Hash, DH and RSA

Short Version

Chun-Jen Chung

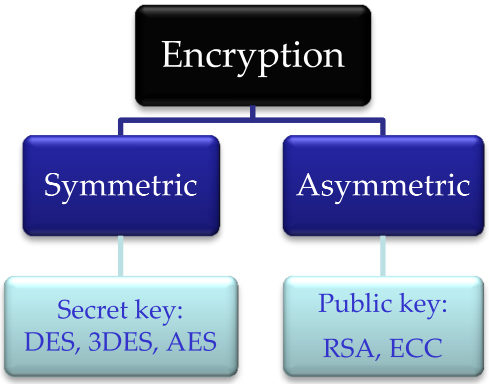
Arizona State University

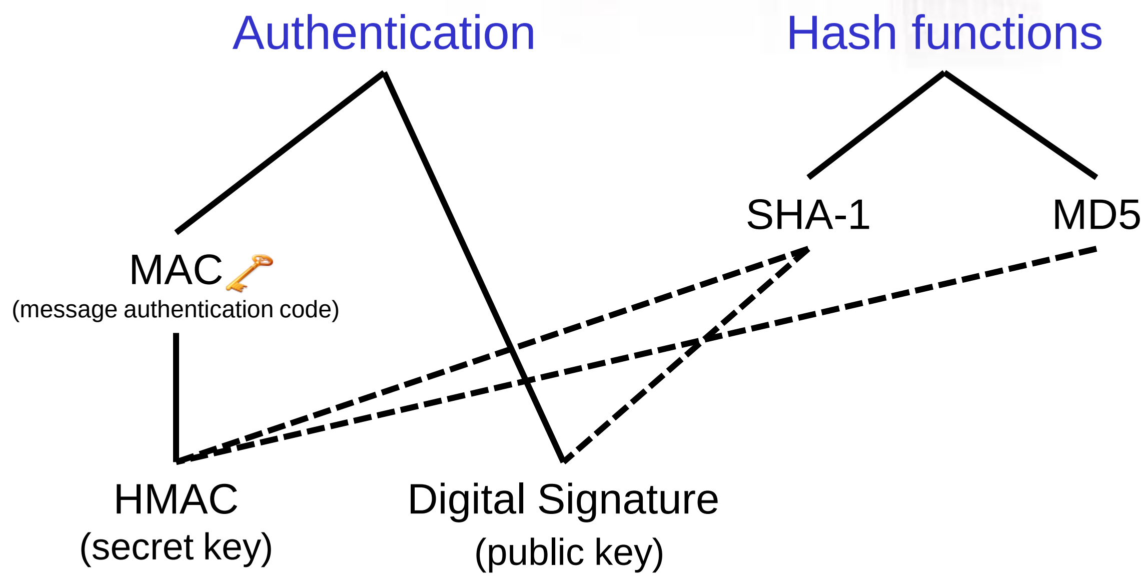
Outline

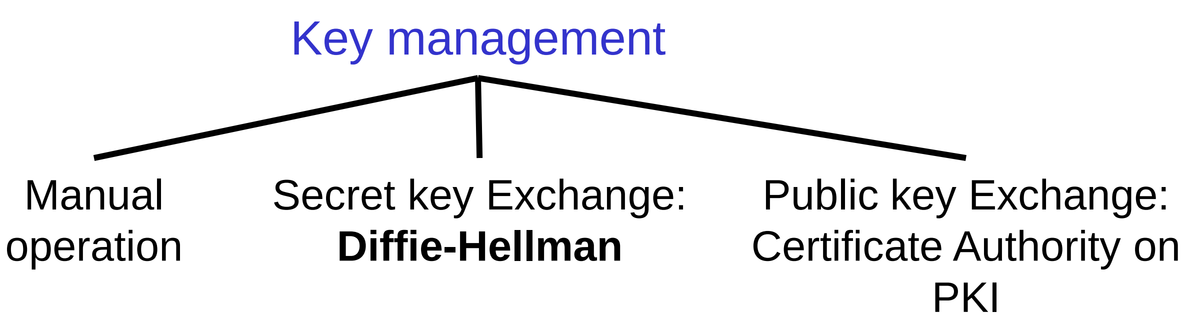
- Background
- Hash Functions
- Public key cryptography (PKC)
  - DH
  - RSA
- Summary

Background

Crypto algorithms review







Introduction to Hash Functions

Hash Algorithms

Message of arbitrary length → **hash H** (Math transformation) → A fixed-length short message

- Also known as
  - (Cryptographic) Hash functions
  - Message digests
  - One-way transformations
  - One-way functions
- Length of $H(m)$ much shorter than length of m
- Usually fixed lengths: 128 or 160 bits
- Example algorithms
  - MD5 (**Message-Digest**) – 128 bits output
  - SHA-1 (**secure hash algorithm**) : 160 bits output
  - SHA-2: 256/224, 512/384

Hash Algorithms (cont'd)

User password

8 bytes

One way hash (SHA-1)

43 B0 4C 54 3B
67 A2 23 3F 7D
36 2B 7A 2B 49
3C D3 AF 27 4A

Hash value 20 bytes (160 bits)

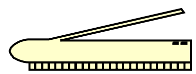


Image from scanner

512 K bytes

One way hash (SHA-1)

73 BF 4C 34 3B
67 A2 45 23 76
3F 76 D2 37 F6
44 47 8F 93 D2

Hash value 20 bytes

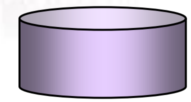
All files of a floppy disk

1.4 M bytes

One way hash (SHA-1)

54 3B 4C 34 3B
62 3C D3 AF A2
45 67 A2 23 3F
7D 43 B0 4C 19

Hash value 20 bytes



All files of a hard disk

80Giga bytes

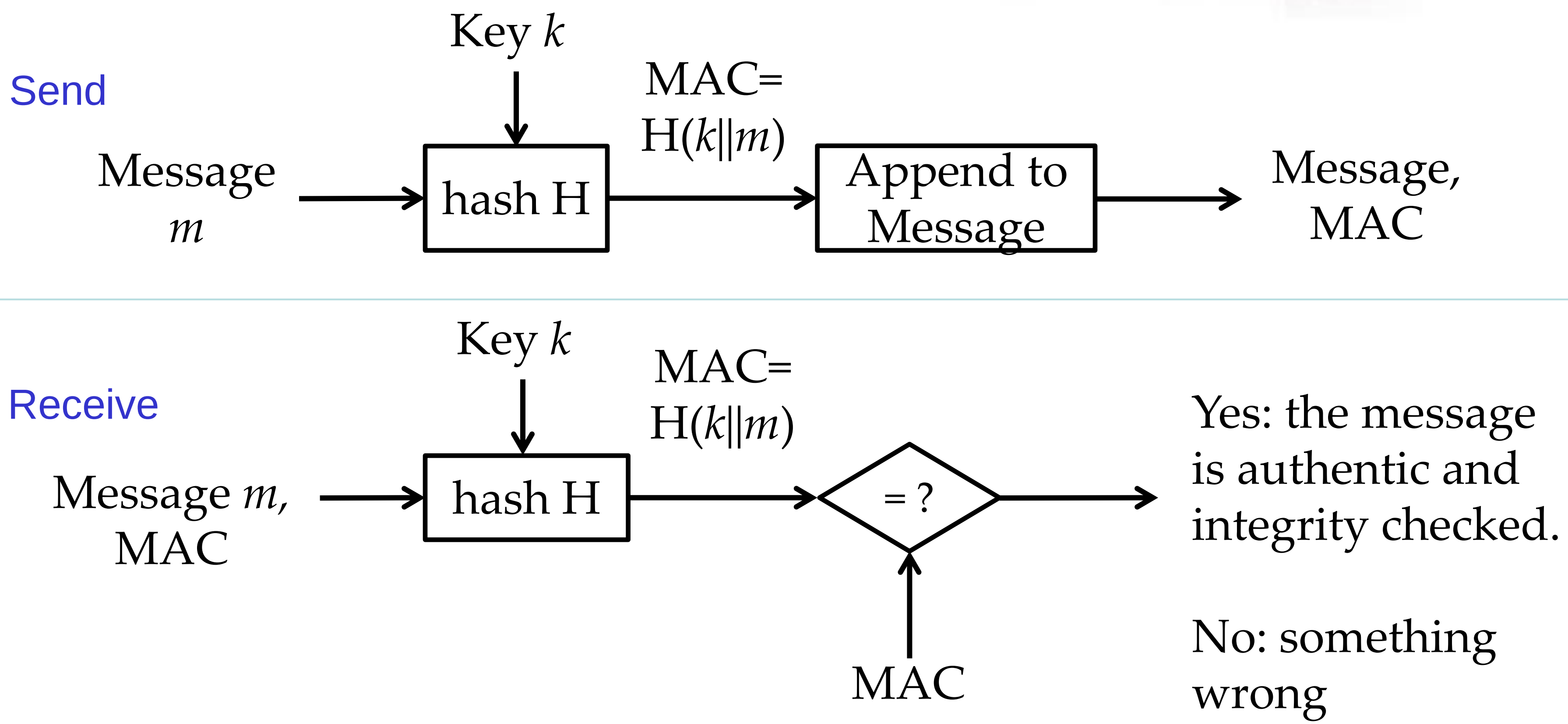
One way hash (SHA-1)

32 2B 23 70 7A
2B 4F 43 B0 4C
54 3B 49 28 67
A2 23 8F 7D 36

Hash value 20 bytes

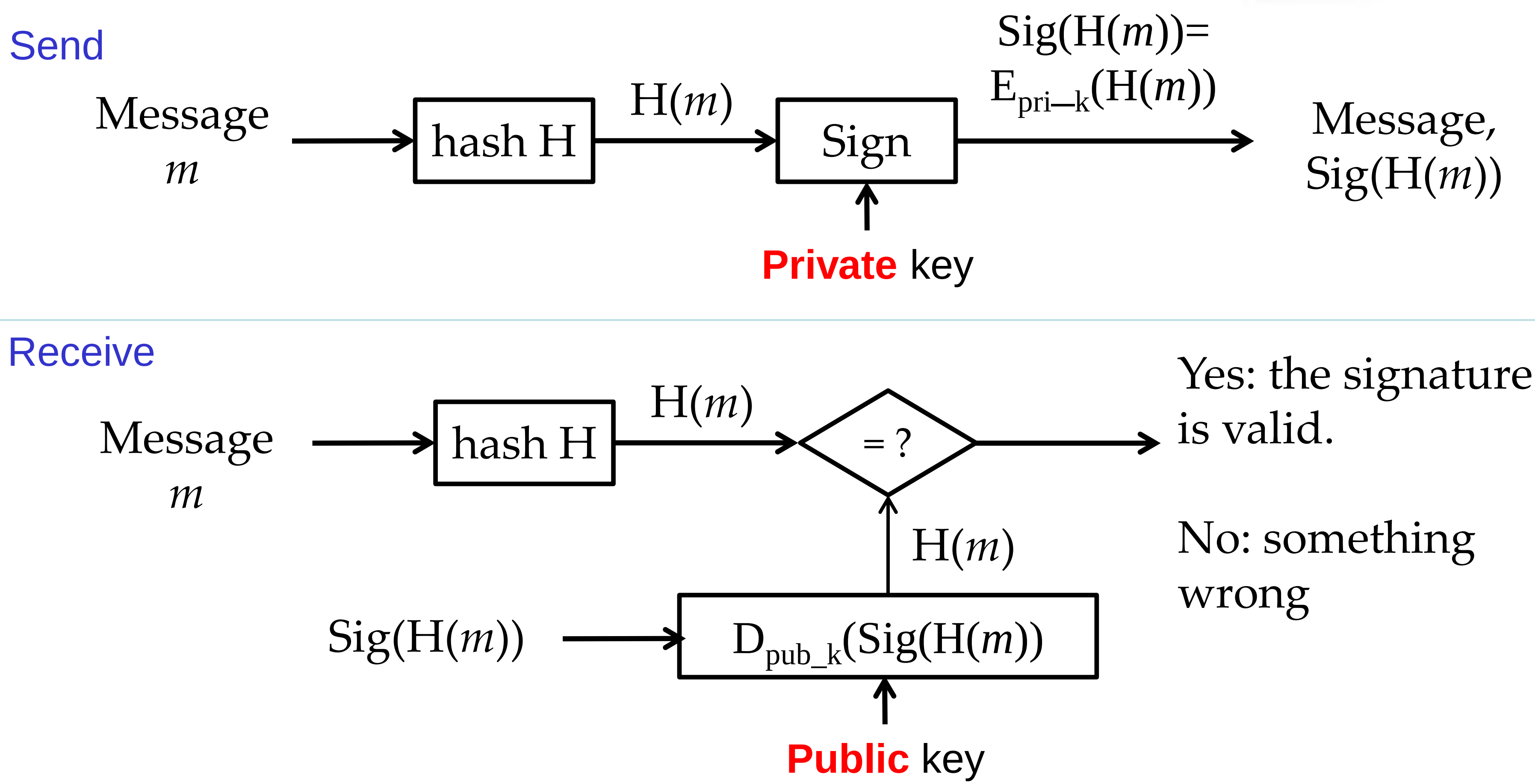
Applications of Hash Functions

- Message Authentication Code (keyed hash)

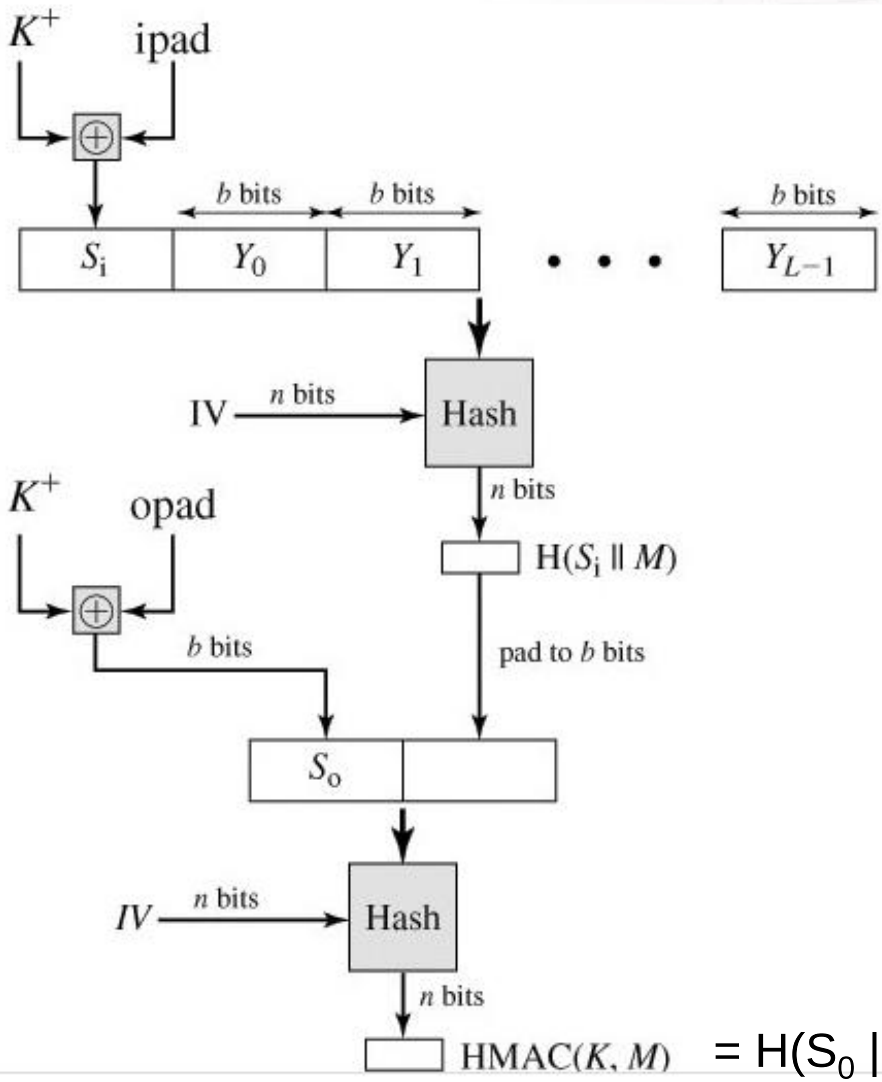


Applications of Hash Functions

- Generate/verify digital signature

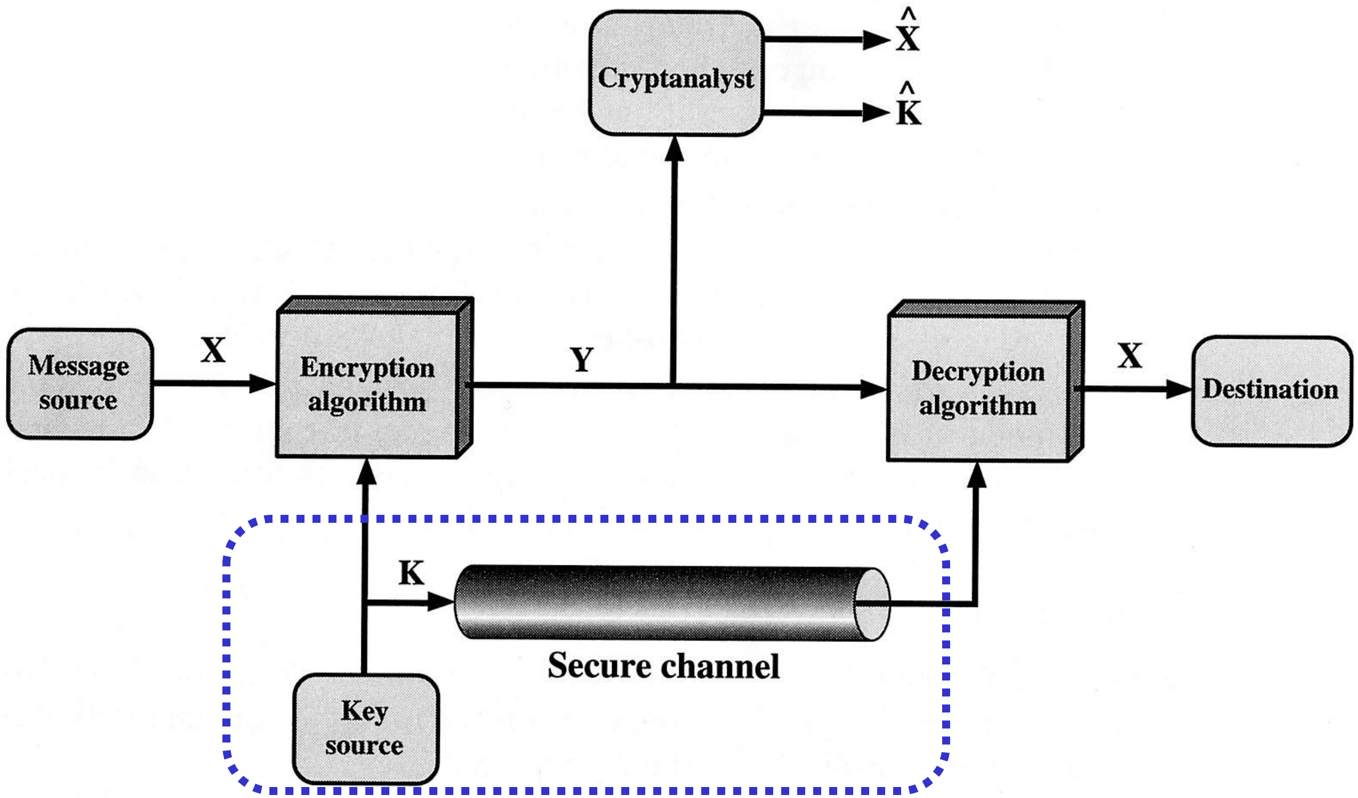


HMAC



$= H(S_0 \,||\, H(S_i \,||\, M))$

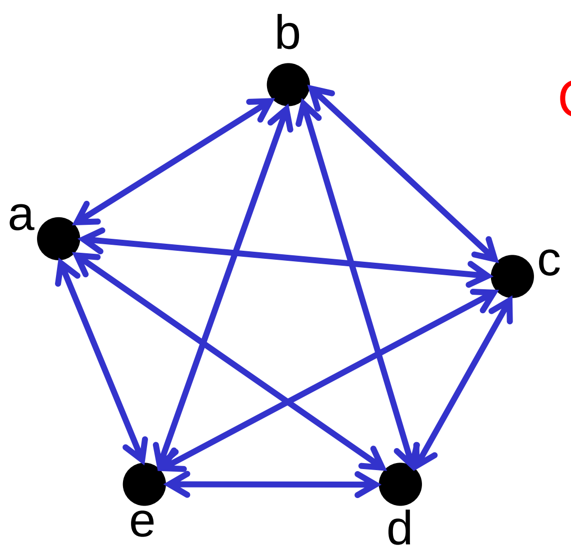
Problem of Symmetric key Crypto



How can you share a secret key securely?

Problem of Symmetric key Crypto

- Sharing key in Secret key cryptosystem
  - Given complete graph with n nodes (entities), ${}_nC_2 = n(n-1)/2$ pairs secret keys are required.
  - Ex.) If n=100, 99x50=4,950 keys
  - Problem: managing large number of secret keys is difficult. (e.g., all ASU students? Keys are lost? add new members? remove new member)



Q: how many different secret keys for five students?

A: Secret keys are required between

(a,b), (a,c), (a,d), (a,e), (b,c), (b,d), (b,e) (c,d), (c,e), (d,e)

Crypto algorithms review

- Encryption
  - Symmetric
    - Secret key: DES, 3DES, AES
  - Asymmetric
    - Public key: RSA, ECC

- Authentication
  - MAC (message authentication code)
    - HMAC (secret key)
  - Digital Signature (public key)

- Hash functions
  - SHA-1
  - MD5

- Key management
  - Manual operation
  - Secret key Exchange: **Diffie-Hellman**
  - Public key Exchange: Certificate Authority on PKI

DLP (Discrete logarithm problem)

	Exponentiation	Logarithm
P (Polynomial problem)	$g^x = Y \rightarrow x = \log_g Y$	
	e.g., $10^x = 10{,}000{,}000{,}000$: $x=10$	
NP (Nondeterministic Polynomial problem)	Modular Exponentiation	Discrete Logarithm (**Hard**)
	$g^x \bmod p = Y \rightarrow x=?$	
	e.g., $10^x \bmod 19 = 9$: $x=10$	

Q: how difficult?

An example

- Q: $7^x \bmod 13 = 8$, $x=?$
- A:
  - $x=0 \rightarrow 7^0 \bmod 13 = 1$
  - $x=1 \rightarrow 7^1 \bmod 13 = 7$
  - $x=2 \rightarrow 7^2 \bmod 13 = 10$
  - $x=3 \rightarrow 7^3 \bmod 13 = 5$
  - $x=4 \rightarrow 7^4 \bmod 13 = 9$
  - $x=5 \rightarrow 7^5 \bmod 13 = 11$
  - $x=6 \rightarrow 7^6 \bmod 13 = 12$
  - $x=7 \rightarrow 7^7 \bmod 13 = 6$
  - $x=8 \rightarrow 7^8 \bmod 13 = 3$
  - **$x=9 \rightarrow 7^9 \bmod 13 = 8$**

How difficult??

Brute Force:
It would take p steps at least.
What if prime p is a large number with at least 512 bits?

P & NP problem

- P problem (Polynomial problem):
  - fast solutions exist
- NP problem (Nondeterministic Polynomial problem):
  - Fast solutions do not exist.
  - As input increases, time to solve the problem increases exponentially
  - But validation (Yes/No) of the answer can be done quickly
- NP-complete problem:
  - Most hard problem among NP problems.

DLP (Discrete logarithm problem)

- Problem:
  - Given g, y, and prime p, find an integer x, if any, such that $y = g^x \bmod p$

Given g, x, p $\xrightarrow{\text{easy}}$ $y=g^x \bmod p$	Modular Exponentiation
x $\xleftarrow{\text{hard}}$ Given g, y, p	Discrete Logarithm

- Application:
  - Used to construct Diffie-Hellman & ElGamal-type public systems: DH, DSA (Digital Signature Algorithm), …

Diffie and Hellman key exchange

Diffie and Hellman (DH) key exchange

- Diffie-Hellman is a public key distribution scheme
- First public-key type scheme, proposed in 1976.

Whitfield Diffie

Martin Hellman

Diffie, W., and Hellman, M. New directions in cryptography. IEEE Trans. Inform. Theory IT-22, (Nov. 1976), 644-654.

DH Applications

- DH is currently used in many protocols, namely:
  - Internet Protocol Security (IPSec)
    - Internet Key Exchange (IKE)
  - **Secure Sockets Layer (SSL)**/Transport Layer Security (TLS)
    - Key agreement; in conjunction with DES (40-bit key) or 3-DES (128-bit key)
  - Secure Shell (SSH)
  - Public Key Infrastructure (PKI)

HTTP	FTP	SMTP
TCP		
IPsec		
IP		

At the network layer

HTTP	FTP	SMTP
SSL/TLS		
TCP		
IP		

At the presentation layer

SSH	PGP
FTP	SMTP
TCP	
IP	

At the application layer

Source: Riley Lochridge, The Diffie-Hellman Algorithm, 2003

DH key agreement protocol

- Allows two users to exchange a secret key
- Requires no prior secrets
- Real-time over an *untrusted* network
- Based on the difficulty of computing discrete logarithms of large numbers.
- Requires two large numbers:
  - p: one prime
  - g: a primitive root of p (or a base), e.g: 3 is a primitive root modulo 7, why?
  - x: a secret key

$$y = g^x \, mod \, p$$

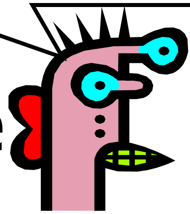
g is a **generator** of a group G if every element in G can be expressed as the product of finitely many powers of *g*.

DH Key Exchange Protocol

F={1,2,3,…,p-1}

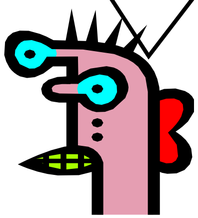
(1) Pick secret, random a from F

Alice



(2) $X = g^{a} \bmod p$ →

(3) $Y = g^{b} \bmod p$ ←



Bob

(1) Pick secret, random b from F

(4) Compute

$$k = Y^{a} \bmod p = (g^{b})^{a} \bmod p = g^{ab} \bmod p$$

(4) Compute

$$k = X^{b} \bmod p = (g^{a})^{b} \bmod p = g^{ab} \bmod p$$

Eve has to compute g^{ab} from g^{a} and g^{b} without knowing a and b…
She faces the Discrete Logarithm Problem in finite fields

Source: D. Mukhopadhyay, ECC, IITM

DH example

- Alice and Bob get public numbers
  - g = 2, p = 3

$$X = g^a \bmod p$$

$$Y = g^b \bmod p$$

- Alice and Bob compute public values with their private key a=4, b=3 respectively
  - $X = 2^4 \bmod 3 = 16 \bmod 3 = 1$
  - $Y = 2^3 \bmod 3 = 8 \bmod 3 = 2$
- Alice and Bob exchange public numbers

Q: How?

DH Example (cont'd)

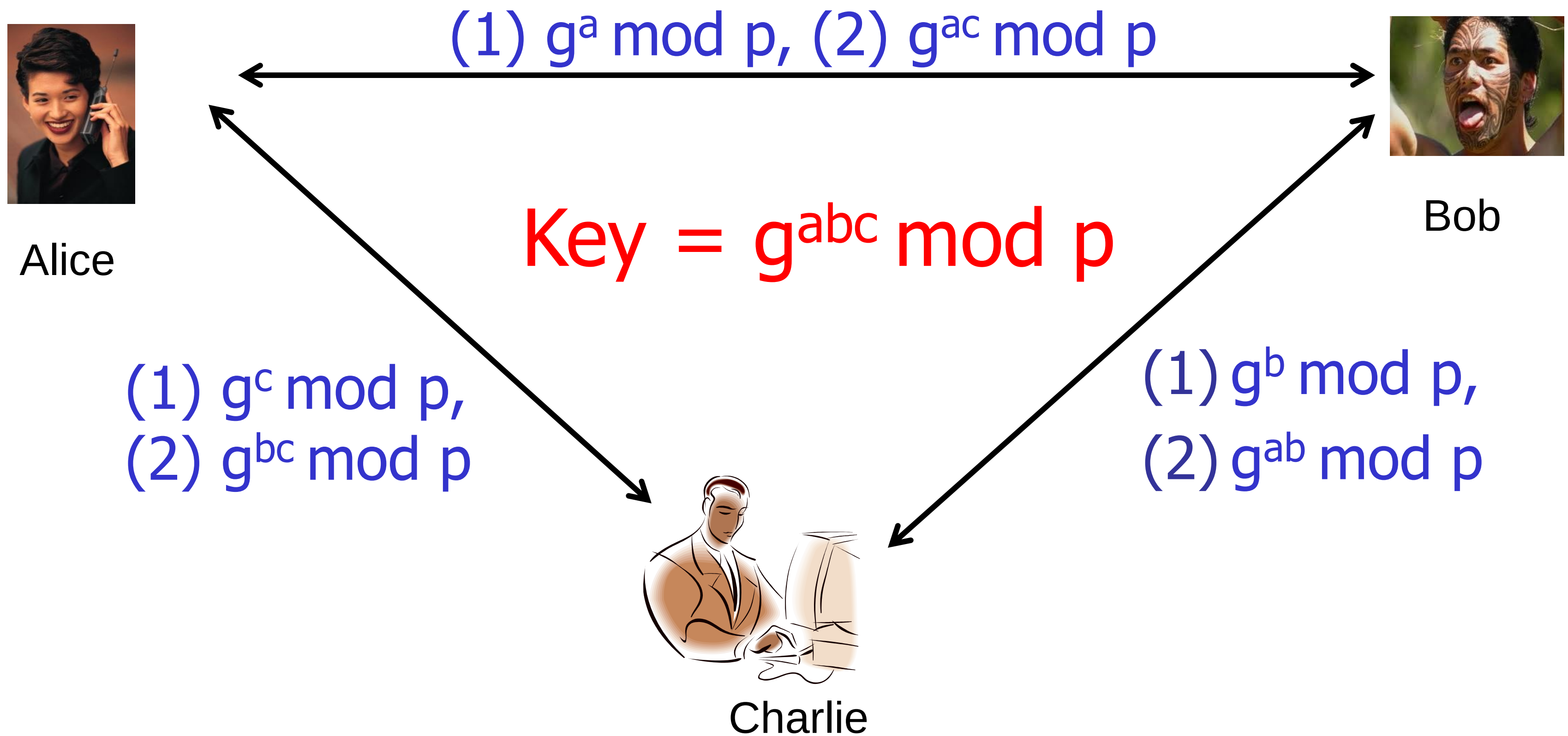
$X = 2^4 \bmod 3 = 16 \bmod 3 = 1$

$Y = 2^3 \bmod 3 = 8 \bmod 3 = 2$

- Alice and Bob compute symmetric keys
  - $k_a = Y^a \bmod p = 2^4 \bmod 3 = 1$
  - $k_b = X^b \bmod p = 1^3 \bmod 3 = 1$

DH for three parties

- How can **thre**e persons (Alice, Bob, Charlie) share a common secret key using DH key exchange?



MITM attack in DH Scheme (formal)

Alice

x_a : private

$Y_a = g^{x_a}$: public

x_b : private

$Y_b = g^{x_b}$: public

Bob

$Y_a \longrightarrow$

$\longleftarrow Y_c$

$$Y_c = g^{x_c} \quad \textit{for some } x_c$$

$Y_c \longrightarrow$

$\longleftarrow Y_b$

Alice computes the session key

$$K_a = Y_c^{x_a} = g^{x_c x_a}$$

C (MITM)

Bob computes the session key

$$K_b = Y_c^{x_b} = g^{x_c x_b}$$

Adversary computes both **session** keys

$$K_b = Y_b^{x_c} = g^{x_c x_b}$$

$$K_a = Y_a^{x_c} = g^{x_c x_a}$$

Q: why this happens?

Man-in-the middle attack comes from no authentication

Use authentication (e.g., signature)

A Possible Solution

A $\qquad\qquad\qquad\qquad$ B

$A \xrightarrow{g^a} B$

$A \xleftarrow{B,\ g^b,\ SIG_B(g^a, g^b, g^{ab})} B$

$A \xrightarrow{A,\ SIG_A(g^b, g^a, g^{ab})} B$

Is this protocol secure?

E

$E \xrightarrow{E,\ SIG_E(g^b, g^a, g^{ab})} B$

Identity misbinding attack

Source: Hugo Krawczyk, Design and Analysis of Authenticated Diffie-Hellman Protocols, 2003

A Possible Solution (ISO-9796)

A B

A, g^a →

← $B, g^b, SIG_B(g^a, g^b, A)$

$SIG_A(g^b, g^a, B)$ →

Thwarts the identity-misbinding attack by including the identity of the peer under the signature

Source: Hugo Krawczyk, Design and Analysis of Authenticated Diffie-Hellman Protocols, 2003

The ISO defense

A $\xrightarrow{A,\ g^a}$ E $\xrightarrow{E,\ g^a}$ B

A $\xleftarrow{B,\ g^b,\ SIG_B(g^a, g^b, E)}$ E $\xleftarrow{B,\ g^b,\ SIG_B(g^a, g^b, E)}$ B

A: aha! B is talking to E not to me!

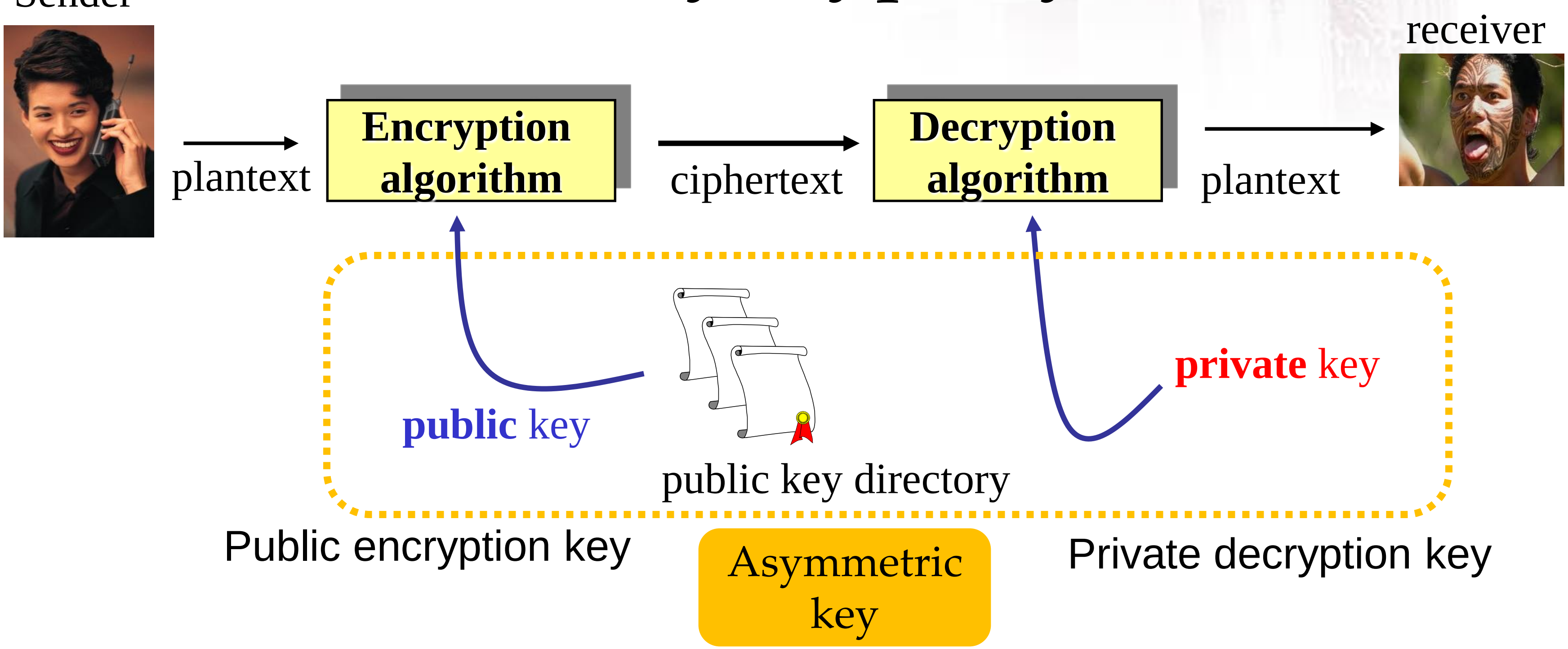
Note that E cannot produce $SIG_B(g^a, g^b, A)$

- The ISO protocol thus avoids the misbinding attack

Source: Hugo Krawczyk, Design and Analysis of Authenticated Diffie-Hellman Protocols, 2003

RSA

Public key cryptosystem

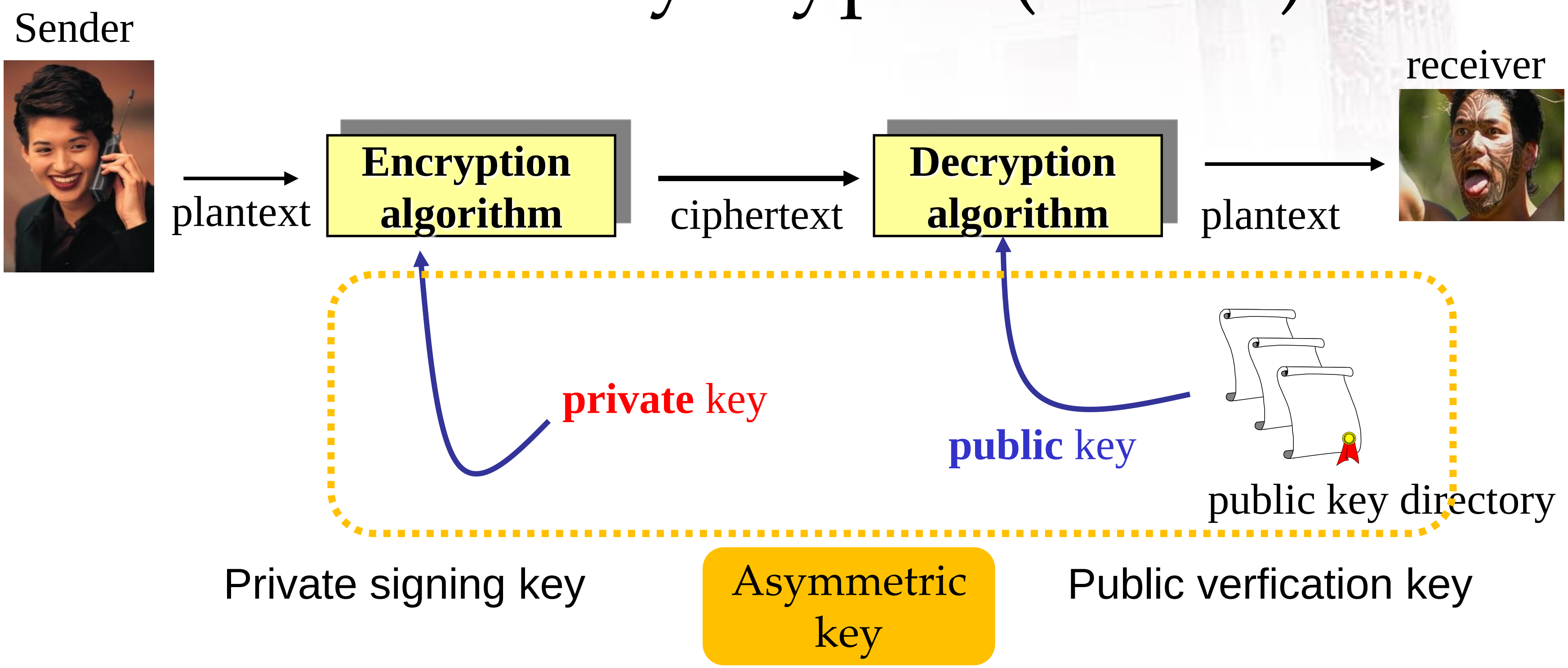


Public-key encryption

Public key: open to the public

Private key: key owner only

Public key crypto. (cont'd)



Digital Sigatures

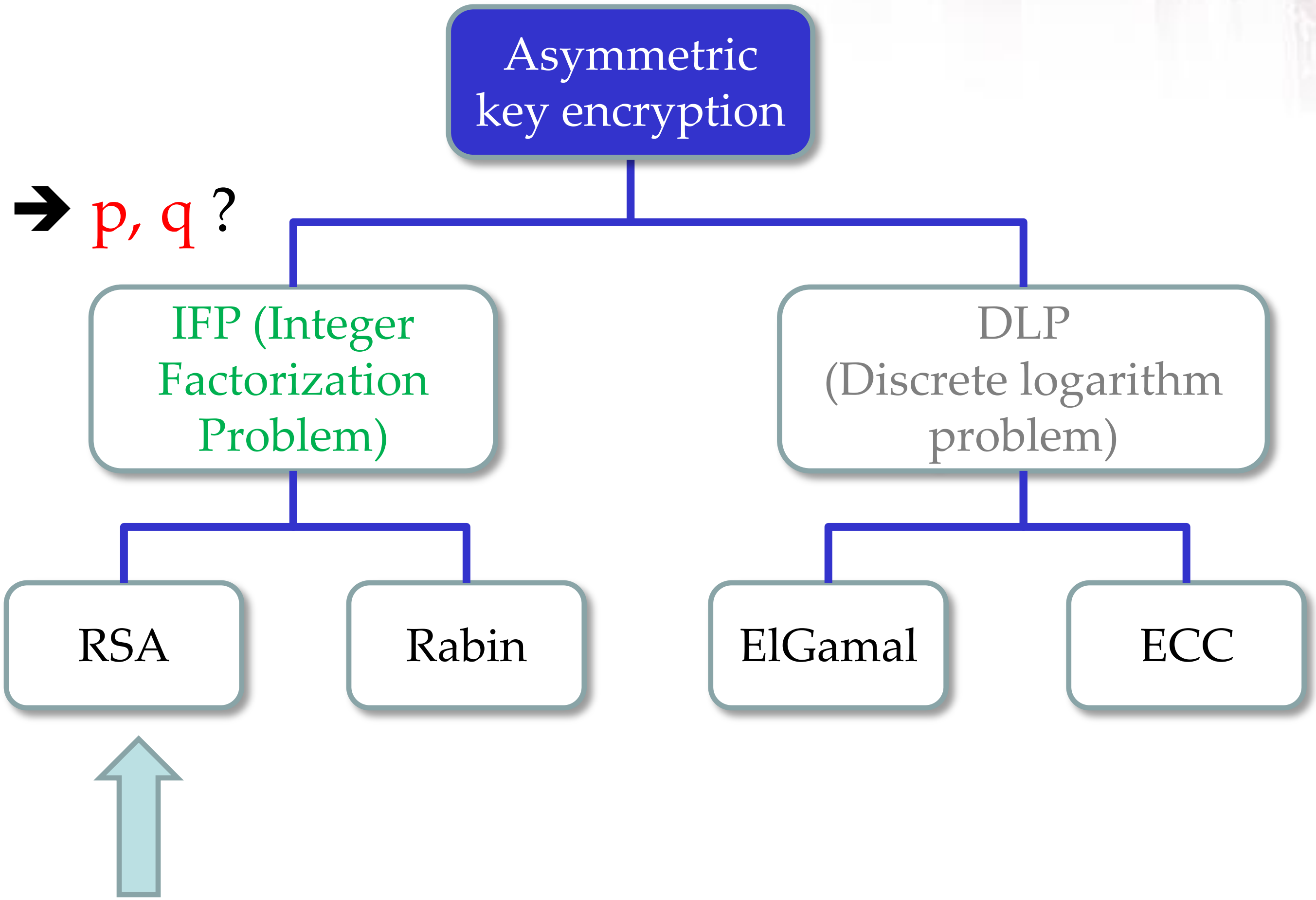
Public key: open to the public

Private key: key owner only

Also, combined with private key encryption algorithms

Asymmetric key ciphers

$n=pq \rightarrow p, q$?



IFP (Integer Factorization Problem)

- Problem: Given a composite number *n*, find its prime factors

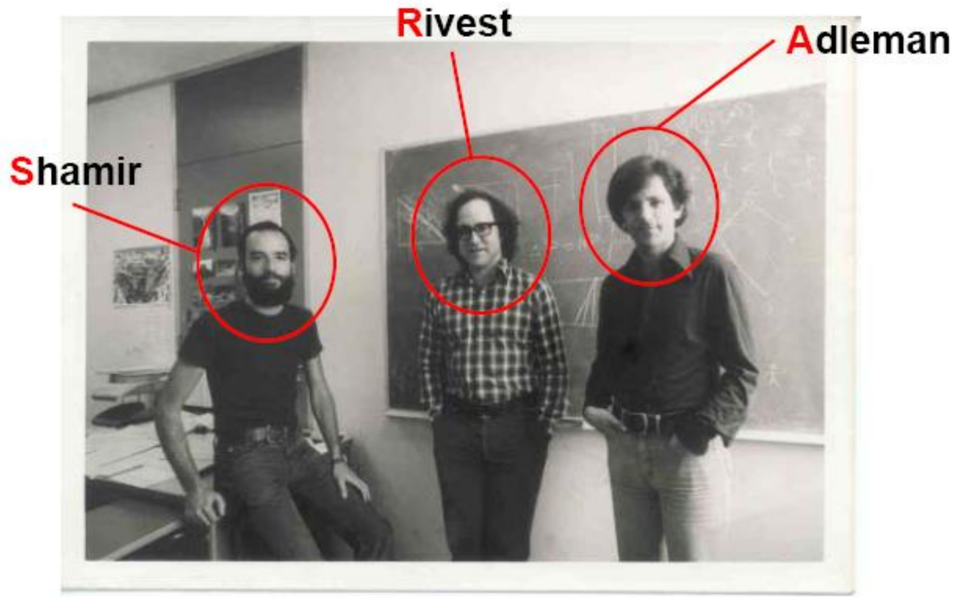
$$\textbf{Primes p, q} \underset{\text{hard}}{\overset{\text{easy}}{\rightleftarrows}} \textbf{n=pq}$$

- Application: Used to construct RSA-type public key cryptosystems

RSA

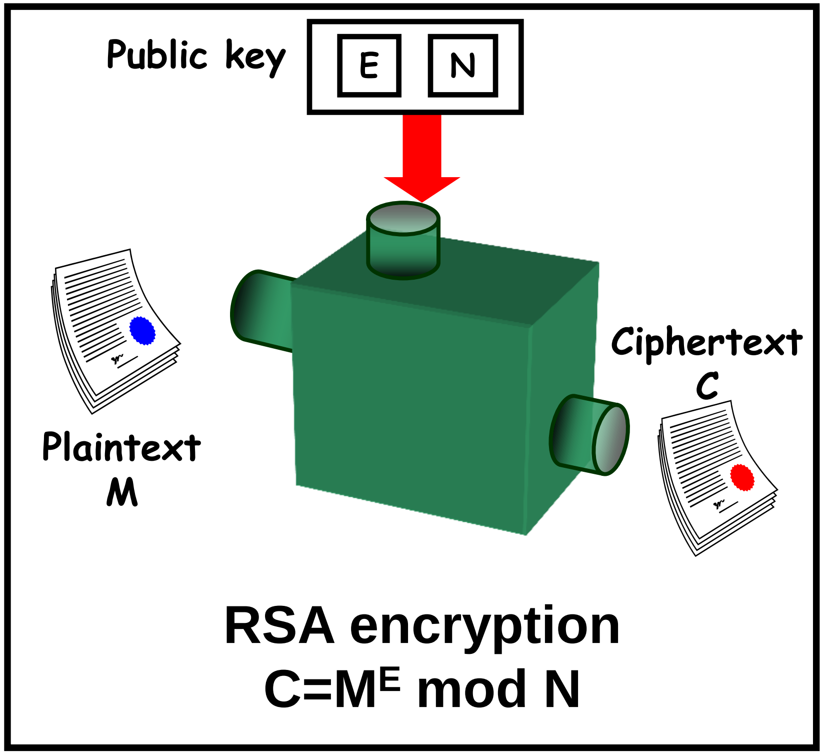
- 1st public key cryptosystem   Cf. DH – key exchange
- Believed to be secure if IFP (Integer Factorization Problem) is hard and worldwide standard for last 30 years.

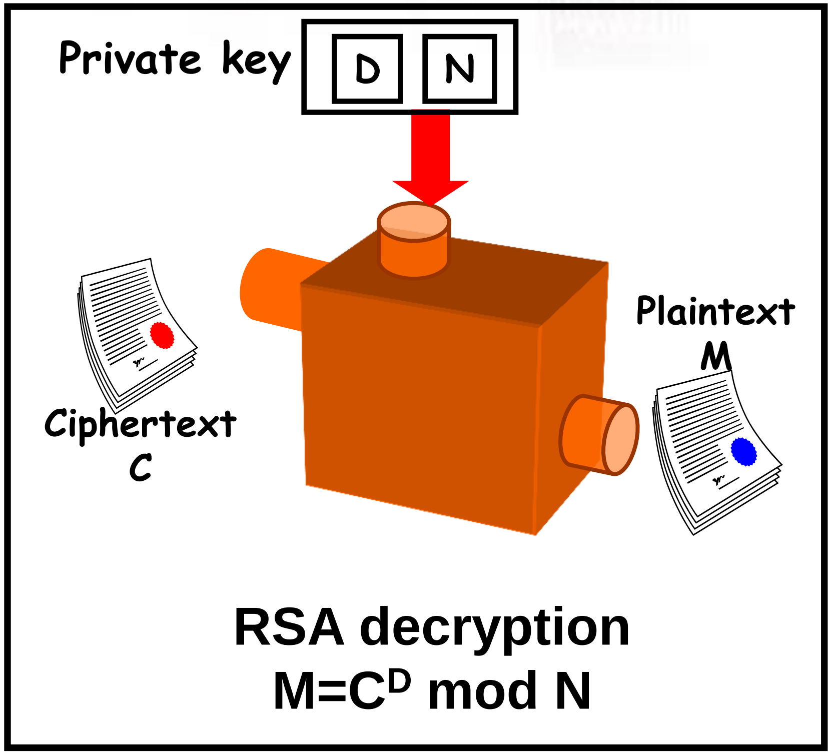
RSA (Ron Rivest, Adi Shamir and Leonard Adleman)



R.L.Rivest, A.Shamir, L.Adleman, “A Method for Obtaining Digital Signatures and Public Key Cryptosystems”, CACM,(Communications of the Association for Computing Machinery) Vol.21, No.2, pp.120-126,Feb,1978

RSA encryption and decryption





How to generate the key pair?

RSA encryption and decryption (cont'd)

Public key

E N

Message M

$C = f(M) = M^{e} \bmod n$

Ciphertext C

$M = f^{-1}(C) = C^{d} \bmod n$

Private key

D N

n=pq (p & q: primes)

ed = 1 mod (p-1)(q-1)

RSA Key generation

1. Select two large (1,024 bits or larger) primes p, q
2. Compute modulus $n = pq$ and
3. Compute $\varphi(n) = (p-1)(q-1)$
4. Pick an integer e relatively prime to $\varphi(n)$, $gcd(e, \varphi(n))=1; 1<e<\varphi(n)$ (or Z^*_n set of residues)
5. Compute d such that $ed = 1 \bmod \varphi(n)$ *using the Euclidean algorithm*
6. Public key (n, e) : public
   Private key (n, d) : keep secret
7. Encryption $C=f(M) = M^e \bmod n$
   Decryption $M=f^{-1}(C) = C^d \bmod n$

φ: Euler's Totient function

gcd (greatest common divisor)

Why RSA Works?

- We need to proof $(m^e)^d \bmod n = m$

$$
\begin{aligned}
&(m^e)^d \bmod n \\
&= m^{k\,\phi(n)+1} \bmod n \\
&= m^{(p-1)(q-1)k+1} \bmod n \\
&= (m^{p-1})^{(q-1)k}\, m \bmod n \\
&= (\mathbf{1 \bmod p})^{(q-1)k}\, m \bmod n \\
&= 1\, m \bmod p \bmod n \\
&= m \bmod p
\end{aligned}
$$

From Fermat's Little Theorem

n is divisible by p

We also can proof that $(m^e)^d \bmod n = m \bmod q$

since n=pq, then $(m^e)^d \bmod n = m \bmod pq$

which is $(m^e)^d \bmod n = m \bmod n$

Fermat's Little Theorem

- From a simple theorem: $\boxed{a^p \equiv a \bmod p}$
  - Eg. $2^5 = 32 = 2 \bmod 5$
- If we divide the equation by a: $\boxed{a^{p-1} \equiv 1 \bmod p}$
  - Eg. $2^{5-1} = 16 = 1 \bmod 5$

RSA Example

1. Choose p=3 and q=11
2. Compute $n=p*q$=3*11=33
3. Compute $\varphi(n) = (p-1)(q-1)$ = 2*10=20
4. Choose e such that $1<e<\varphi(n)$ and e and $\varphi(n)$ are coprime. Let e=7
5. Compute a value for d such that $(d*e) = 1 \bmod \varphi(n)$. One solution is d=3 ➔ [(3*7) = 1 mod 20]
6. Public key is (e, n) = (7, 33)
7. Private key is (d, n) = (3, 33)
8. The encryption of $m = 2$ is $c = 2^7 \bmod 33 = 29$
9. The decryption of $c = 29$ is $m = 29^3 \bmod 33 = 2$

Q: How secure RSA is?

Attacks against RSA

- Weak RSA if
  - Small p and q
  - Small difference of p and q
- Is RSA computational secure?
- Is RSA secure against a known-plantext attack?
- Is RSA secure against a chosen-plaintext attack?

Attacks against RSA

- Is RSA secure against a chosen-ciphertext attack?
- Suppose Eve collects $c=m^e \bmod n$ from Bob, she needs to recover m, for which $m=c^d \bmod n$
- Eve first choose a random number $r < n$, and compute
  - $x = r^e \bmod n \Rightarrow x^d = r^{ed} \bmod n \Rightarrow x^d = r \bmod n$ Chosen-plaintext
  - $y = xc \bmod n$ Chosen-ciphertext
  - $t = r^{-1} \bmod n$
- Eve gets Bob to sign y with his private key (suppose he sign the message, not hash of the message). Bob return
  - $u = y^d \bmod n$
- Now Eve computes
  - $tu \bmod n = r^{-1}y^d \bmod n = r^{-1}x^dc^d \bmod n = c^d \bmod n = m$

Common Modulus Attack on RSA

- Suppose RSA gives everyone the same n, but different exponents.
- Let m be the plaintext, e_1 and e_2 are two encryption key. Usually gcd(e_1, e_2) =1.
  - $c_1 = m^{e1} \bmod n$
  - $c_2 = m^{e2} \bmod n$
- Attacker knows c_1, c_2, e_1, e_2, and n, he can recover m by the extended Euclidean algorithm $se_1+te_2=1$.
  - $c_1^s \cdot c_2^t = (m^{e1} \bmod n)^s \cdot (m^{e2} \bmod n)^t = m^{e1 \cdot s + e2 \cdot t} \bmod n$
    $= m^1 \bmod \mathrm{n}$

	Symmetric	Asymmetric
Key relation	Enc. Key = Dec. key	Enc. Key ≠ Dec. key
Encryption Key	**Secret**	**Public, {Private}**
Decryption Key	**Secret**	**Private, {Public}**
Algorithm	**Classified/Open**	**Open**
Example	**DES (56 bits), AES**	**RSA (1024 bits)**
Key Distribution	**Required**	**Not required**
Number of key	**Many (Mbits/second)**	**Small (eg., kbits/second)**
Performance	**Fast**	**slow**

Remember this

- Key distribution by Public key cipher
  - Secret key is distributed by asymmetric (public) key cipher (e.g., DH, RSA)
- Data encryption by symmetric (private) key cipher
  - The shared secret key (note: master key -> session key) is used to encrypt/decrypt the message
- Most security protocols use this idea