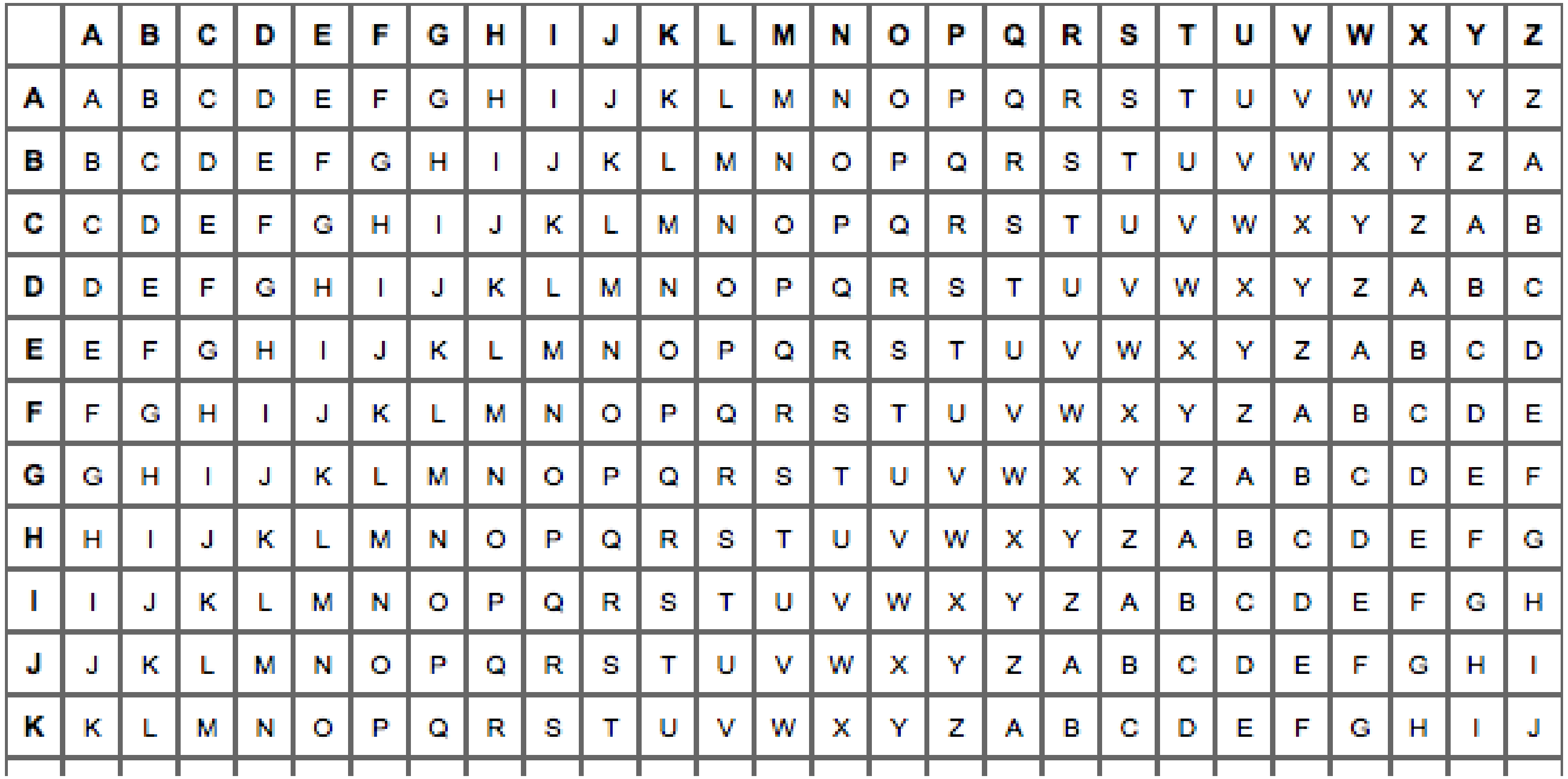
Write column by column

ETTHEAKIMAOTYCNZNTSG

Ciphertext

Vigenere Cipher

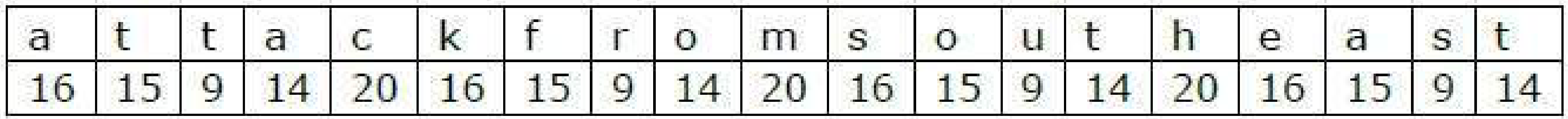
- Vigenere tableau is used;
- Each of the 26 ciphers is laid out horizontally, with the key letter for each cipher in a column



	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

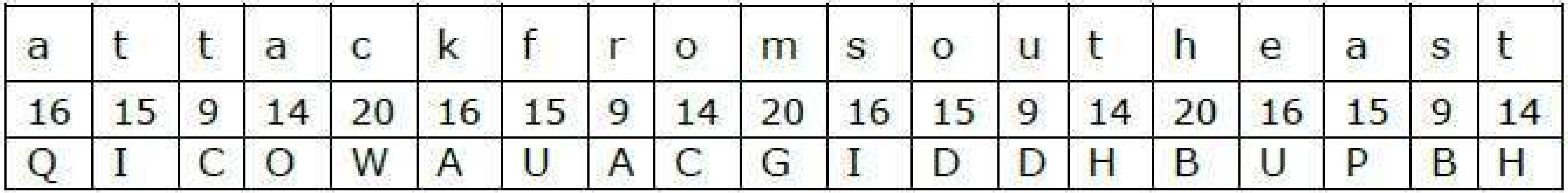
Vinegere Cipher

- For example, let’s assume the key is ‘point’. Each alphabet of the key is converted to its respective numeric value: In this case,
- p → 16, o → 15, i → 9, n → 14, and t → 20.
- Thus, the key is: 16 15 9 14 20
- Plaintext = ‘attack from the south east’



a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14

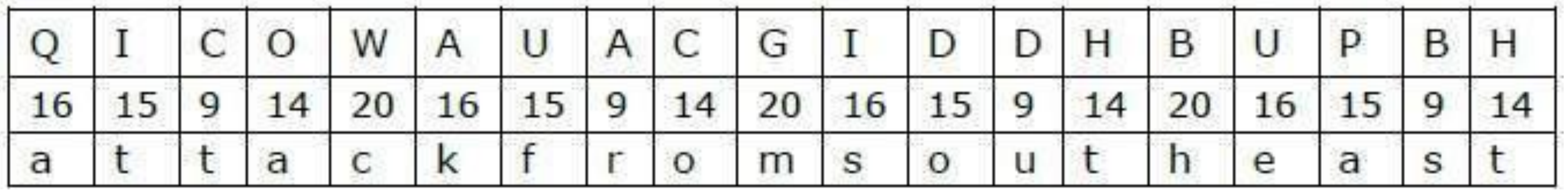
- Ciphertext



a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14
Q	I	C	O	W	A	U	A	C	G	I	D	D	H	B	U	P	B	H

Vinegere Cipher

- For Decryption, the receiver uses the same key and shifts received ciphertext in reverse order to obtain the plaintext.



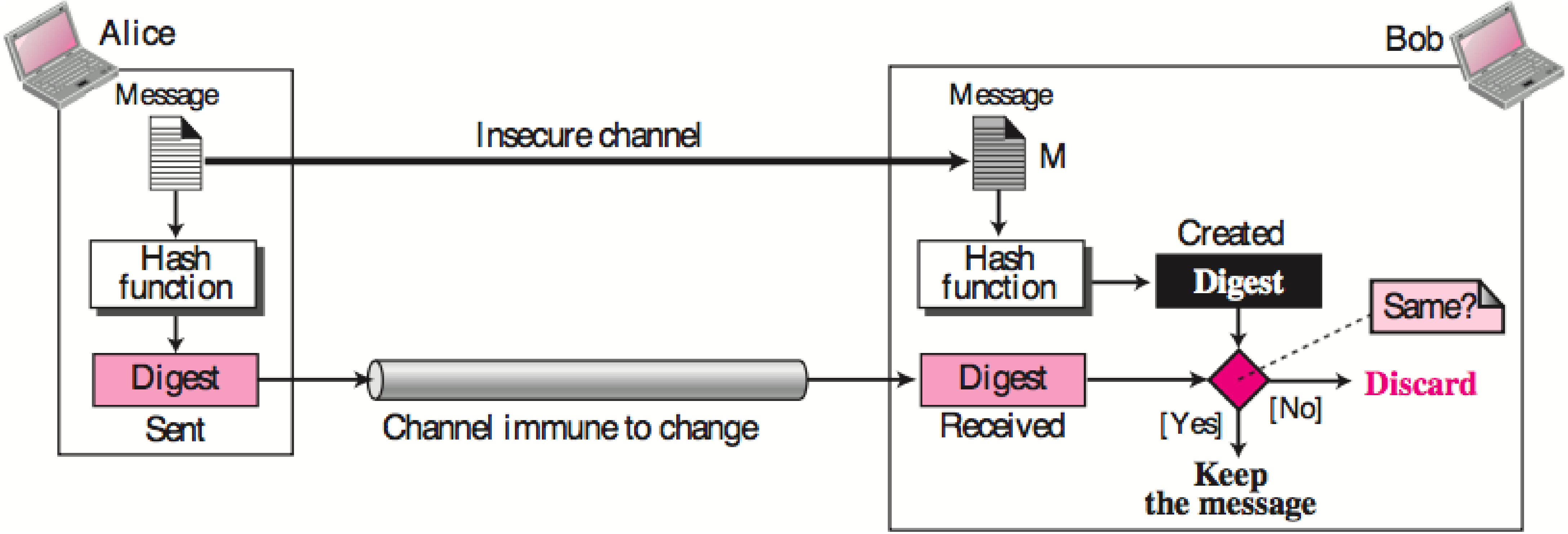
Q	I	C	O	W	A	U	A	C	G	I	D	D	H	B	U	P	B	H
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14
a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t

- In the history, it was regularly used for protecting sensitive political and military information. It was referred to as the **unbreakable cipher** due to the difficulty it posed to the cryptanalysis.

- Variation: One-time pad (Vernam cipher).

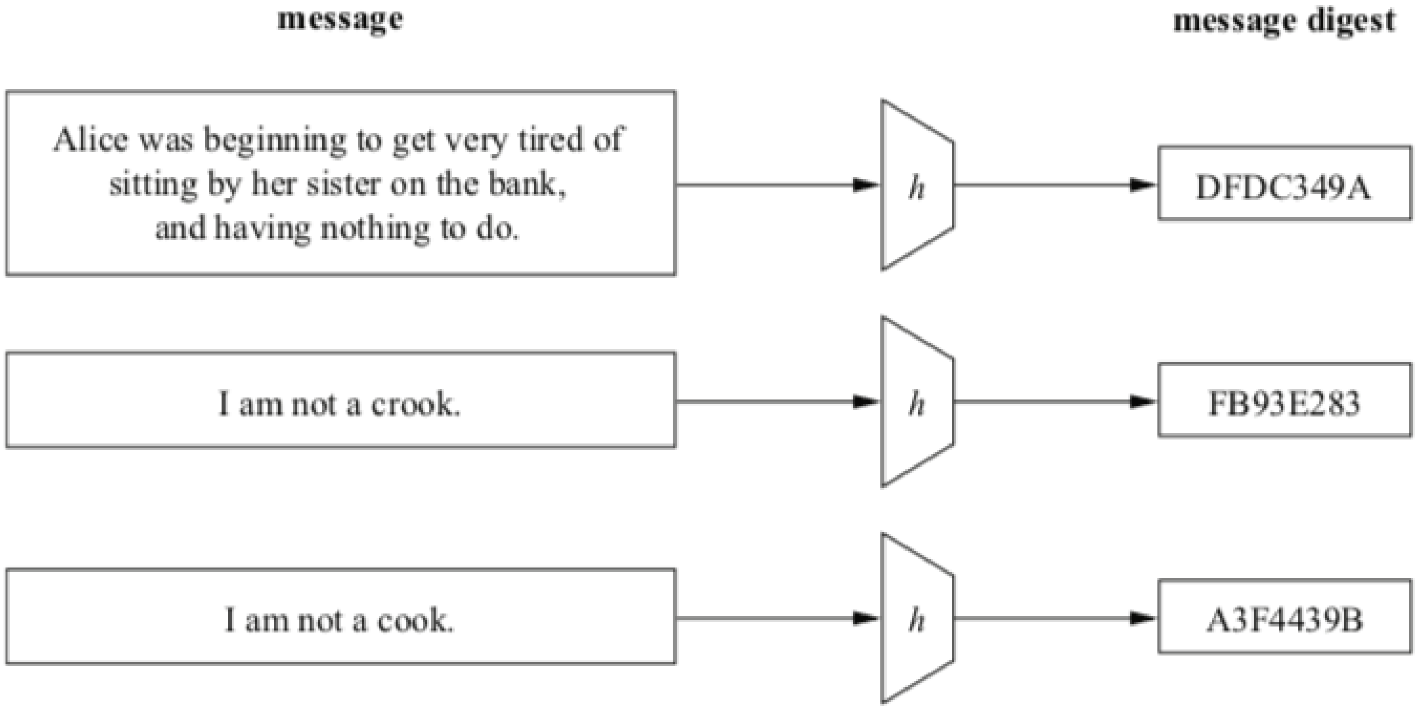
Message integrity

- To preserve the integrity of a document → fingerprint.
- To preserve the integrity of a message, the message is passed through a **cryptographic hash algorithm** to creates a **digest** which is the compressed image of the message.



Crypto Hash Properties

- Compression: length of y = h(x) is fixed (128-512b) regardless of input
- Computational Efficiency: Easy to compute h(x) but cannot too fast
- Output should look random
- Avalanche effect



Alice was beginning to get very tired of sitting by her sister on the bank, and having nothing to do.

DFDC349A

I am not a crook.

FB93E283

I am not a cook.

A3F4439B

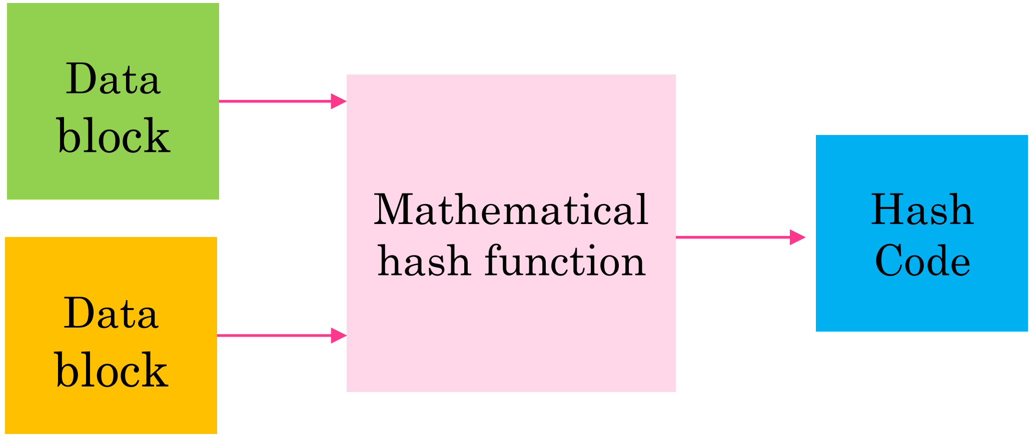
Crypto Hash Function - Requirements

- One-way: (pre-image resistance) – no feasible way to invert the hash
- Weak collision resistance: (second pre-image resistance) - Given x and h(x), infeasible to find any y , with y !=x that h(y) = h(x) ➔ No feasible way to modify a message without changing its hash value.
- (Strong) Collision resistance: infeasible to find any x, y that x!=y and h(x) = h(y) ➔ Can not find 2 inputs that hash to the same output.

Hash functions

- The heart of a hashing is a mathematical function that operates on two fixed-size blocks of data to create a hash code.

- The size of each data block varies depending on the algorithm. Typically, the block sizes are from 128 bits to 512 bits



- Two general types of hash functions:
  1. Dedicated hash functions
  2. Block cipher-based hash functions

Dedicated hash functions

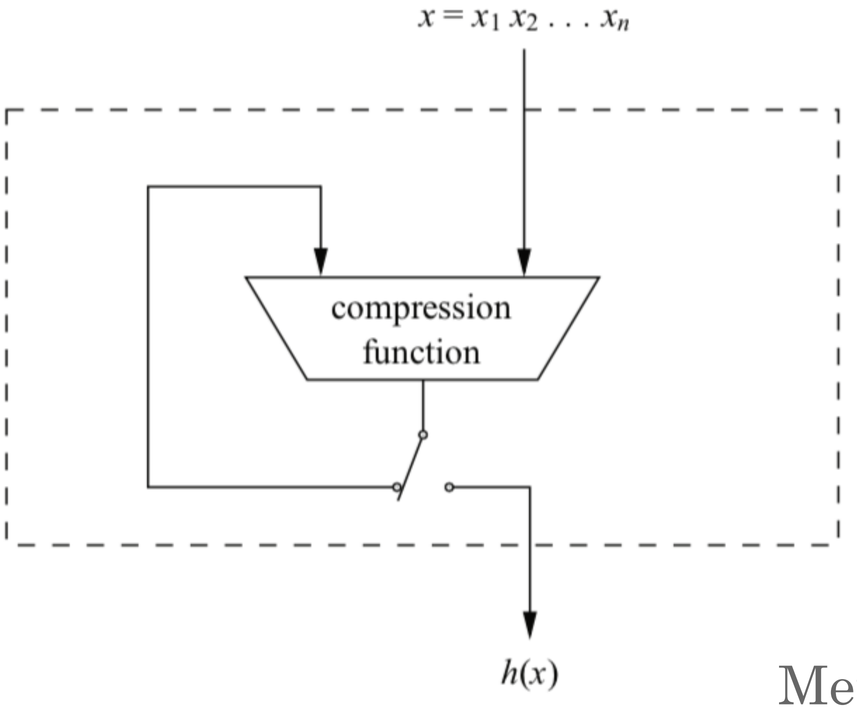
MD4 family:

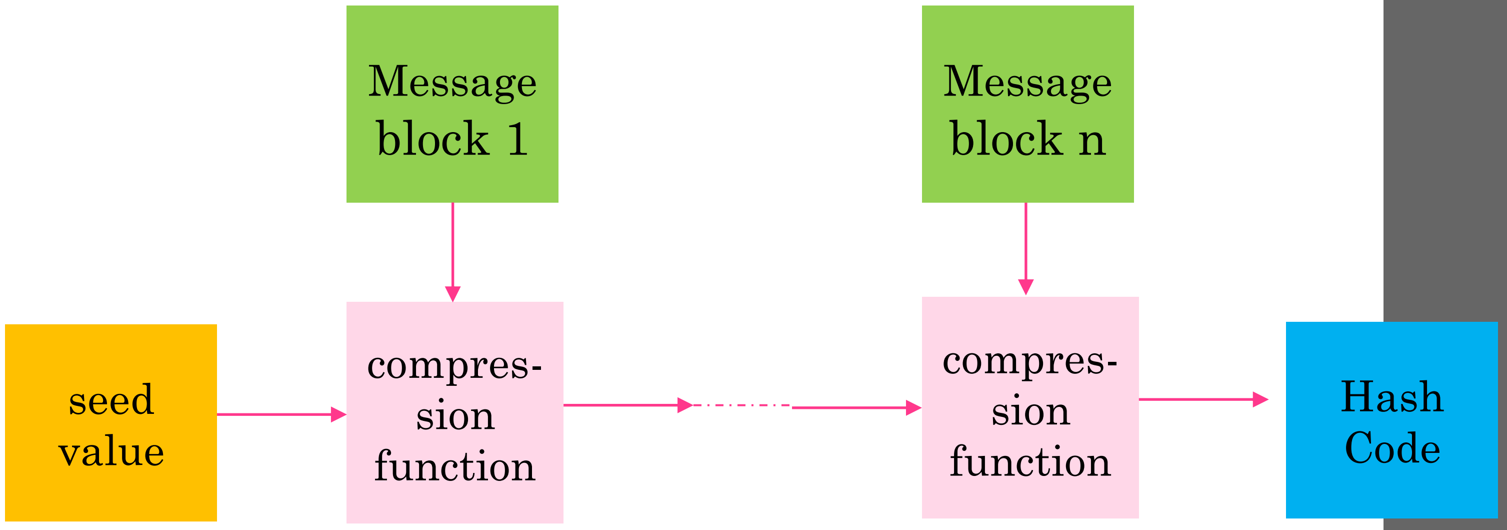
- MD4 (Ronald Rivest): was especially designed to allow very efficient software implementation. It uses 32-bit variables, and all operations are bitwise-functions such as logical AND, OR, XOR and negation
- MD5 (Rivest – 1991): 128-bit output → collision resistance about 2^{64}
- SHA-1 (NSA-2005): 160-bit output
- SHA-2 (NSA-2001): SHA-256, SHA-384, SHA-512, SHA-224 (2004)

MD4 Family of hash functions

Algorithm		Output (bit)	Input (bit)	# of round	Collision found
MD5		128	512	64	Yes
SHA-1		160	512	80	no yet
SHA-2	SHA-224	224	512	64	no
	SHA-256	256	512	64	no
	SHA-384	384	1024	80	no
	SHA-512	512	1024	80	no

Hashing Algorithms





Merkle-Damgard construction

if the compression function is collision resistant, then so is the resultant iterated hash function [Merkle-Damgard]

Birthday attack

- The number of messages we need to hash to find a collision is roughly equal to the square root of the number of possible output values (about √2n = $2^{n/2}$).
- For a security level of x bit, the hash function needs to have an output length of $2x$ bit

Applications

Integrity verification

```
$ echo -n "Hello world" | sha256sum
$ echo -n "Hallo world" | sha256sum
```

Password verification

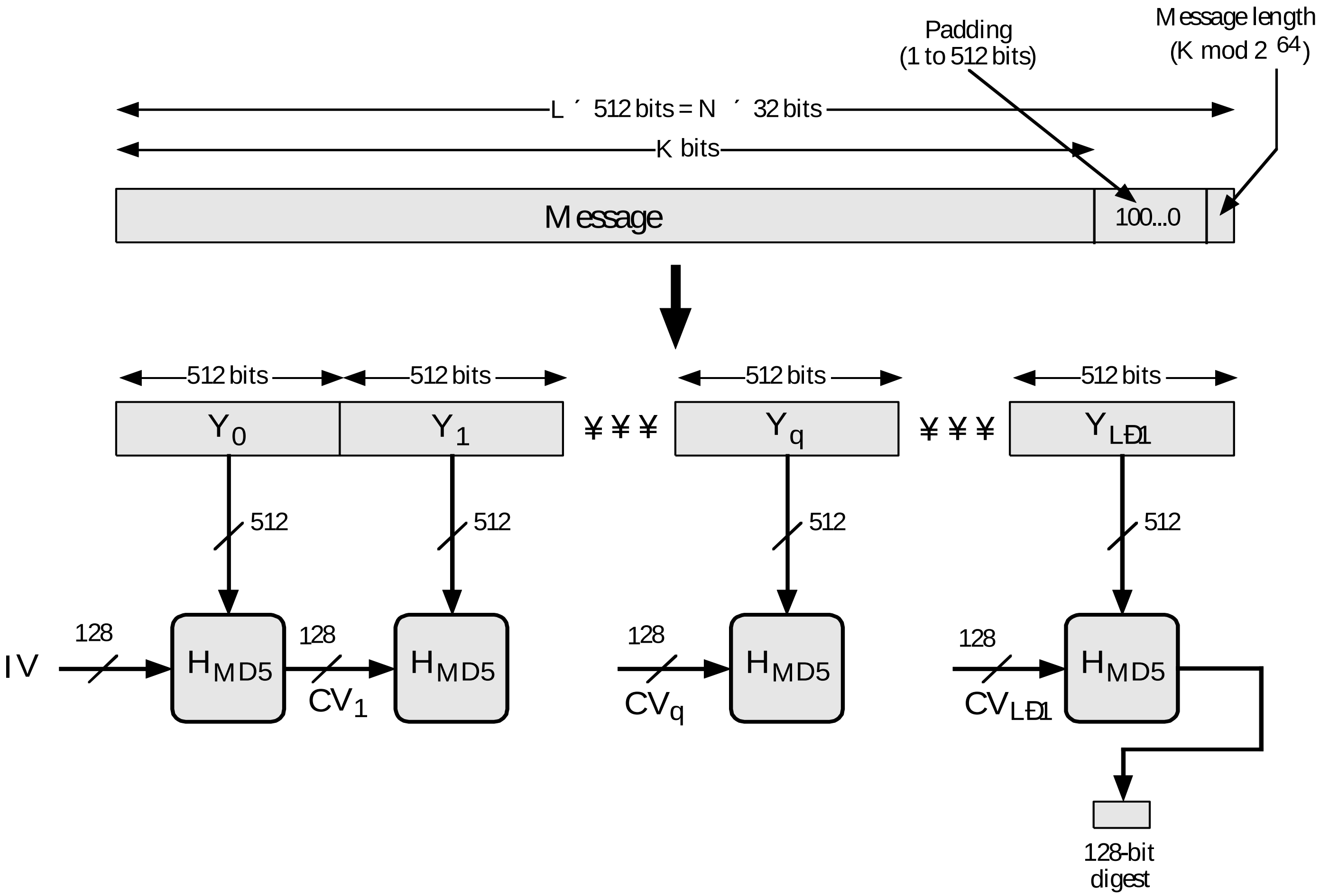
```
seed:$6$RoyYcXFsV388I4B5$paCi7.ZO3Pbn03UtNMIpv8ZkwMha8i2FoSrl1
:19301:0:99999:7:::

test:$6$NhHt4NWQVw41JJ6r$IpnUKtvRgP/X/UnvLUl6e.GCOo0yU1EIWeUZ8dTyW.:
193010:99999:7:::
```

$1 – MD5, $5 – SHA-256, $6 – SHA-512

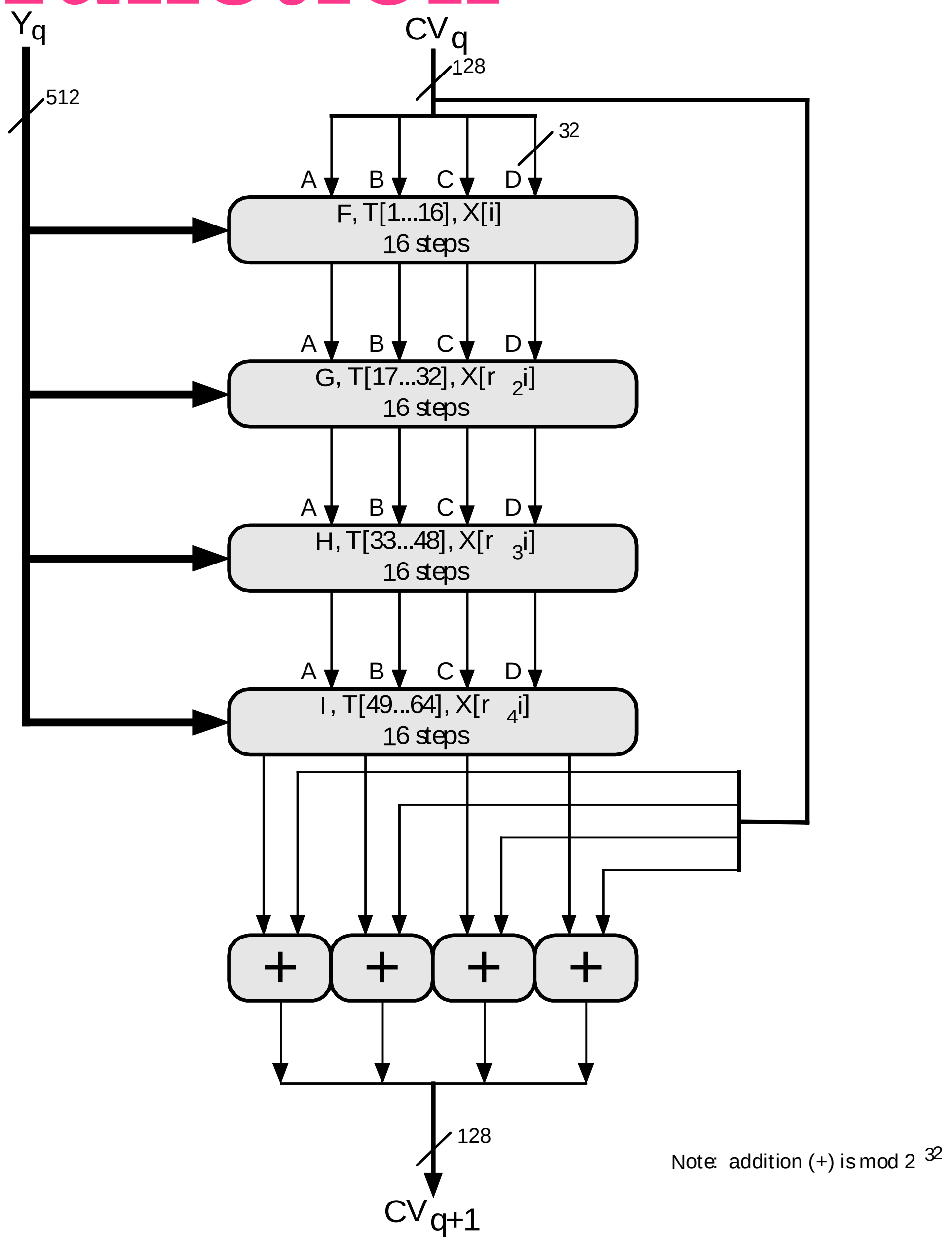
Hashed password: many rounds of hash function. For example, 5000 rounds for SHA-512

MD5 algorithm



- https://bit.ly/2yOLdAj

MD5 compression function



Note: addition (+) is mod 2 32

Elementary MD5 operation

The helper functions:

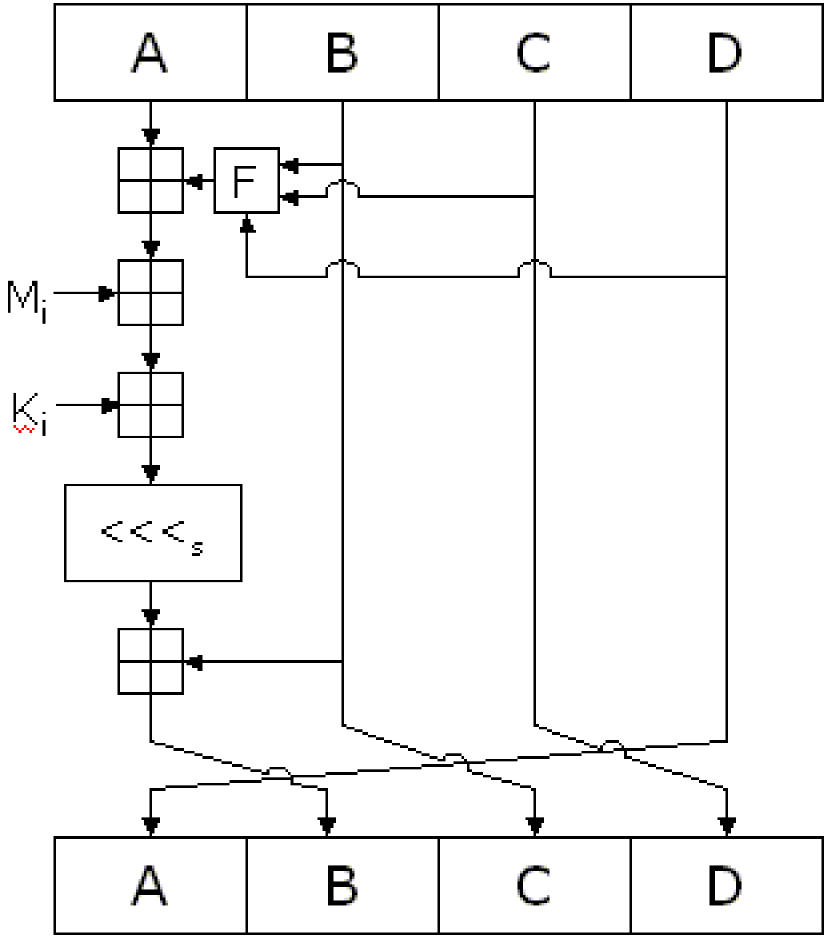
- The initialized buffer words:
  A = 0x67452301, B = 0xefcdab89,
  C = 0x98badcfe, D = 0x10325476

- The table K (64 elements) to speed-up the computation $K_i = abs(sin(i + 1)) * 2^{32}$

- Four auxiliary functions

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Y) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

- Shift amount per round



```
s[ 0..15] := { 7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22 }
s[16..31] := { 5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20 }
s[32..47] := { 4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23 }
s[48..63] := { 6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21 }
```

Secure Hash Algorithm (SHA-1)

Summary

- Hash functions are keyless. The two most important applications of hash functions are their use in **digital signatures** and in **message authentication codes** such as HMAC.
- The three security requirements for hash functions are one-way, second preimage resistance and collision resistance.
- Hash functions should have at least 160-bit output length in order to withstand collision attacks; 256 bit or more is desirable for long-term security.
- MD5, which was widely used, is insecure. Serious security weaknesses have been found in SHA-1, and the hash function should be phased out. The SHA- 2 algorithms all appear to be secure.

Popular Hash Functions

Message Digest (MD)

- MD5 was most popular and widely used hash function for some years.
- The MD family comprises of hash functions MD2, MD4, MD5 and MD6. It was adopted as Internet Standard RFC 1321. It is a 128-bit hash function.
- MD5 digests have been widely used in the software world to provide assurance about integrity of transferred file.
- In 2004, collisions were found in MD5. An analytical attack was reported to be successful only in an hour by using computer cluster. This collision attack resulted in compromised MD5 and hence it is no longer recommended for use.

Popular Hash Functions

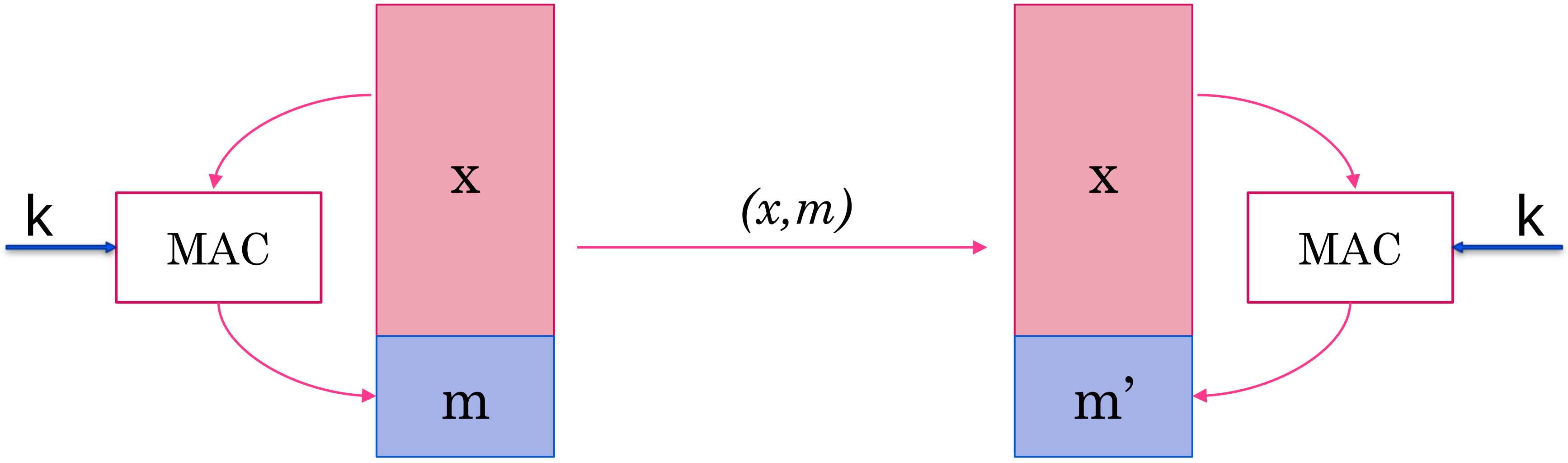
Secure Hash Function (SHA)

- Family of SHA comprise of four SHA algorithms; SHA-0, SHA-1, SHA-2, and SHA-3. Though from same family, there are structurally different.
- The original version is SHA-0 (160-bit), was published by NIST in 1993. It had few weaknesses and did not become very popular. Later in 1995, SHA-1 was designed later in 1995 to correct weaknesses of SHA-0.
- SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols including Secure Socket Layer (SSL) security.
- SHA-2 family has four further SHA variants, SHA-224, SHA-256, SHA-384, and SHA-512 depending up on number of bits in their hash value. No successful attacks have yet been reported on SHA-2 hash function.

Message Authentication Code (MAC)

- Known as a *tag*, is a short piece of information used to authenticate a message.
- The MAC value protects both a message's data integrity as well as its authenticity.
- MAC is actually a **hash**, which provides integrity, with **key,** which provides authentication for the source of the message.
- MAC consists of three algorithms:
  1. A **key** generation algorithm selects a key from the key space uniformly at random.
  2. A signing algorithm efficiently returns a **tag** given the key and the message.
  3. A verifying algorithm efficiently verifies the authenticity of the message given the key and the tag. That is, return *accepted* when the message and tag are not tampered with or forged, and otherwise return *rejected*.

Message Authentication Code (MAC)



verification :

$$m \overset{?}{=} m'$$

Properties of MAC

1. Arbitrary input length,
2. Fix output length,
3. Message authentication: Alice is certain that Bob sent the message
4. Integrity: Manipulations in transit will be detected by Alice,

Limitations

- **Establishment of Shared Secret**
  - It can provide message authentication among pre-decided legitimate users who have shared key.
  - This requires establishment of shared secret prior to use of MAC.
- **Inability to provide non-repudiation**
  - If the sender and receiver get involved in a dispute over message origination, MACs cannot provide a proof that a message was indeed sent by the sender.

Constructing MAC & Potential Attacks

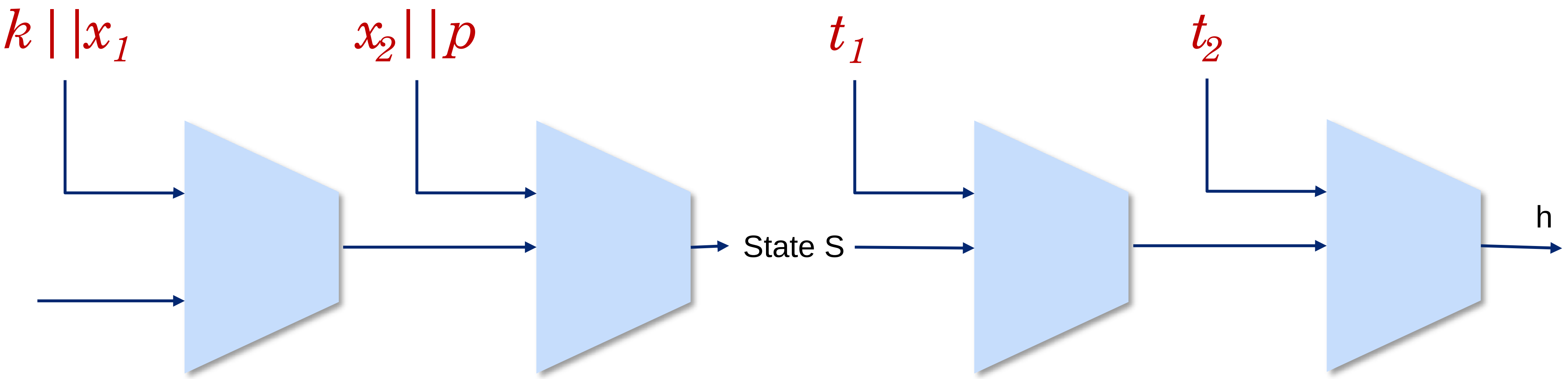
The basic idea behind all hash-based message authentication codes is that the key is hashed together with the message. Two obvious constructions are possible.

- Secret Prefix MAC: $m = \mathrm{MAC}_k(x) = h(k \,||\, x)$
- Secret Suffix MAC: $m = \mathrm{MAC}_k(x) = h(x \,||\, k)$

Consider the prefix MAC $m = h(k \,||\, x)$, the message x that needs to be signed is the sequence of blocks x = (x_1, x_2, x_3, …x_n). The computed $m = h(k \,||x_1, x_2, x_3, \ldots x_n)$.

It is found that given a hash h(x), the attacker can compute $h(k \,||x\,||\,p\,||\,t)$ for any additional string (block) t *where p is the padding used when calculating* $h(k \,||\, x)$. The padding does not depend on the content of k or x, but depends on their length

Length Extension Attack



S = hash(k || x)

h = hash(k || x || p || t)

MACs from Hash Functions: HMAC

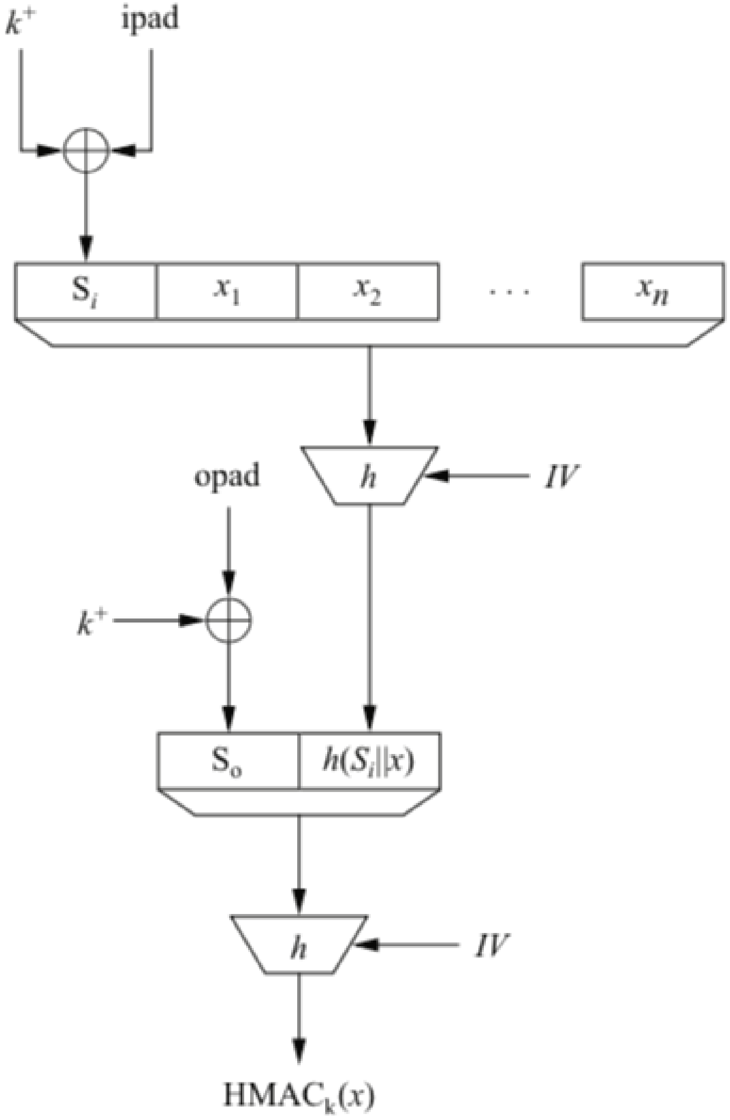
- HMAC or keyed-hash m = MAC$_k$(x)
- Any cryptographic hash function (MD5, SHA) can be used in calculation of an HMAC.
- HMAC is used in both the Transport Layer Security (TLS) protocol and IPsec.

$$\mathrm{HMAC}(K, m) = \mathrm{H}\Big(\big(K' \oplus opad\big) \parallel \mathrm{H}\big(\big(K' \oplus ipad\big) \parallel m\big)\Big)$$

$$K' = \begin{cases} \mathrm{H}(K) & K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

ipad = 00110110, 00110110, ..., 00110110 (0x63)

opad = 01011100, 01011100, ..., 01011100 (0x5c)



Summary

- MACs provide message integrity and message authentication, using symmetric techniques.
- Both of these services are also provided by digital signatures, but MACs are much faster.
- MACs do not provide nonrepudiation
- MACs are either based on block ciphers or on hash functions.
- HMAC is a popular MAC used in many practical protocols such as TLS, IPsec

Digital Signature

- When Alice sends a message to Bob, Bob needs to check the authenticity of the sender; he needs to be sure that the message comes from Alice and not Eve. Bob can ask Alice to sign the message electronically.

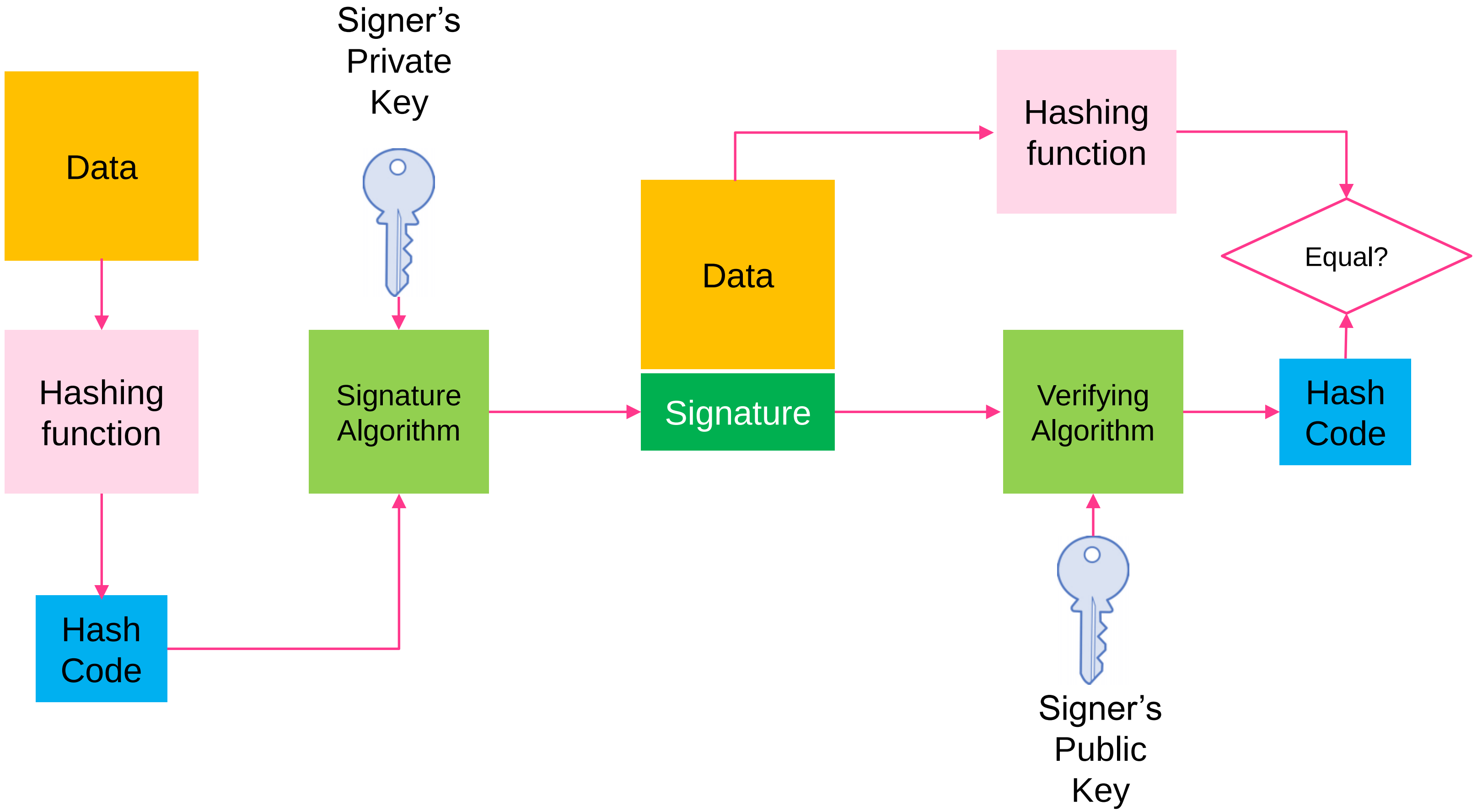
- An electronic signature can prove the authenticity of Alice as the sender of the message. This type of signature is called a **digital signature.**

- Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer

Digital Signature

- A **digital signature** is a mathematical scheme for demonstrating the authenticity of digital messages or documents. A valid digital signature gives a recipient reason to believe that the message was created by a known sender (authentication), that the sender cannot deny having sent the message (non-repudiation), and that the message was not altered in transit (integrity)

Process


Signer's Private Key
Data
Hashing function
Hash Code
Signature Algorithm
Data
Signature
Hashing function
Equal?
Verifying Algorithm
Hash Code
Signer's Public Key

Digital Signature

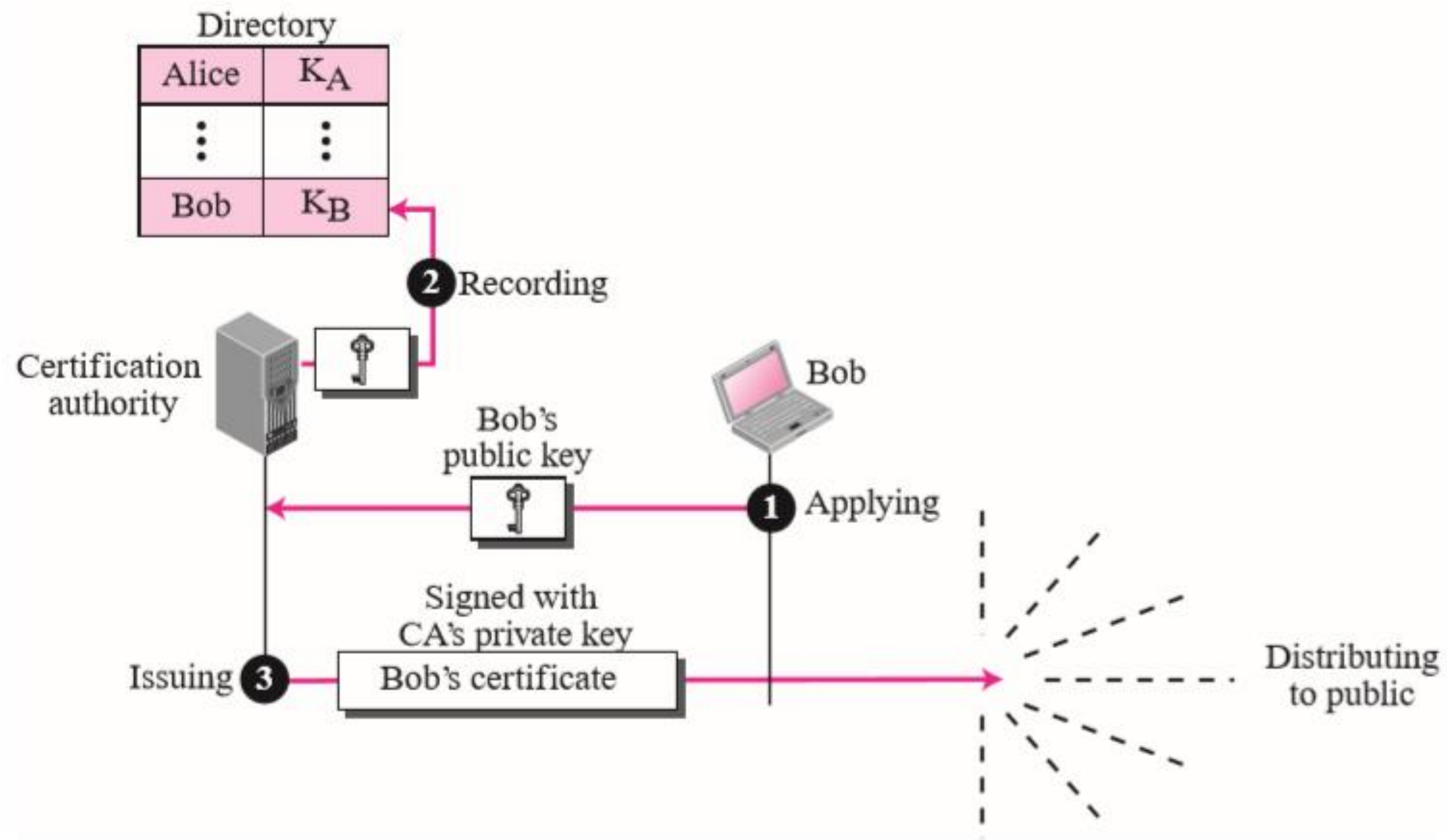
Security Services provide:

- **Message Authentication**: Bob can verify that the message is sent by Alice because Alice's public key is used in verification. Alice's public key cannot verify the signature signed by Eve's private key.
- **Message Integrity**: Through verifying the hash value.
- **Non-repudiation**: Public-key cipher provides this service

Digital Signature

- The public key component is really public and convenient.
- Any user can send his or her public key to any other user or just broadcast it to the world.
- Major weakness: Anyone can forge such a public key. Some user could pretend to be Bob, and send a public key to another user such as Alice, and tell Alice that this is Bob's public key.
- The message sent from Allice to Bob will be intercepted and read by attacker who forged Bob’s public key.

Certification Authority



A certificate consists of:

- Bob's public key and information such as the user ID, name...
- The certificate authority's information.
- The period of validity.

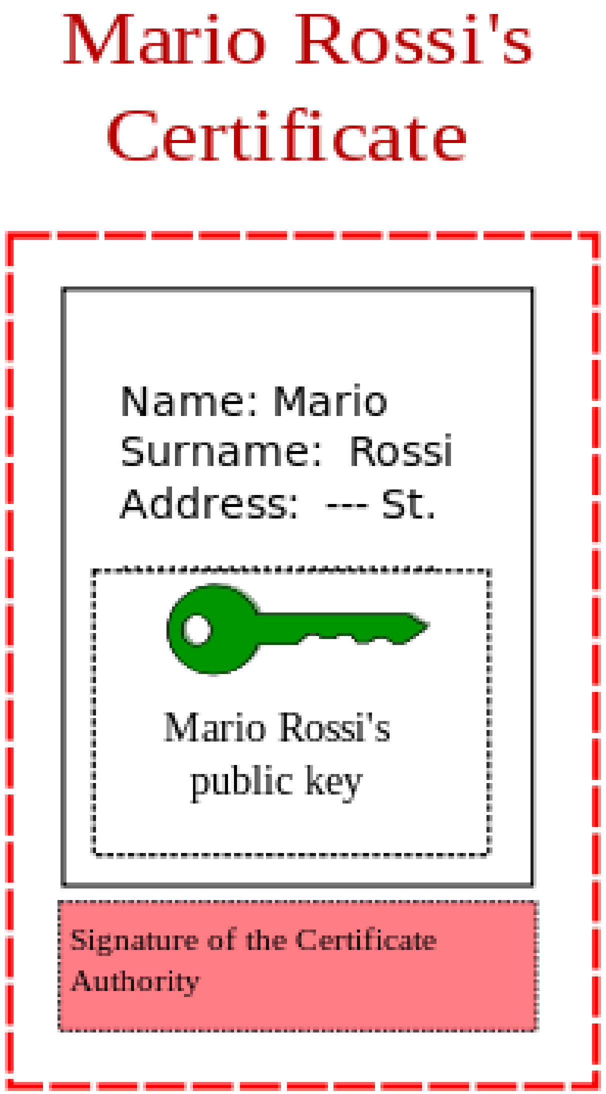
The whole block is then signed using the certificate authority's private key.

Public key certificate

A certificate consists of:

- Bob's public key and information such as the user ID, name…
- The certificate authority's information.
- The period of validity.

And the whole block is signed using the certificate authority's private key.



Public key certificate process

Suppose Bob wants the certificate authority CA to create a certificate for his public key:

1. Bob contact the CA and provide authentication information,
2. Send his public key to CA,
3. The CA will put his id, public key, the period of validity, … and then hash it with his private key,
4. The certificate of Bob’s public key is created.
5. Bob can now send this public key certificate to anybody such as Alice.

Public key certificate process

1. Alice receives Bob’s public key certificate,
2. Alice extracts the key information of Bob, public key, and all the information.
3. Alice decrypts the signature, and verify it by comparing the two hash values. If they match → this public key has been properly signed by the CA.

In practice, the CA is a well-known company such as Verisign, Microsoft, Google, or Apple, and the public keys are already stored in, for example your web browser.

With these public keys already configured on your system, they can automatically validate public key certificates signed.