Hash, DH and RSA

Short Version

Chun-Jen Chung

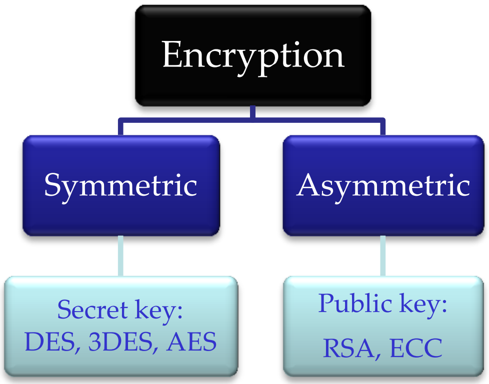
Arizona State University

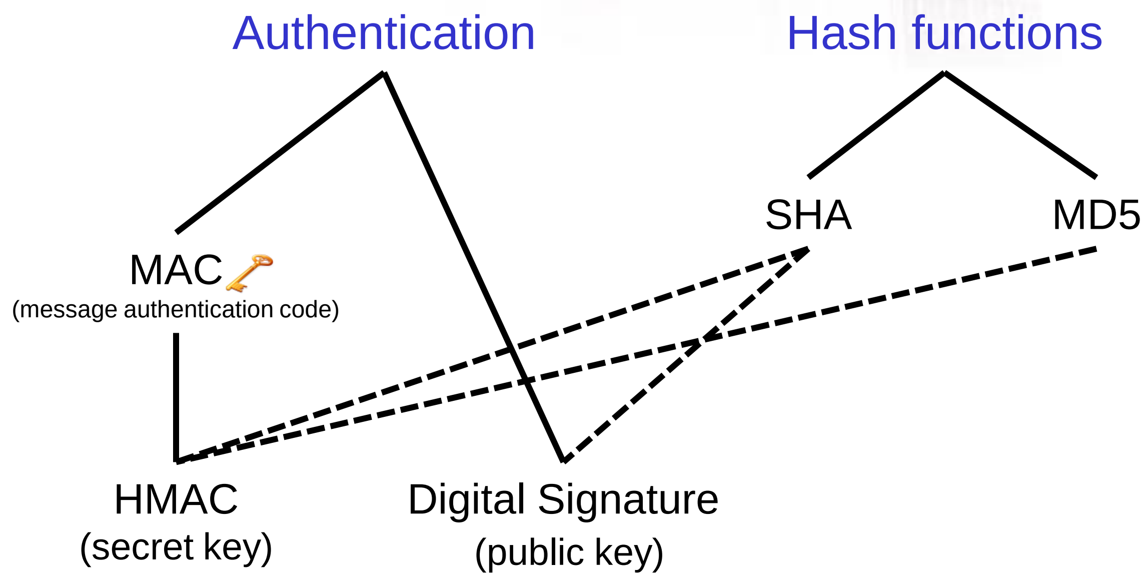
Outline

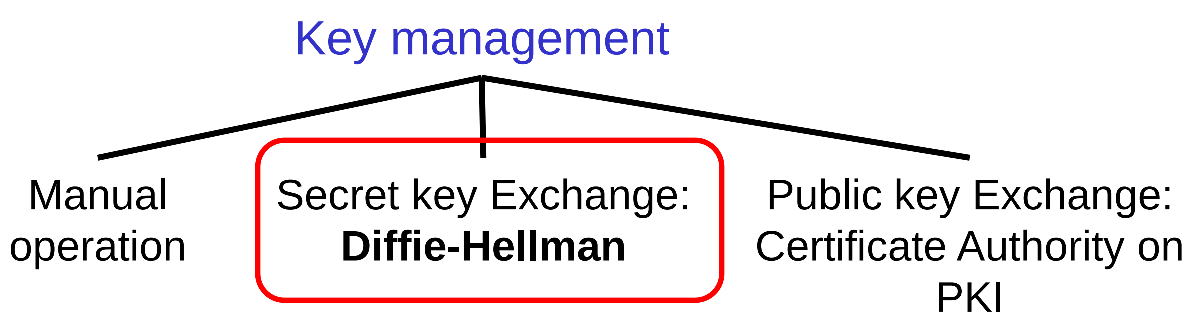
- Background
- Hash Functions
- Public key cryptography (PKC)
  - DH
  - RSA
- Summary

Background

Crypto algorithms review

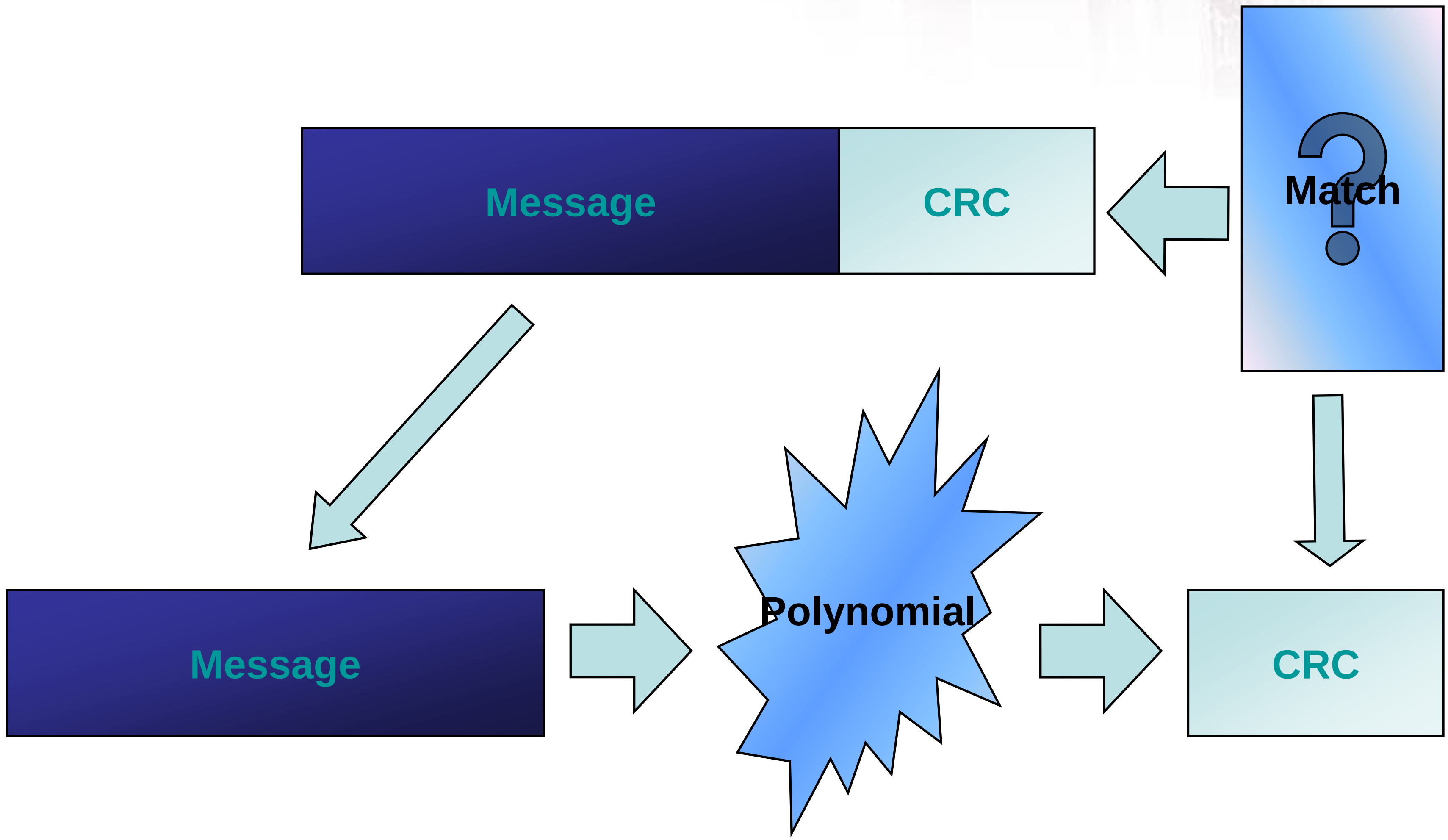




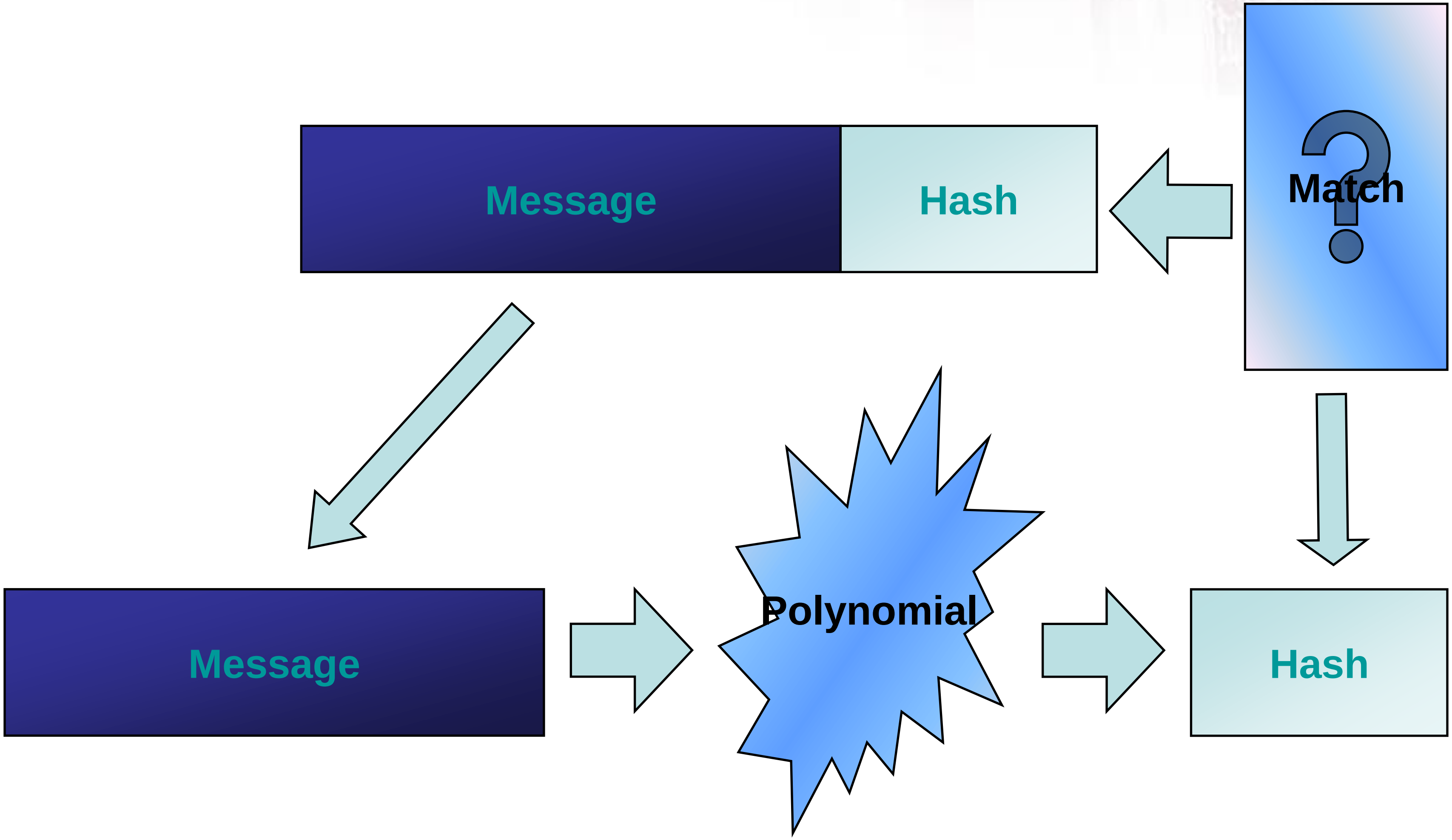


Introduction to Hash Functions

Error checking in frames ...



Hashing is very similar ...



Hash Algorithms

Message of arbitrary length → hash H → A fixed-length short message

- Also known as
  - 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 bit output
  - SHA-1 (secure hash algorithm) : 160bit output
  - SHA-2: 256/224, 512/384

Hash Algorithms (contd)

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)

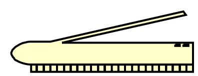


Image from scanner

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

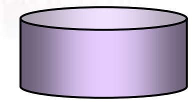
All files of a floppy disk

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)

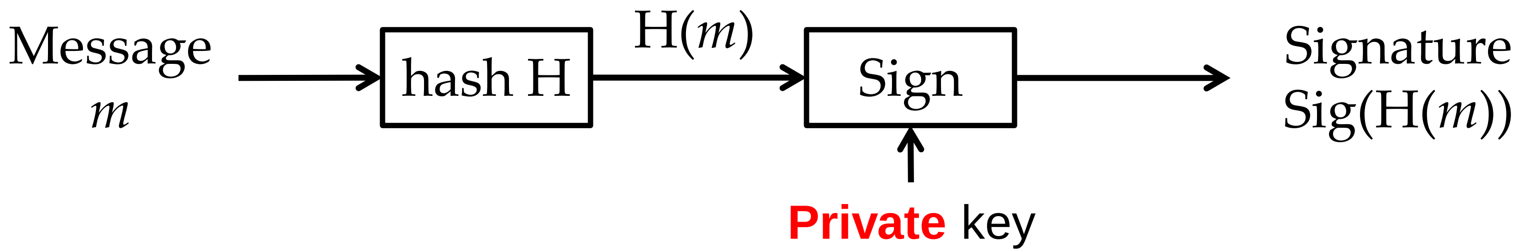
32 2B 23 70 7A
2B 4F 43 B0 4C
54 3B 49 28 67
A2 23 8F 7D 36

Hash value 20 bytes

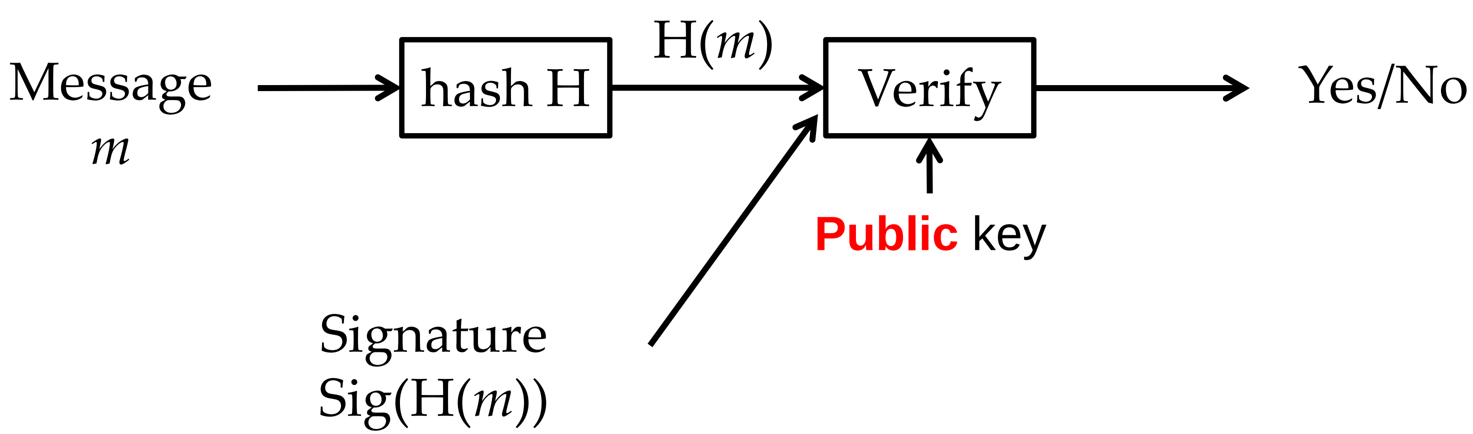
Applications of Hash Functions

- Primary application
  - Generate/verify digital signature

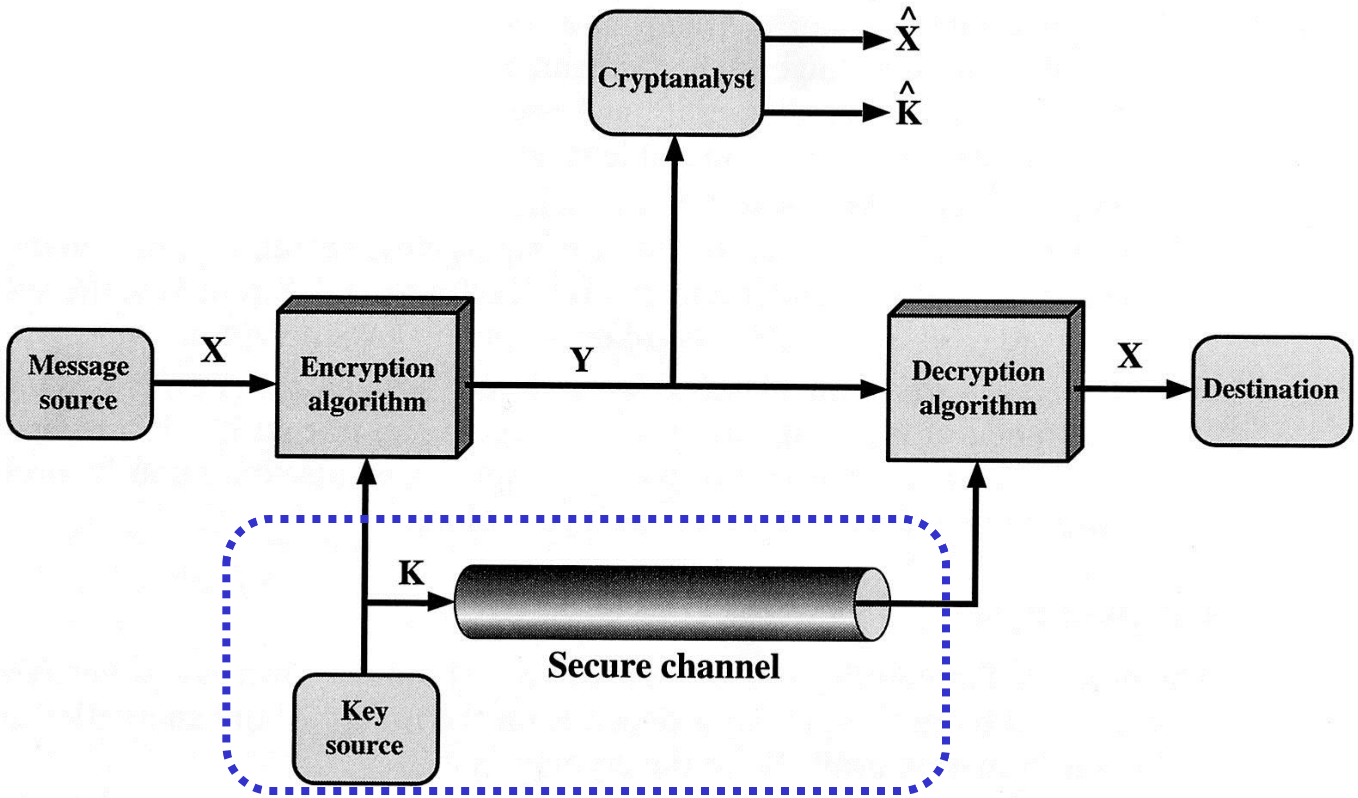
Send



Receive



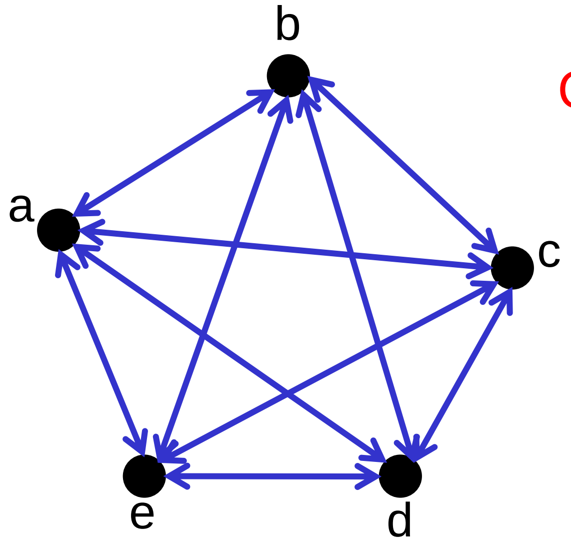
Problem of Symmetric key Crypto



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)

DLP (Discrete logarithm problem)

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)	$g^x \bmod p = Y$ e.g., $10^x \bmod 19 = 9$: $x=10$

Q: how difficult?

An example

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

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 $y = g^x \bmod p$

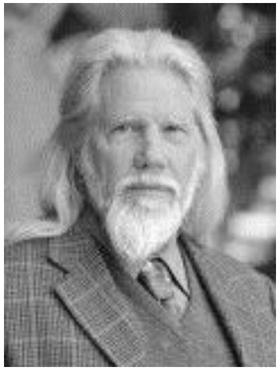
Given g, x, p	$\xrightarrow{\textbf{easy}}$	$y=g^x \bmod p$
x	$\xleftarrow{\textbf{hard}}$	Given g, y, p

- 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 approach

HTTP	FTP	SMTP
SSL/TLS		
TCP		
IP		

At the transport 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

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

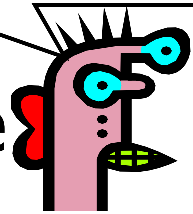
g is a **generator** of a group G if every element in G can be expressed as the product of finitely many powers of g.

DH Key Exchange Protocol

F={1,2,3,…,p-1}

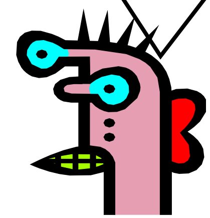
(1) Pick secret, random a from F

Alice



(2) $A = g^a \bmod p$ →

(3) $B = g^b \bmod p$ ←



Bob

(1) Pick secret, random b from F

(4) Compute
$k=B^a \bmod p = (g^b)^a \bmod p$
$= g^{ab} \bmod p$

(4) Compute
$k=A^b \bmod p = (g^a)^b \bmod p$
$= g^{ab} \bmod p$

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

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

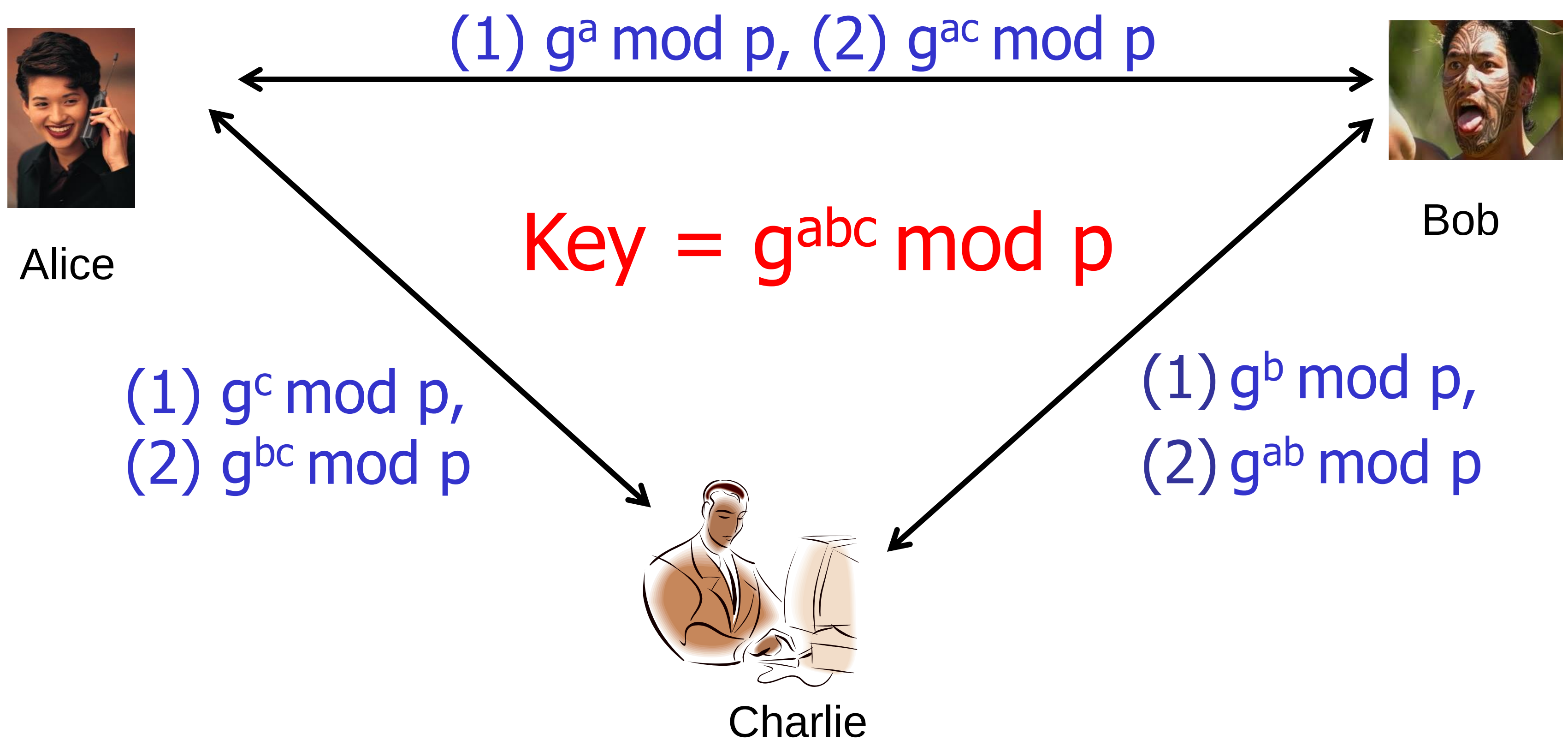
$$y = g^b \bmod p$$

- 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 for three parties

- How can **three** 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

Alice $\xrightarrow{Y_a}$; $\xleftarrow{Y_c}$ Alice

$\xrightarrow{Y_c}$ Bob ; $\xleftarrow{Y_b}$ Bob

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

Alice computes the session key

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

E (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 (ISO-9796)

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

$A \rightarrow B$: A, g^a

$B \rightarrow A$: $B, g^b, SIG_B(g^a, g^b, A)$

$A \rightarrow B$: $SIG_A(g^b, g^a, B)$

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

A $\xleftarrow{B,\ g^y,\ SIG_B(g^x, g^y, E)}$ E $\xleftarrow{B,\ g^y,\ SIG_B(g^x, g^y, E)}$ B

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

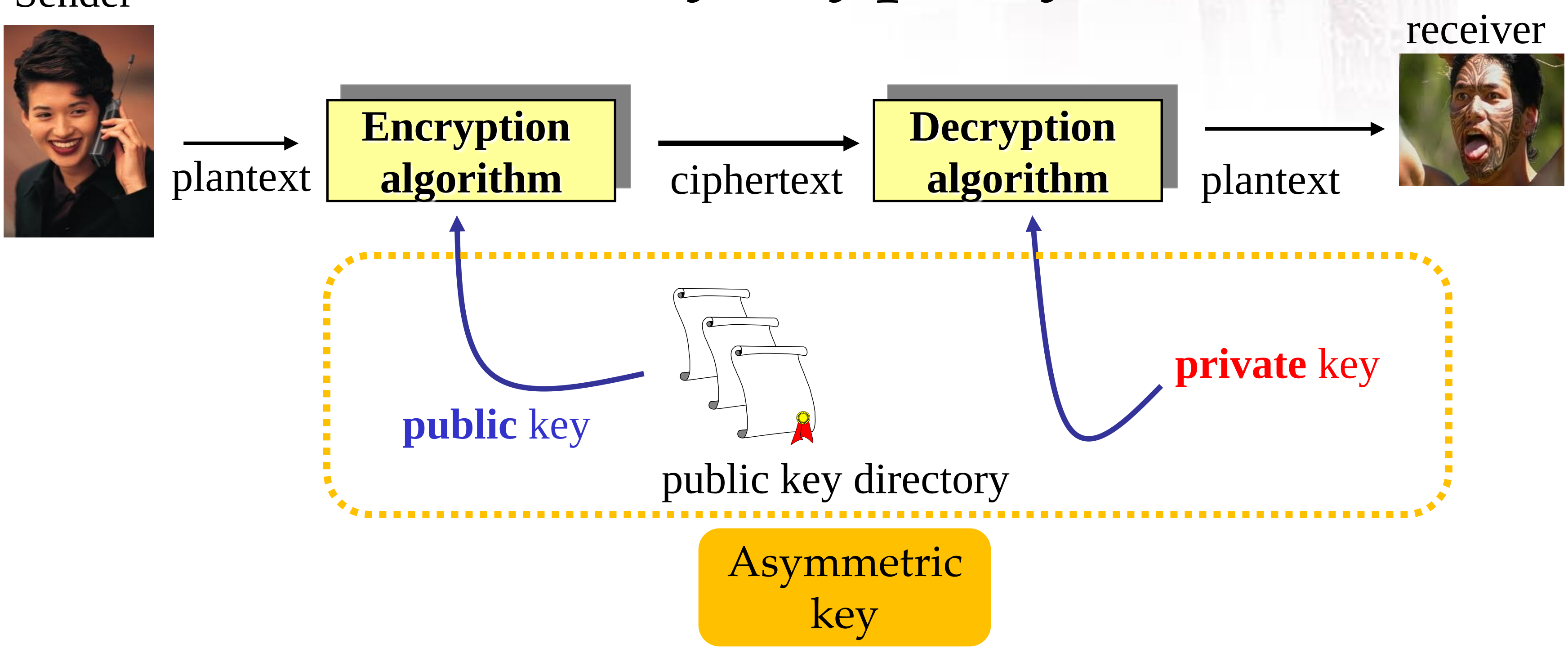
Note that E cannot produce $SIG_B(g^x,g^y,A)$

- The ISO protocol thus avoids the misbinding attack

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

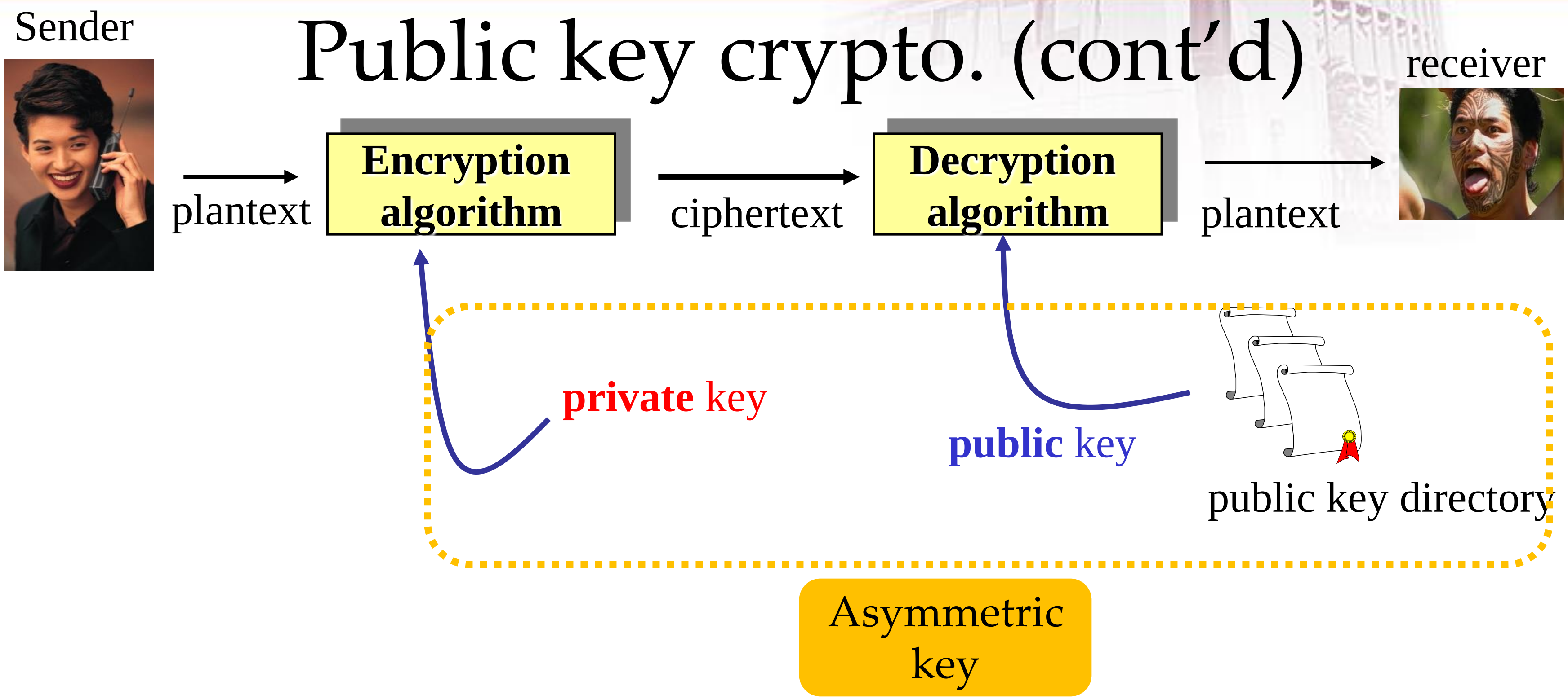
RSA

Public key cryptosystem



Public key: open to the public
Private key: key owner only

Public key crypto. (cont'd)

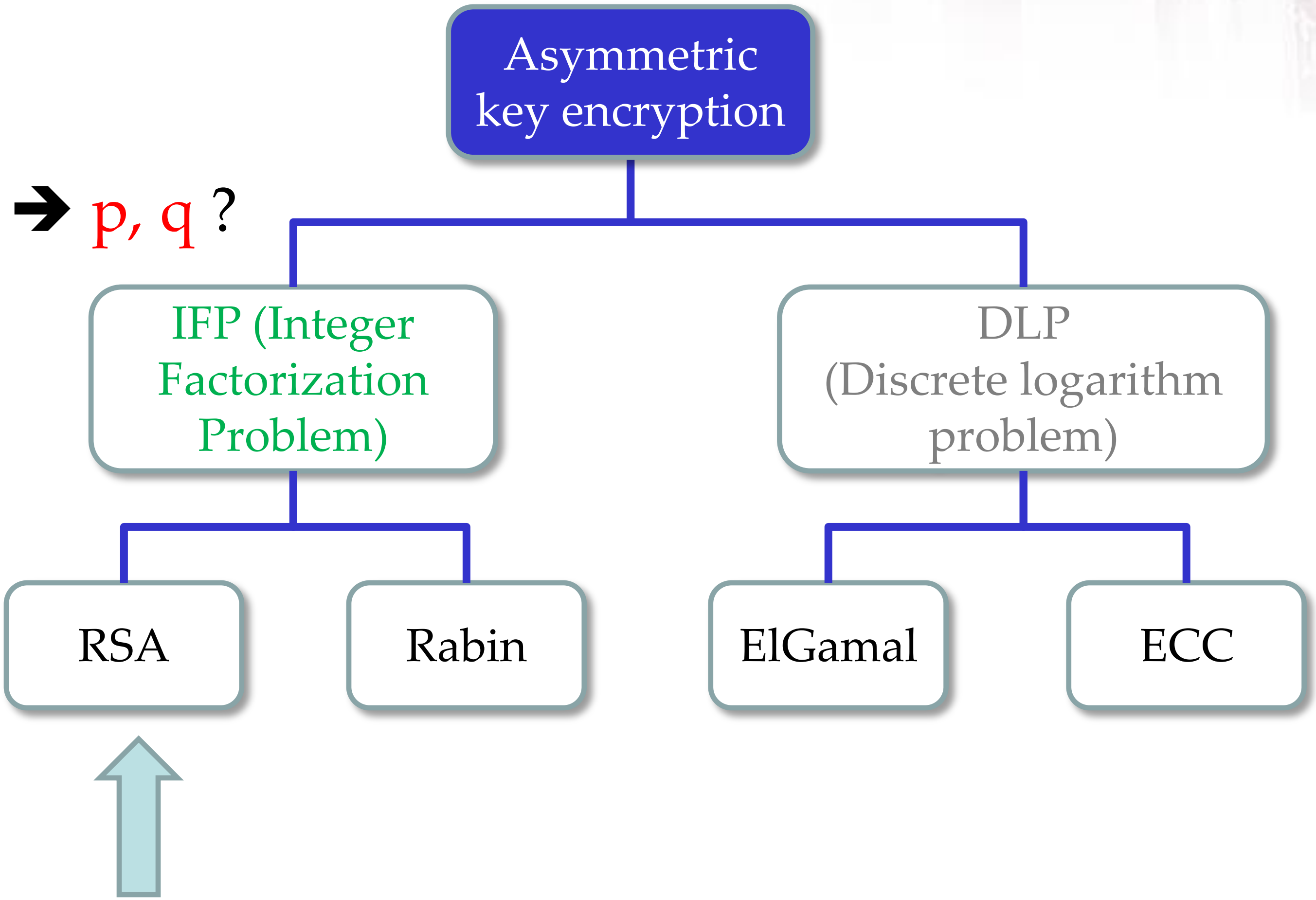


Public key: open to the public
Private key: key owner only

Also, combined with private key encryption algorithms

Asymmetric key ciphers

n=pq ➔ p, q ?



IFP (Integer Factorization Problem)

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

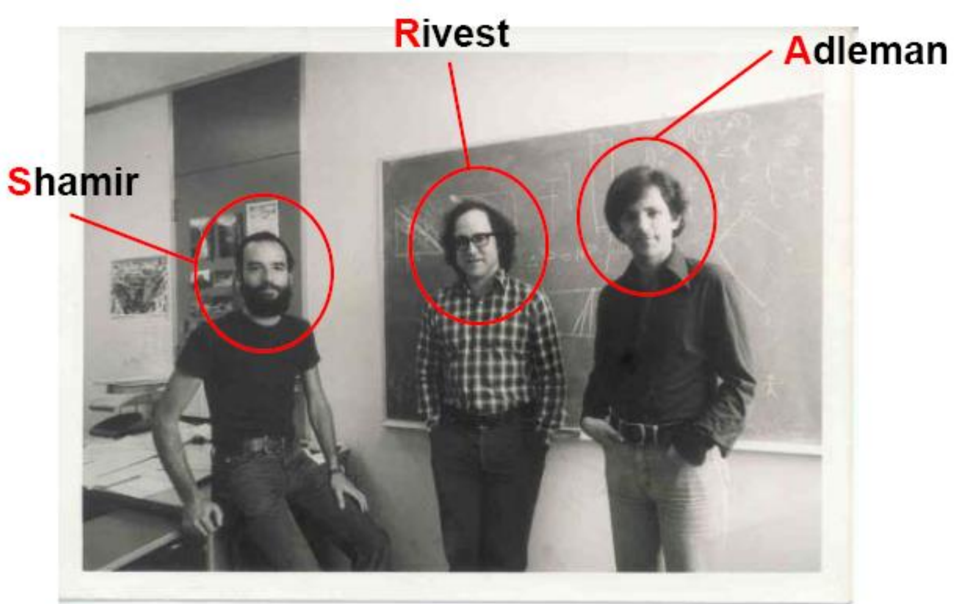
$$\textbf{Primes } \mathbf{p, q} \; \underset{\textbf{hard}}{\overset{\textbf{easy}}{\rightleftarrows}} \; \mathbf{n=pq}$$

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

RSA

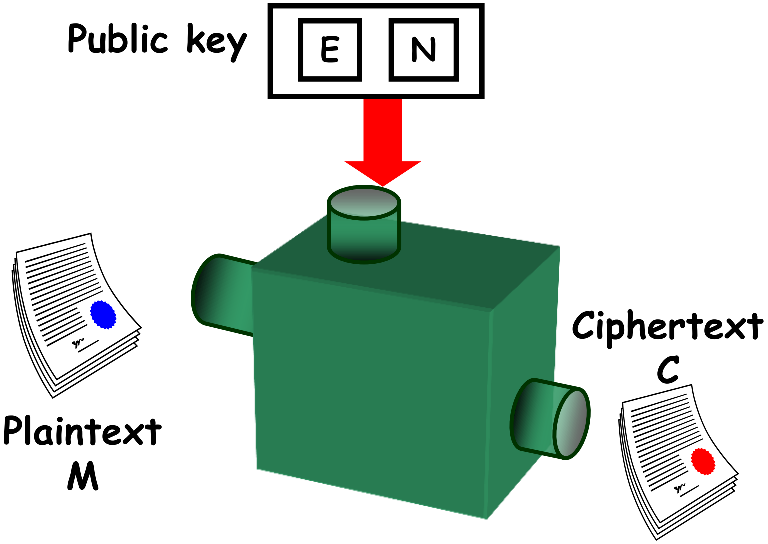
- 1st public key cryptosystem Cf. DH – key exchange
- Believed to be secure if IFP (Integer Factorization Problem) is hard and worldwide standard for last 30 years.

RSA (Ron Rivest, Adi Shamir and Leonard Adleman)



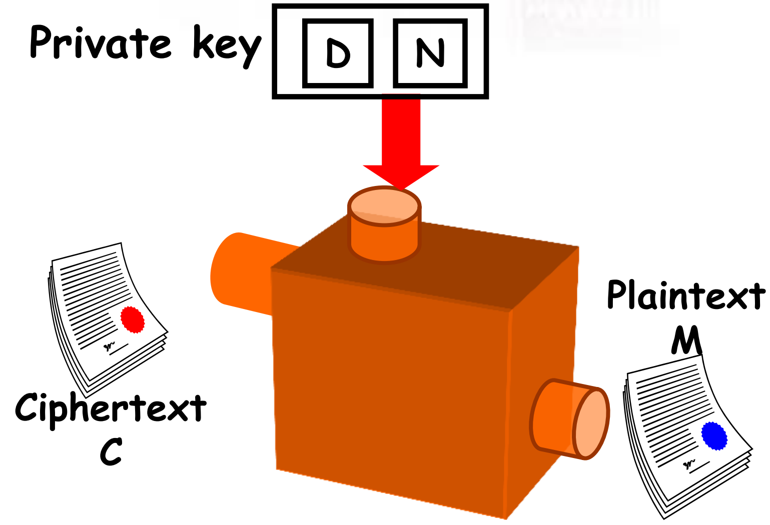
R.L.Rivest, A.Shamir, L.Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", CACM,(Communications of the Association for Computing Machinery) Vol.21, No.2, pp.120-126,Feb,1978

RSA encryption and decryption



RSA encryption

$C=M^E \bmod N$



RSA decryption

$M=C^D \bmod N$

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 $\varphi(n) = (p-1)(q-1)$

   φ: Euler's Totient function

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

- Public key (n, e) : public
- Private key (n, d) : keep secret

gcd (greatest common divisor)

To understand RSA, we need to know

- Prime number
- GCD (great common divisor)
- Relatively prime
- Euclidean Algorithm
- Congruence
- Multiplicative reverse

However, we do not need you to get into them in this class. Only for your own interest!

Q: How secure RSA is?

RSA challenge

- One of the first description of RSA was in the paper.
  - Martin Gardner: Mathematical games, Scientific American, 1977
  - offered a $100 prize for breaking the message
  - and in this paper RSA inventors presented the following challenge.
  - Decrypt the ciphertext:

9686 9613 7546 2206 1477 1409 2225 4355 8829 0575 9991 1245 7431 9874 6951 2093 0816 2982 2514 5708 3569 3147 6622 8839 8962 8013 3919 9055 1829 9451 5781 5154

Encrypted using the RSA cryptosystem with

n *(composite number)*: 114 381 625 757 888 867 669 235 779 976 146 612 010 218 296 721 242 362 562 561 842 935 706 935 245 733 897 830 597 123 513 958 705 058 989 075 147 599 290 026 879 543 541.

and with e = 9007 $C=M^e \bmod n$

M. Gardner, “Mathematical Games: A New Kind of Cipher that Would Take Millions of Years to Break,” Scientific American 237 (1977) 120–124

RSA challenge (cont'd)

- It was solved in 1993–1994 by a large joint computer project
  - More than 600 volunteers contributed CPU time from about 1,600 machines (two of which were fax machines) over six months.
  - The coordination was done via the Internet and was one of the first such projects.
  - By first factorizing n into one 64-bit prime and one 65-bit prime, and then computing the plaintext

p = 32,769,132,993,266,709,549,961,988,190,834,461,413,177,642,
967,992,942,539,798,288,533

q = 3,490,529,510,847,650,949,147,849,619,903,898,133,417,764,
638,493,387,843,990,820,577

- Plaintext: THE MAGIC WORDS ARE SQUEMISH OSSIFRAGE

Note that p, q use 1,024 bits or larger

	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