CS215 DISCRETE MATHEMATICS FOR COMPUTER SCIENCE

Dr. QI WANG

Department of Computer Science and Engineering
Office: Room413, CoE South Tower
Email: wangqi@sustech.edu.cn

The Chinese Remainder Theorem

- **Theorem** (*The Chinese Remainder Theorem*) Let $m_1, m_2, \ldots, m_n$ be pairwise relatively prime positive integers greater than 1 and $a_1, a_2, \ldots, a_n$ arbitrary integers. Then the system

$$
\begin{aligned}
x &\equiv a_1 \pmod{m_1}\\
x &\equiv a_2 \pmod{m_2}\\
&\cdots\\
x &\equiv a_n \pmod{m_n}
\end{aligned}
$$

has a unique solution modulo $m = m_1 m_2 \cdots m_n$.

Back Substitution

- We may also solve systems of linear congruences with pairwise relatively prime moduli by *back substitution*.

Back Substitution

- We may also solve systems of linear congruences with pairwise relatively prime moduli by *back substitution*.

Example

$$\begin{aligned} x &\equiv 2 \pmod{3} \\ x &\equiv 3 \pmod{5} \\ x &\equiv 2 \pmod{7} \end{aligned}$$

Back Substitution

- We may also solve systems of linear congruences with pairwise relatively prime moduli by *back substitution*.

Example

$$\begin{aligned} x &\equiv 2 \pmod{3} \\ x &\equiv 3 \pmod{5} \\ x &\equiv 2 \pmod{7} \end{aligned}$$

$$\begin{aligned} x &\equiv 8 \pmod{15} \\ x &\equiv 2 \pmod{21} \end{aligned}$$

- **Theorem (Fermat's little theorem)** : Let p be a prime, and let x be an integer such that $x \not\equiv 0 \bmod p$. Then
$$x^{p-1} \equiv 1 \pmod{p}.$$

Fermat's Little Theorem

- **Theorem (Fermat's little theorem)** : Let p be a prime, and let x be an integer such that $x \not\equiv 0 \bmod p$. Then
$$x^{p-1} \equiv 1 \pmod{p}.$$

Example: Find $7^{222} \pmod{11}$

Fermat's Little Theorem

- **Theorem (Fermat's little theorem)** : Let p be a prime, and let x be an integer such that $x \not\equiv 0 \bmod p$. Then
$$x^{p-1} \equiv 1 \pmod{p}.$$

Example: Find $7^{222} \pmod{11}$

$$7^{222} = 7^{22 \cdot 10+2} = (7^{10})^{22} 7^2 = 1^{22} \cdot 49 \equiv 5 \pmod{11}$$

Fermat's Little Theorem

- **Theorem (Fermat's little theorem)** : Let p be a prime, and let x be an integer such that $x \not\equiv 0 \bmod p$. Then
$$x^{p-1} \equiv 1 \pmod{p}.$$

Example: Find $7^{222} \pmod{11}$

$$7^{222} = 7^{22\cdot 10+2} = (7^{10})^{22}7^2 = 1^{22} \cdot 49 \equiv 5 \pmod{11}$$

$\mathcal{Q}$: How to prove Fermat's little theorem?

- **Theorem (Fermat's little theorem)** : Let p be a prime, and let x be an integer such that $x \not\equiv 0 \bmod p$. Then
$$x^{p-1} \equiv 1 \pmod{p}.$$

Example: Find $7^{222} \pmod{11}$

$$7^{222} = 7^{22\cdot 10+2} = (7^{10})^{22} 7^2 = 1^{22} \cdot 49 \equiv 5 \pmod{11}$$

$\mathcal{Q}$: How to prove Fermat's little theorem?

$$\{1, 2, \ldots, p-1\} = \{x, 2x, \ldots, x(p-1) \pmod{p}\}$$

Euler's Theorem

- Euler's *totient* function: $\phi(n)$

  the number of positive integers coprime to n in $\mathbb{Z}_n$

Euler's Theorem

- Euler's *totient* function: $\phi(n)$

  the number of positive integers coprime to n in $\mathbb{Z}_n$

$$\phi(p) = p - 1$$

$$\phi(pq) = (p-1)(q-1)$$

$$\phi(p^i) = p^i - p^{i-1}$$

Euler's Theorem

- Euler's *totient* function: $\phi(n)$

  the number of positive integers coprime to n in $\mathbb{Z}_n$

$$\phi(p) = p - 1$$
$$\phi(pq) = (p-1)(q-1)$$
$$\phi(p^i) = p^i - p^{i-1}$$

- **Theorem (Euler's theorem)** : Let n be a positive integer, and let x be an integer such that $\gcd(x, n) = 1$. Then

$$x^{\phi(n)} \equiv 1 \pmod{n}.$$

Euler's Theorem

- Euler's *totient* function: $\phi(n)$

  the number of positive integers coprime to n in $\mathbb{Z}_n$

$$\phi(p) = p - 1$$
$$\phi(pq) = (p-1)(q-1)$$
$$\phi(p^i) = p^i - p^{i-1}$$

- **Theorem (Euler's theorem)** : Let n be a positive integer, and let x be an integer such that $\gcd(x, n) = 1$. Then

$$x^{\phi(n)} \equiv 1 \pmod{n}.$$

$\mathcal{Q}$: How to prove Euler's theorem?

Primitive Roots

- A *primitive root* modulo a prime p is an integer $r \in \mathbb{Z}_p$ such that every nonzero element of $\mathbb{Z}_p$ is a power of r.

Primitive Roots

- A *primitive root* modulo a prime p is an integer $r \in \mathbb{Z}_p$ such that every nonzero element of $\mathbb{Z}_p$ is a power of r.

  Example: 3 is a primitive root of $\mathbb{Z}_7$. 2 is not a primitive root of $\mathbb{Z}_7$.

Primitive Roots

- A *primitive root* modulo a prime p is an integer $r \in \mathbb{Z}_p$ such that every nonzero element of $\mathbb{Z}_p$ is a power of r.

  Example: 3 is a primitive root of $\mathbb{Z}_7$. 2 is not a primitive root of $\mathbb{Z}_7$.

  Theorem * There is a primitive root modulo n if and only if $n = 2, 4, p^e$ or $2p^e$, where p is an odd prime.

  $\mathcal{Q}$: proof? The number of primitive roots? $*$

Number Theory and Cryptography

- Division, Primes
- Congruence
- Greatest Common Divisor (GCD)
- Euler's Theorem / Fermart's Little Theorem

- Division, Primes
  $a = dq + r$
- Congruence
- Greatest Common Divisor (GCD)
- Euler's Theorem / Fermart's Little Theorem

- Division, Primes

  $a = dq + r \qquad q = a \; div \; d \qquad r = a \; mod \; d$

- Congruence

- Greatest Common Divisor (GCD)

- Euler's Theorem / Fermart's Little Theorem

- Division, Primes

  $a = dq + r \qquad q = a \; div \; d \qquad r = a \; mod \; d$

- Congruence

- Greatest Common Divisor (GCD)

- Euler's Theorem / Fermart's Little Theorem

Number Theory and Cryptography

- Division, Primes

  $a = dq + r \quad q = a \ \mathit{div} \ d \quad r = a \ \mathit{mod} \ d$

- Congruence

  $a \equiv b \pmod{m}$ if m divides $a - b$

- Greatest Common Divisor (GCD)

- Euler's Theorem / Fermart's Little Theorem

- Division, Primes

  $a = dq + r \qquad q = a \; div \; d \qquad r = a \; mod \; d$

- Congruence

  $a \equiv b \pmod{m}$ if m divides $a - b$

- Greatest Common Divisor (GCD)

- Euler's Theorem / Fermart's Little Theorem

Number Theory and Cryptography

- Division, Primes

  $a = dq + r \qquad q = a \textit{ div } d \qquad r = a \textit{ mod } d$

- Congruence

  $a \equiv b \pmod{m}$ if m divides $a - b$

- Greatest Common Divisor (GCD)

  (extended) Euclidean algorithm

- Euler's Theorem / Fermart's Little Theorem

Number Theory and Cryptography

- Division, Primes

  $$a = dq + r \qquad q = a \text{ div } d \qquad r = a \text{ mod } d$$

- Congruence

  $$a \equiv b \pmod{m} \text{ if } m \text{ divides } a - b$$

- Greatest Common Divisor (GCD)

  Find the GCD of 286 and 503.

$$
\begin{array}{ll}
\gcd(503, 286) & 503 = 1 \cdot 286 + 217 \\
= \gcd(286, 217) & 286 = 1 \cdot 217 + 69 \\
= \gcd(217, 69) & 217 = 3 \cdot 69 + 10 \\
= \gcd(69, 10) & 69 = 6 \cdot 10 + 9 \\
= \gcd(10, 9) & 10 = 1 \cdot 9 + 1 \\
= 1 & 9 = 9 \cdot 1
\end{array}
$$

$$
\begin{aligned}
1 &= 10 - 1 \cdot 9 \\
1 &= 7 \cdot 10 - 1 \cdot 69 \\
1 &= 7 \cdot 217 - 22 \cdot 69 \\
1 &= 29 \cdot 217 - 22 \cdot 286 \\
1 &= 29 \cdot 503 - 51 \cdot 286
\end{aligned}
$$

- E

Number Theory and Cryptography

- Division, Primes

  $a = dq + r \qquad q = a \textit{ div } d \qquad r = a \textit{ mod } d$

- Congruence

  $a \equiv b \pmod{m}$ if m divides $a - b$

- Greatest Common Divisor (GCD)

  (extended) Euclidean algorithm
  find the modular inverse
  solve linear congruence $ax \equiv b \pmod{m}$ $(\gcd(a, m) = 1)$

- Euler's Theorem / Fermart's Little Theorem

Number Theory and Cryptography

- Division, Primes

  $a = dq + r \qquad q = a \textit{ div } d \qquad r = a \textit{ mod } d$

- Congruence

  $a \equiv b \pmod{m}$ if m divides $a - b$

- Greatest Common Divisor (GCD)

  (extended) Euclidean algorithm
  find the modular inverse
  solve linear congruence $ax \equiv b \pmod{m}$ $(\gcd(a, m) = 1)$
  Chinese Remainder Theorem / back substitution

- Euler's Theorem / Fermart's Little Theorem

Number Theory Summary

- Division, Primes

  $a = dq + r \quad q = a \ \mathit{div}\ d \quad r = a \ \mathit{mod}\ d$

- Congruence

  $a \equiv b \pmod{m}$ if m divides $a - b$

- Greatest Common Divisor (GCD)

  (extended) Euclidean algorithm
  find the modular inverse
  solve linear congruence $ax \equiv b \pmod{m}$ $(\gcd(a, m) = 1)$
  Chinese Remainder Theorem / back substitution

- Euler's Theorem / Fermart's Little Theorem

  $x^{\phi(n)} \equiv 1 \bmod n$ if $\gcd(x, n) = 1$
  $x^{p-1} \equiv 1 \bmod p$ if $x \not\equiv 0 \bmod p$

Modular Arithmetic in CS

- Modular arithmetic and congruencies are used in CS:
  - Pseudorandom number generators
  - Hash functions
  - Cryptography

Pseudorandom Number Generators

- *Linear congruential method*

  We choose four numbers:

  - the modulus m
  - multiplier a
  - increment c
  - seed x_0

Pseudorandom Number Generators

- *Linear congruential method*

  We choose four numbers:

  - the modulus m
  - multiplier a
  - increment c
  - seed x_0

  We generate a sequence of numbers $x_1, x_2, \ldots, x_n, \ldots$ with $0 \leq x_i < m$ by using the congruence

$$x_{n+1} = (ax_n + c) \pmod{m}$$

Pseudorandom Number Generators

- *Linear congruential method*

$$x_{n+1} = (ax_n + c) \pmod{m}$$

Pseudorandom Number Generators

- *Linear congruential method*

$$x_{n+1} = (ax_n + c) \pmod{m}$$

Example:

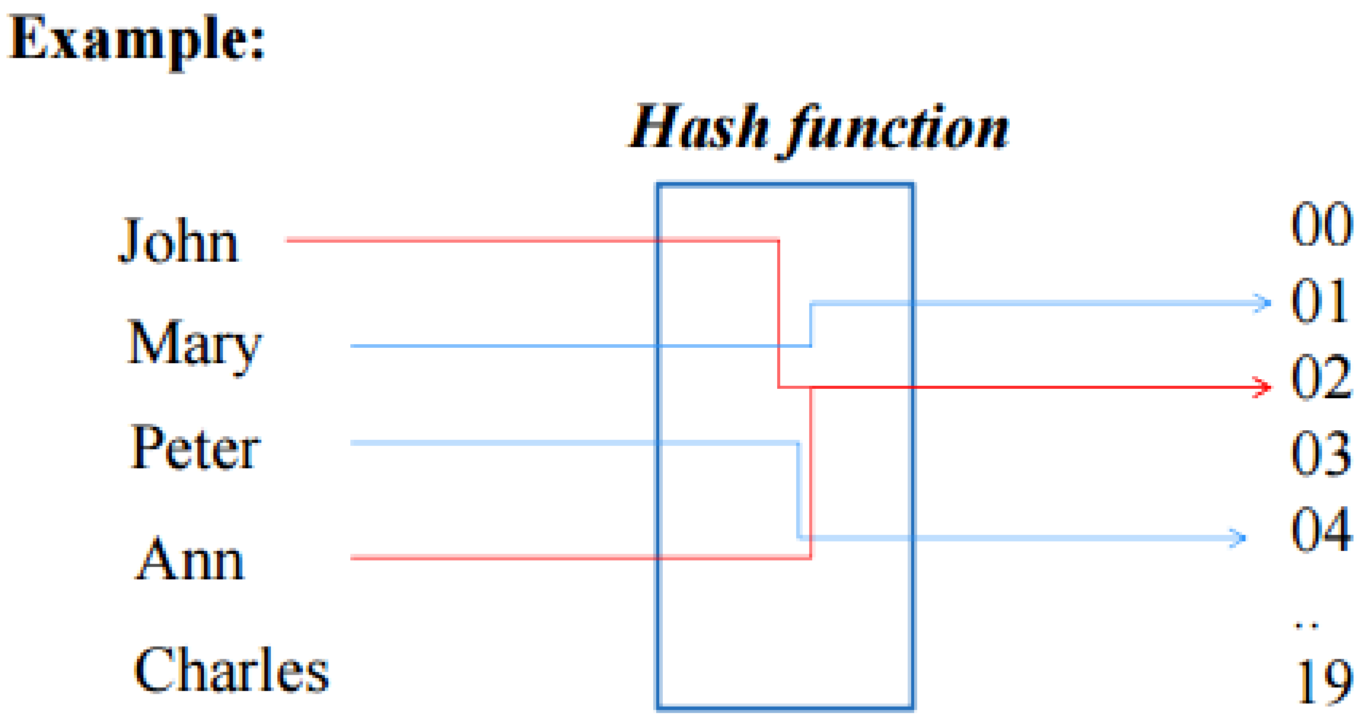
- Assume : m=9,a=7,c=4, $x_0 = 3$
- x_1= 7*3+4 mod 9=25 mod 9 =7
- x_2 = 53 mod 9 = 8
- x_3 = 60 mod 9 = 6
- x_4= 46 mod 9 =1
- x_5 = 11 mod 9 =2
- x_6 = 18 mod 9 =0
- ....

Hash Functions

- A *hash function* is an algorithm that maps data of arbitrary length to data of a fixed length. The values returned by a hash function are called *hash values* or hash codes.

Hash Functions

- A *hash function* is an algorithm that maps data of arbitrary length to data of a fixed length. The values returned by a hash function are called *hash values* or hash codes.



Hash Functions

- **Problem**: Given a large collection of records, how can we store and find a record quickly?

Hash Functions

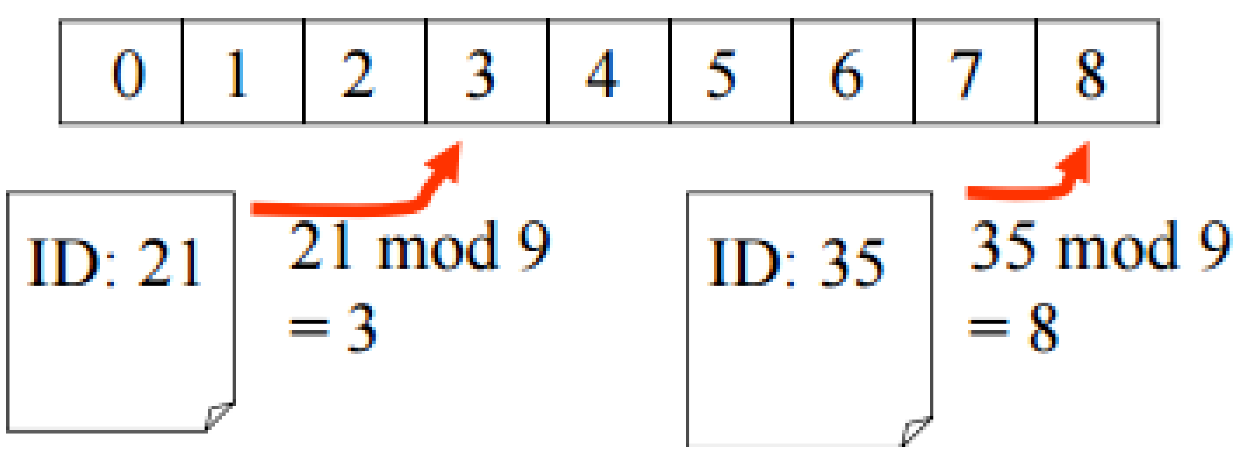
- **Problem**: Given a large collection of records, how can we store and find a record quickly?

  Solution: Use a hash function, calculate the location of the record based on the record's ID.

  Example: A common hash function is

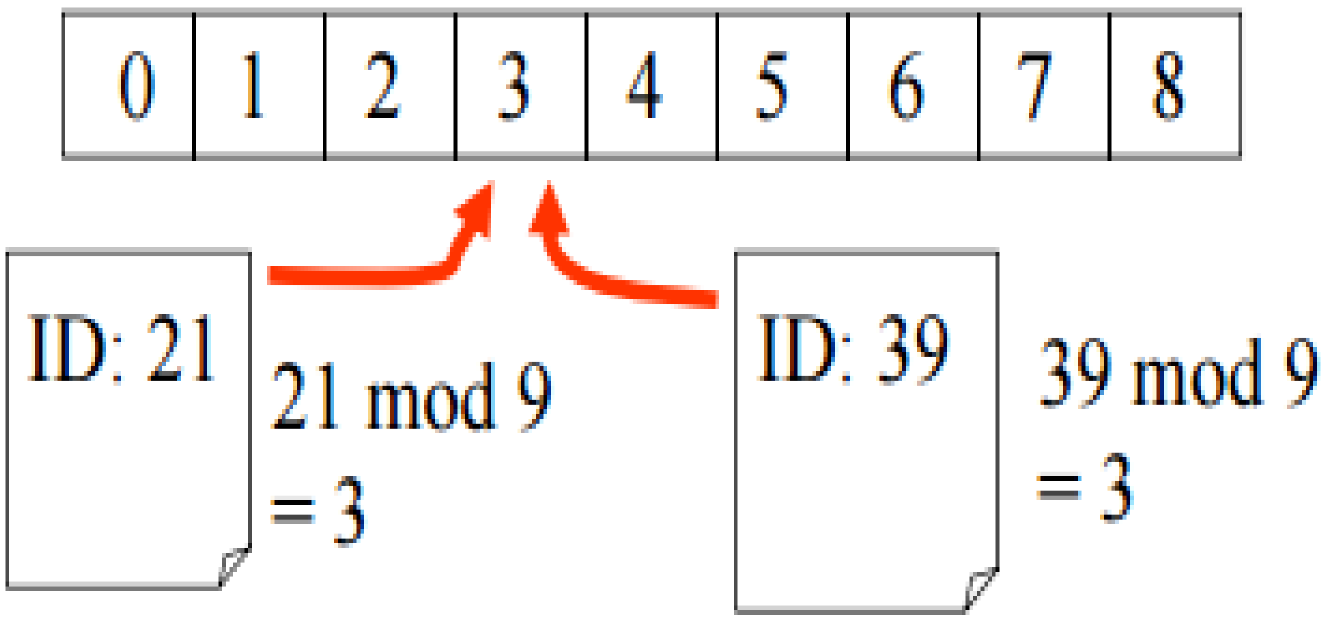
  - $h(k) = k \textbf{ mod } n$,

  where n is the number of available storage locations.



Hash Functions

- Two records mapped to the same location

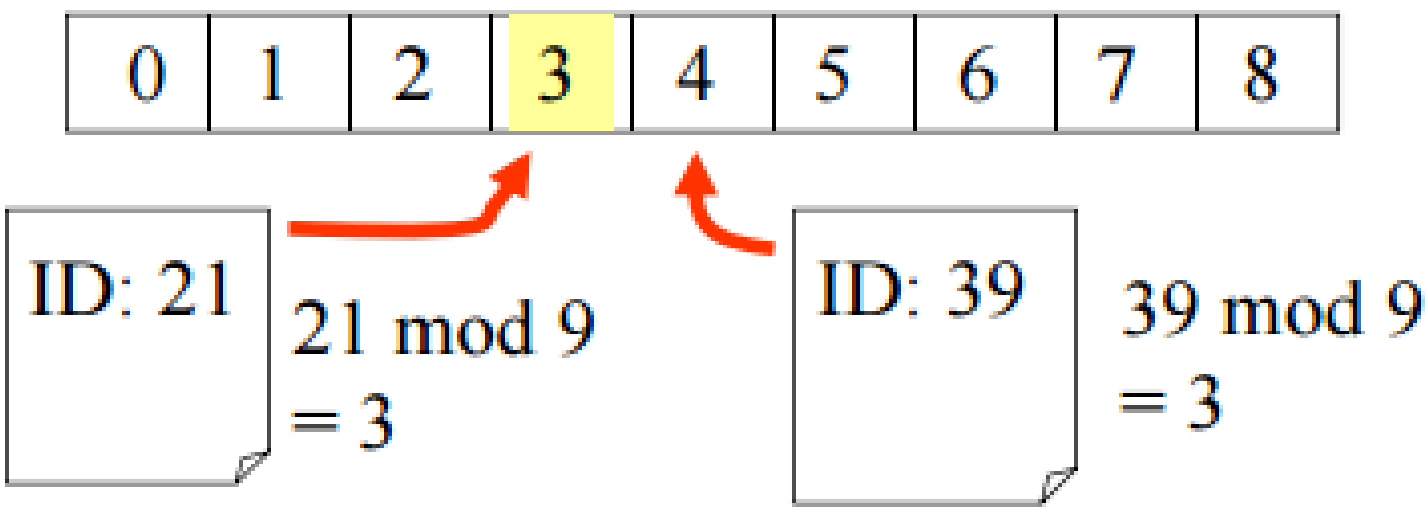


Hash Functions

- **Solution 1**: move to the next available location

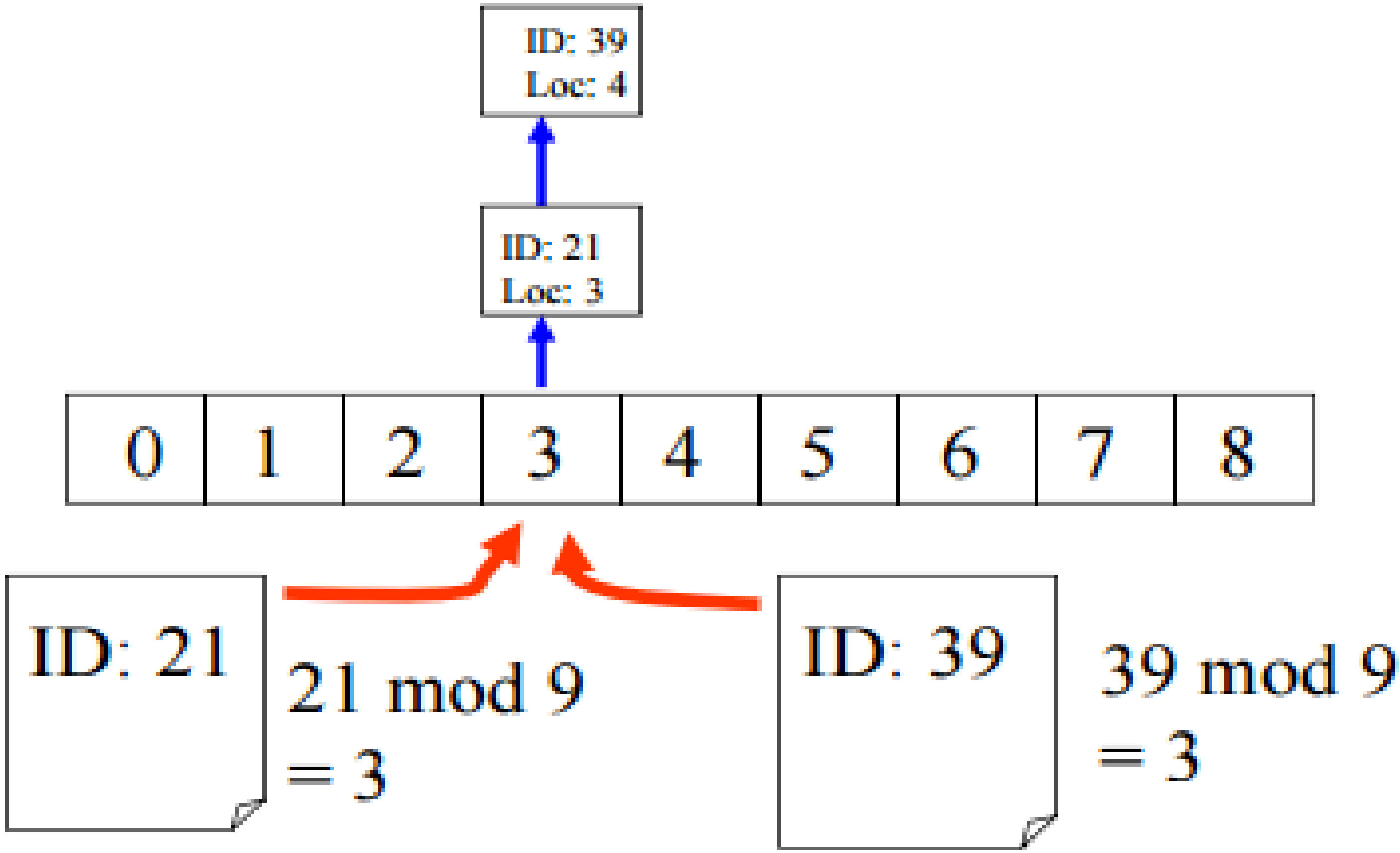
try

$$h_0(k) = k \textbf{ mod } n$$
$$h_1(k) = (k+1) \textbf{ mod } n$$
$$\dots$$
$$h_m(k) = (k+m) \textbf{ mod } n$$



Hash Functions

- **Solution 2**: remember the exact location in a secondary structure that is searched sequentially



Applications of Number Theory in Cryptography

- Introduction
- Symmetric cryptography
- Asymmetric cryptography
- RSA Cryptosystem
- DLP and El Gamal cryptography
- Diffie-Hellman key exchange protocol
- Crytocurrency, e.g., bitcoin

Cryptography

- History of almost 4000 years (from 1900 B.C.)

  Cryptography = kryptos + graphos

Cryptography

- History of almost 4000 years (from 1900 B.C.)

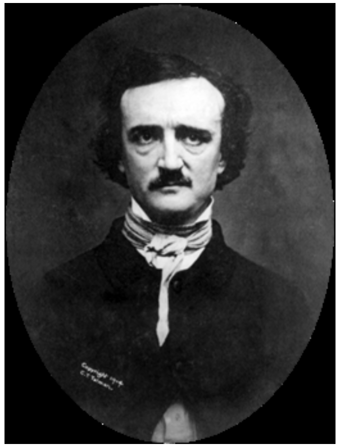
$$\text{Cryptography} = \underset{(\text{secret})}{\text{kryptos}} + \underset{(\text{writing})}{\text{graphos}}$$

Cryptography

- History of almost 4000 years (from 1900 B.C.)

Cryptography = kryptos + graphos
(secret) (writing)

The term was first used in *The Gold-Bug*, by Edgar Allan Poe (1809 - 1849).



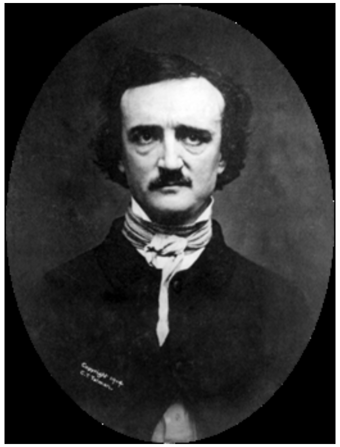
Cryptography

- History of almost 4000 years (from 1900 B.C.)

Cryptography = kryptos + graphos

(secret) (writing)

The term was first used in *The Gold-Bug*, by Edgar Allan Poe (1809 - 1849).



"Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve." – 1941

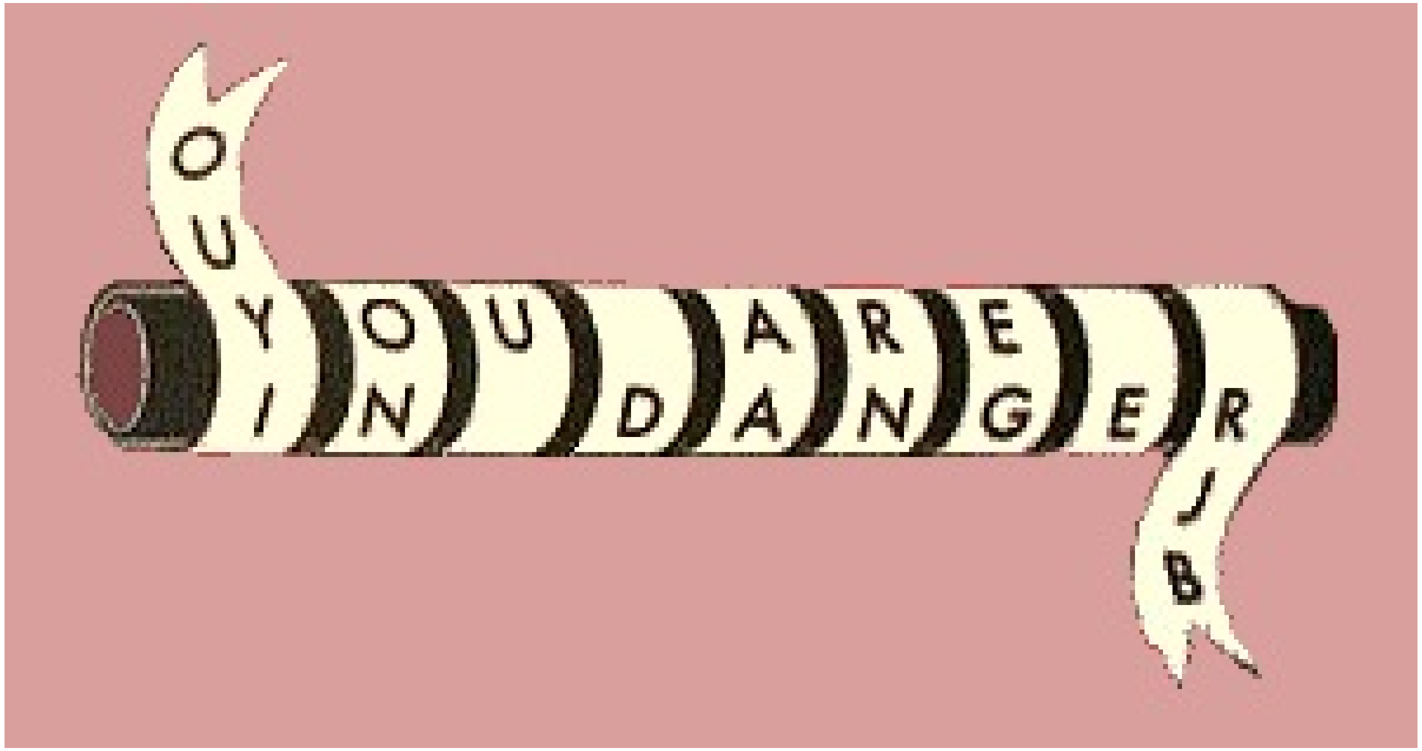
Cryptography

- One-sentence definition:

  "Cryptography is the practice and study of techniques for secure communication in the presence of third parties called *adversaries*." – Ronald L. Rivest

Some Examples

- In 405 BC, the Greek general LYSANDER OF SPARTA was sent a coded message written on the inside of a servant's belt.

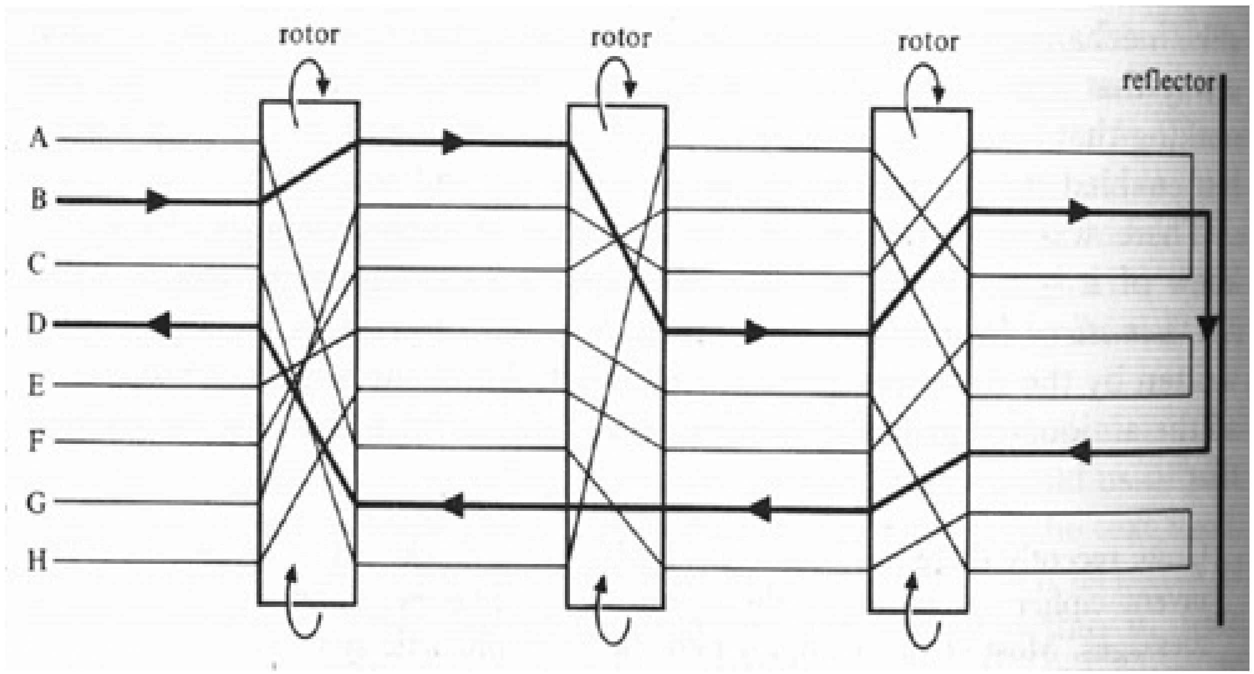


Some Examples

- The Greeks also invented a cipher which changed letters to numbers. A form of this code was still being used during *World War I*.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

- Enigma, Germany coding machine in *World War II.*



Cryptography History

- History (until 1970's)

  "*Symmetric*" cryptography

Cryptography History

- History (until 1970's)

  "*Symmetric*" cryptography

Cryptography History

- History (until 1970's)

  "*Symmetric*" cryptography

Cryptography History

- History (until 1970's)

  "*Symmetric*" cryptography

Cryptography History

- History (until 1970's)

  "*Symmetric*" cryptography

Cryptography History

- History (until 1970's)

  "*Symmetric*" cryptography

They need agree in advance on the secret key k.

Cryptography History

- History (until 1970's)

  "*Symmetric*" cryptography

They need agree in advance on the secret key k.

$\mathcal{Q}$: How can they do this?

- History (until 1970's)

"*Symmetric*" cryptography

They need agree in advance on the secret key k.

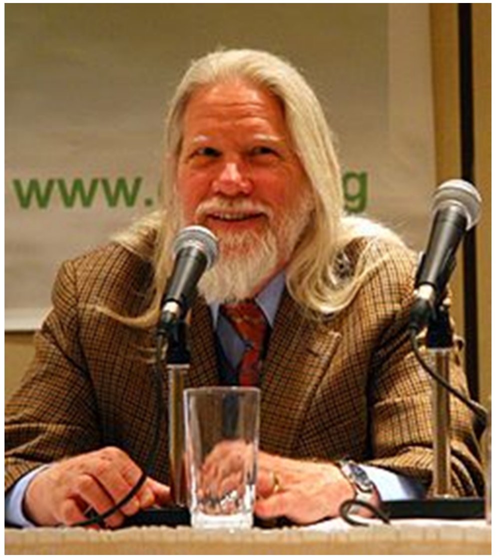
$\mathcal{Q}$: How can they do this?

$\mathcal{Q}$: What if Bob could send Alice a "special key" useful only for encryption but no help for decryption?

Cryptography History

- History (from 1976)
  - ◇ W. Diffie, M. Hellman, "New direction in cryptography", *IEEE Transactions on Information Theory, vol. 22, pp. 644-654, 1976.*

"We stand today on the brink of a revolution in cryptography."



Bailey W. Diffie

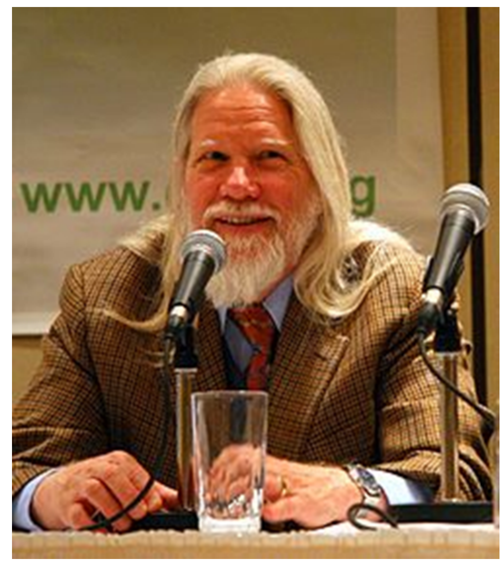
Martin E. Hellman

Cryptography History

- History (from 1976)
  - ◇ W. Diffie, M. Hellman, “New direction in cryptography”, *IEEE Transactions on Information Theory, vol. 22, pp. 644-654, 1976.*

“We stand today on the brink of a revolution in cryptography.”

2015 **Turing Award**



Bailey W. Diffie

Martin E. Hellman

2015	Martin E. Hellman Whitfield Diffie	For fundamental contributions to **modern cryptography**. Diffie and Hellman's groundbreaking 1976 paper, "New Directions in Cryptography,"[39] introduced the ideas of public-key cryptography and digital signatures, which are the foundation for most regularly-used security protocols on the internet today.[40]

Public Key Cryptography

- Alice wants to send a message to Bob

Public Key Cryptography

- Alice wants to send a message to Bob

Public Key Cryptography

- Alice wants to send a message to Bob

Public Key Cryptography

- Alice wants to send a message to Bob

Public Key Cryptography

- Alice wants to send a message to Bob

Ronald L. Rivest

Adi Shamir

Leonard M. Adleman

R. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol. 21-2, pages 120-126, 1978.

RSA Public Key Cryptosystem

- **R**ivest-**S**hamir-**A**dleman 2002 **Turing Award**

2002	Ronald L. Rivest, Adi Shamir and Leonard M. Adleman	For their ingenious contribution for making **public-key cryptography** useful in practice.

RSA Public Key Cryptosystem

- **R**ivest-**S**hamir-**A**dleman   2002 **Turing Award**

2002	Ronald L. Rivest, Adi Shamir and Leonard M. Adleman	For their ingenious contribution for making **public-key cryptography** useful in practice.

Pick two large primes, p and q. Let $n = pq$, then $\phi(n) = (p-1)(q-1)$. Encryption and decryption keys e and d are selected such that

- $\gcd(e, \phi(n)) = 1$
- $ed \equiv 1 \pmod{\phi(n)}$

RSA Public Key Cryptosystem

- **R**ivest-**S**hamir-**A**dleman   2002 **Turing Award**

2002	Ronald L. Rivest, Adi Shamir and Leonard M. Adleman	For their ingenious contribution for making **public-key cryptography** useful in practice.

Pick two large primes, p and q. Let $n = pq$, then $\phi(n) = (p-1)(q-1)$. Encryption and decryption keys e and d are selected such that

- $\gcd(e, \phi(n)) = 1$
- $ed \equiv 1 \pmod{\phi(n)}$

$C = M^e \bmod n$ (RSA **encryption)**

$M = C^d \bmod n$ (RSA **decryption)**

RSA Public Key Cryptosystem

- $C = M^e \bmod n$ (RSA **encryption)**

  $M = C^d \bmod n$ (RSA **decryption)**

Theorem (*Correctness*) : Let p and q be two odd primes, and define $n = pq$. Let e be relatively prime to $\phi(n)$ and let d be the multiplicative inverse of e modulo $\phi(n)$. For each integer x such that $0 \leq x < n$,

$$x^{ed} \equiv x \pmod{n}.$$

RSA Public Key Cryptosystem

- $C = M^e \bmod n$ (RSA **encryption)**

  $M = C^d \bmod n$ (RSA **decryption)**

Theorem (*Correctness*) : Let p and q be two odd primes, and define $n = pq$. Let e be relatively prime to $\phi(n)$ and let d be the multiplicative inverse of e modulo $\phi(n)$. For each integer x such that $0 \leq x < n$,

$$x^{ed} \equiv x \pmod{n}.$$

$\mathcal{Q}$: How to prove this?

RSA Public Key Cryptosystem: Example

Parameters:	p	q	n	$\phi(n)$	e	d
	5	11	55	40	7	23

RSA Public Key Cryptosystem: Example

Parameters:	p	q	n	$\phi(n)$	e	d
	5	11	55	40	7	23

Public key: $(7, 55)$

Private key: 23

RSA Public Key Cryptosystem: Example

Parameters:	p	q	n	$\phi(n)$	e	d
	5	11	55	40	7	23

Public key: $(7, 55)$

Private key: 23

Encryption: $M = 28$, $C = M^7 \bmod 55 = 52$

Decryption: $M = C^{23} \bmod 55 = 28$

RSA Public Key Cryptosystem: Parameters

Parameters: $p \quad q \quad n \quad \phi(n) \quad e \quad d$

Public key: (e, n)

Private key: d

p, q, $\phi(n)$ must be kept secret!

RSA Public Key Cryptosystem: Parameters

Parameters: $p \quad q \quad n \quad \phi(n) \quad e \quad d$

Public key: (e, n)

Private key: d

p, q, $\phi(n)$ must be kept secret!

$\mathcal{Q}$: Why?

RSA Public Key Cryptosystem: Parameters

Parameters: $p \quad q \quad n \quad \phi(n) \quad e \quad d$

Public key: (e, n)

Private key: d

p, q, $\phi(n)$ must be kept secret!

$\mathcal{Q}$: Why?

Comment: It is believed that determining $\phi(n)$ is equivalent to factoring n. Meanwhile, determining d given e and n, appears to be at least as time-consuming as the integer factoring problem.

RSA Public Key Cryptosystem: Parameters

Parameters: $p \quad q \quad n \quad \phi(n) \quad e \quad d$

Public key: (e, n)

Private key: d

p, q, $\phi(n)$ must be kept secret!

$\mathcal{Q}$: Why?

Comment: It is believed that determining $\phi(n)$ is equivalent to factoring n. Meanwhile, determining d given e and n, appears to be at least as time-consuming as the integer factoring problem.

The Security of the RSA

In practice, RSA keys are typically 1024 to 2048 bits long.

The Security of the RSA

In practice, RSA keys are typically 1024 to 2048 bits long.

Remark: There are some suggestions for choosing p and q.

A. Salomaa, *Public-Key Cryptography*, 2nd Edition, Springer, 1996, pp. 134-136.

The Security of the RSA

In practice, RSA keys are typically 1024 to 2048 bits long.

Remark: There are some suggestions for choosing p and q.

A. Salomaa, *Public-Key Cryptography*, 2nd Edition, Springer, 1996, pp. 134-136.

$\mathcal{Q}$: Consider the RSA system, where $n = pq$ is the modulus. Let (e, d) be a key pair for the RSA. Define

$$\lambda(n) = \mathrm{lcm}(p-1, q-1)$$

and compute $d' = e^{-1} \bmod \lambda(n)$. Will decryption using d' instead of d still work?

Applications of RSA

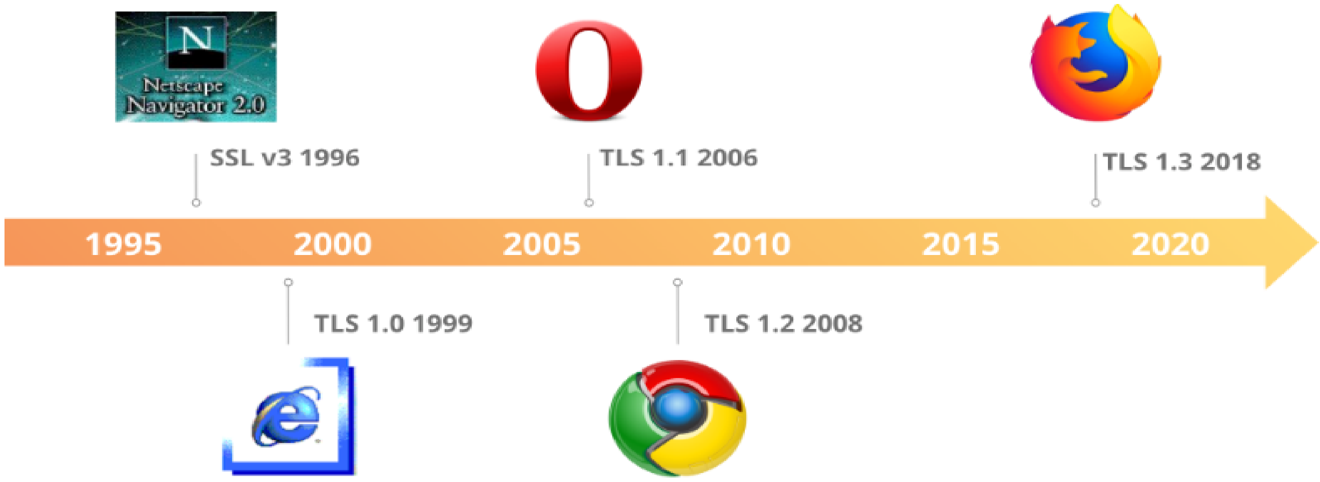
- SSL/TLS protocol

Applications of RSA

- SSL/TLS protocol

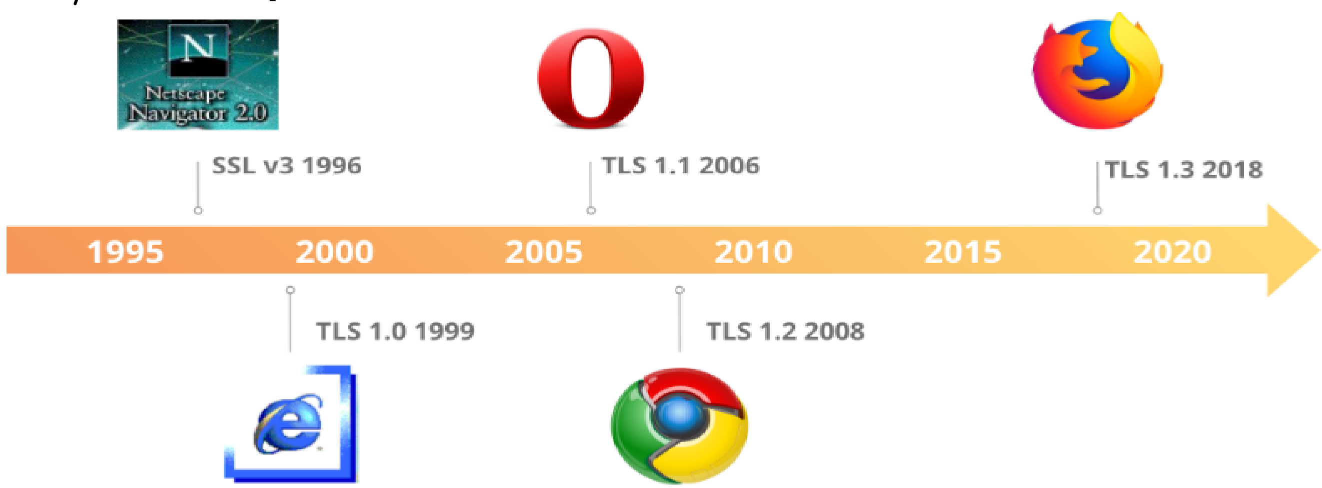
Applications of RSA

- SSL/TLS protocol



Applications of RSA

- SSL/TLS protocol

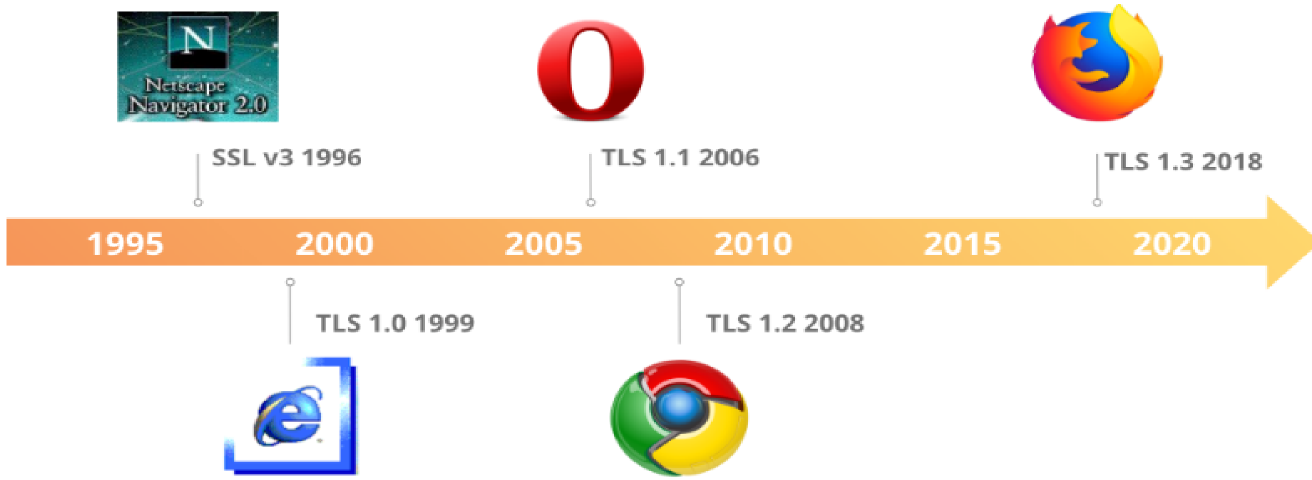


Key exchange/agreement and authentication

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
RSA	Yes	Yes	Yes	Yes	Yes	No
DH-RSA	No	Yes	Yes	Yes	Yes	No
DHE-RSA (forward secrecy)	No	Yes	Yes	Yes	Yes	Yes
ECDH-RSA	No	No	Yes	Yes	Yes	No
ECDHE-RSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes

Applications of RSA

- SSL/TLS protocol

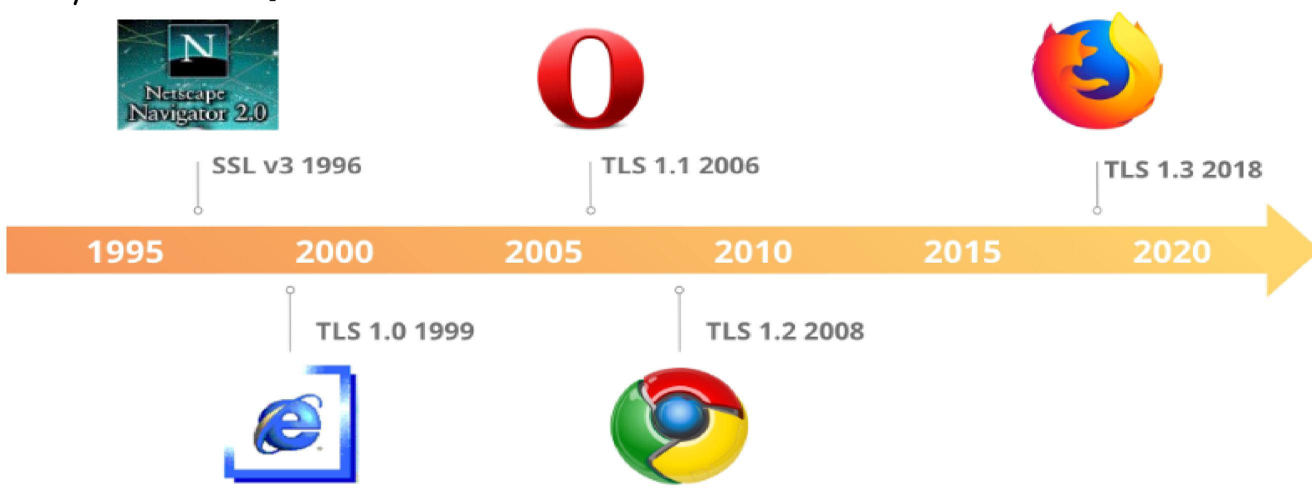


Key exchange/agreement and authentication

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
RSA	Yes	Yes	Yes	Yes	Yes	No
DH-RSA	No	Yes	Yes	Yes	Yes	No
DHE-RSA (forward secrecy)	No	Yes	Yes	Yes	Yes	Yes
ECDH-RSA	No	No	Yes	Yes	Yes	No
ECDHE-RSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes

Applications of RSA

- SSL/TLS protocol



Key exchange/agreement and authentication

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
RSA	Yes	Yes	Yes	Yes	Yes	No
DH-RSA	No	Yes	Yes	Yes	Yes	No
DHE-RSA (forward secrecy)	No	Yes	Yes	Yes	Yes	Yes
ECDH-RSA	No	No	Yes	Yes	Yes	No
ECDHE-RSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes

Using RSA for Digital Signature

$S = M^d \bmod n$ (RSA **signature**)

$M = S^e \bmod n$ (RSA **verification**)

Why?

- **The discrete logarithm** of an integer y to the base b is an integer x, such that

$$b^x \equiv y \bmod n.$$

- **The discrete logarithm** of an integer y to the base b is an integer x, such that

$$b^x \equiv y \bmod n.$$

Discrete Logarithm Problem:
Given n, b and y, find x.

The Discrete Logrithm

- **The discrete logarithm** of an integer y to the base b is an integer x, such that

$$b^x \equiv y \bmod n.$$

Discrete Logarithm Problem:

Given n, b and y, find x.

This is very hard!

El Gamal Encryption

- **Setup** Let p be a prime, and g be a generator of $\mathbb{Z}_p$. The private key x is an integer with $1 < x < p - 2$. Let $y = g^x \bmod p$. The public key for *El Gamal encryption* is (p, g, y).

- **Setup** Let p be a prime, and g be a generator of $\mathbb{Z}_p$. The private key x is an integer with $1 < x < p-2$. Let $y = g^x \bmod p$. The public key for *El Gamal encryption* is (p, g, y).

  El Gamal Encryption: Pick a random integer k from $\mathbb{Z}_{p-1}$,

  $$a = g^k \bmod p$$
  $$b = My^k \bmod p$$

  The ciphertext C consists of the pair (a, b).

  El Gamal Decryption:

  $$M = b(a^x)^{-1} \bmod p$$

Using El Gamal for Digital Signature

$$a = g^k \bmod p$$
$$b = k^{-1}(M - xa) \bmod (p-1)$$

(El Gamal **signature**)

$$y^a a^b \equiv g^M \pmod p$$

(El Gamal **verification**)

Using El Gamal for Digital Signature

$$a = g^k \bmod p$$
$$b = k^{-1}(M - xa) \bmod (p-1)$$

(El Gamal **signature**)

$$y^a a^b \equiv g^M \pmod{p}$$

(El Gamal **verification**)

Q : How to verify it?

An Example

Choose $p = 2579$, $g = 2$, and $x = 765$. Hence $y = 2^{765} \bmod 2579 = 949$.

An Example

Choose $p = 2579$, $g = 2$, and $x = 765$. Hence $y = 2^{765} \bmod 2579 = 949$.

- **(Public key)** $k_e = (p, g, y) = (2579, 2, 949)$
- **(Private key)** $k_d = x = 765$

An Example

Choose $p = 2579$, $g = 2$, and $x = 765$. Hence $y = 2^{765} \bmod 2579 = 949$.

- **(Public key)** $k_e = (p, g, y) = (2579, 2, 949)$
- **(Private key)** $k_d = x = 765$

Encryption: Let $M = 1299$ and choose a random $k = 853$,

$$\begin{aligned}(a, b) &= (g^k \bmod p, My^k \bmod p) \\ &= (2^{853} \bmod 2579, 1299 \cdot 949^{853} \bmod 2579) \\ &= (435, 2396).\end{aligned}$$

Decryption:

$$M = b(a^x)^{-1} \bmod p = 2396 \times (435^{765})^{-1} \bmod 2579 = 1299.$$

Question 1: Is it feasible to derive x from (p, g, y)?

Question 1: Is it feasible to derive x from (p, g, y)?

It is equivalent to solving the DLP. It is believed that there is NO polynomial-time algorithm. p should be large enough, typically 160 bits.

Question 1: Is it feasible to derive x from (p, g, y)?

It is equivalent to solving the DLP. It is believed that there is NO polynomial-time algorithm. p should be large enough, typically 160 bits.

Question 2: Given a ciphertext (a, b), is it feasible to derive the plaintext M?

Question 1: Is it feasible to derive x from (p, g, y)?

It is equivalent to solving the DLP. It is believed that there is NO polynomial-time algorithm. p should be large enough, typically 160 bits.

Question 2: Given a ciphertext (a, b), is it feasible to derive the plaintext M?

Attack 1: Use $M = by^{-k}$. However, k is randomly picked.

Attack 2: Use $M = b(a^x)^{-1} \bmod p$, but x is secret.

Diffie-Hellman Key Exchange Protocol

User A		User B
Generate random $X_A < p$ calculate $Y_A = \alpha^{X_A} \bmod p$	$\xrightarrow{Y_A}$ $\xleftarrow{Y_B}$	Generate random $X_B < p$ Calculate $Y_B = \alpha^{X_B} \bmod p$
Calculate $k = (Y_B)^{X_A} \bmod p$		Calculate $k = (Y_A)^{X_B} \bmod p$

Next Lecture

- induction ...