ASYMETRIC ENCRYPTION

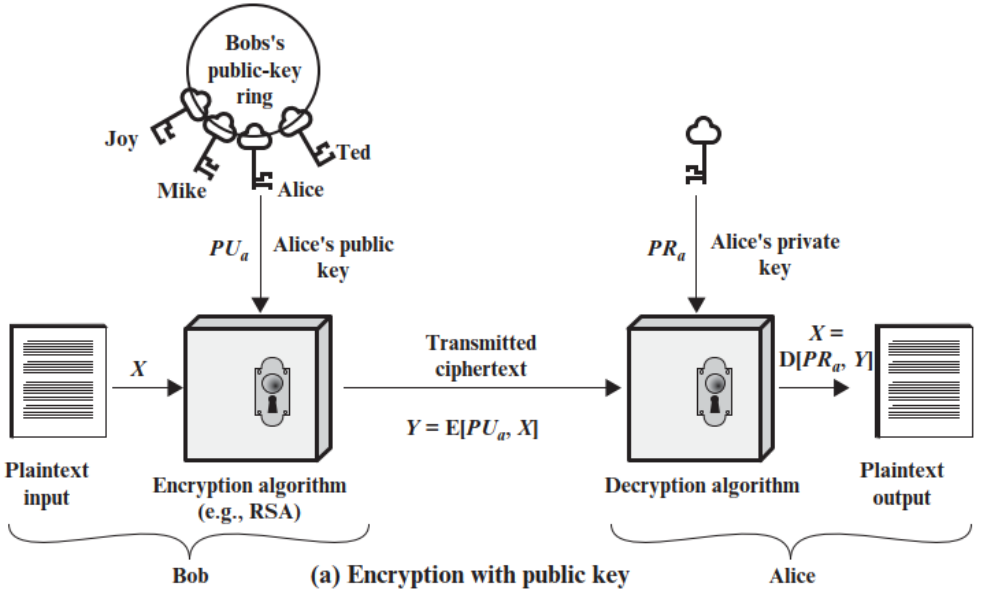
LECTURE CONTENT

- Public key cryptography
- Key issues and differences between secret/public schemes
- RSA
- Other Public Key Cryptosystems

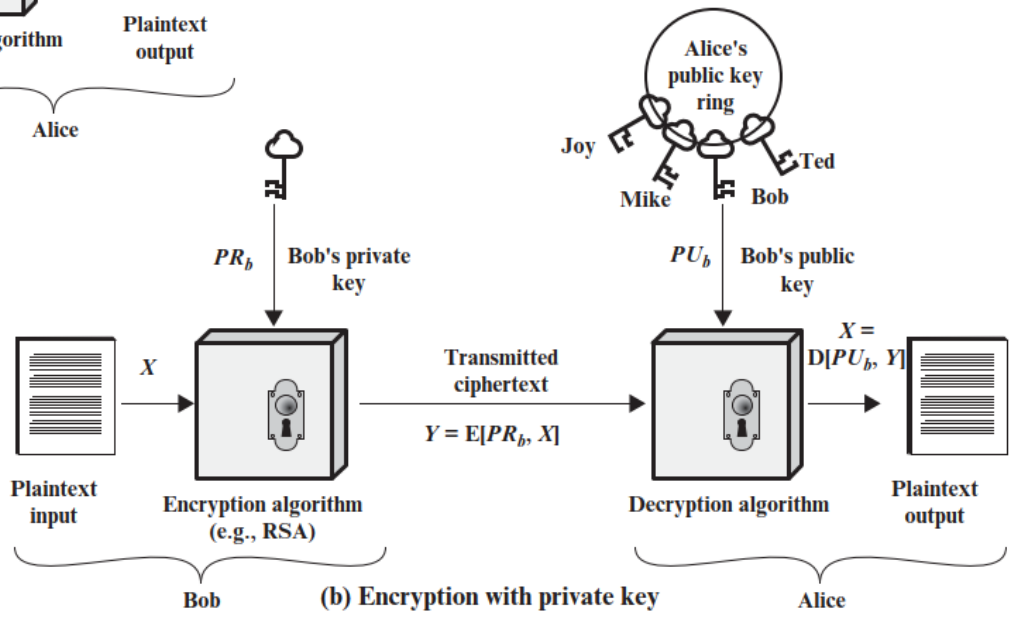
Asymmetric Ciphers

- In contrast to symmetric ciphers, such as DES and AES, which uses the same key, asymmetric ciphers use two separate keys:
  - Public key (PU) and private key (PR) forms a pair of keys
  - Public key is open to the public
  - Private key is kept secret
- Asymmetric ciphers are not to replace symmetric ciphers, or more secure than symmetric ciphers.
- Nowadays, both collaborate to secure the network communication.

Public-Key Cryptography



- Either the private key or the public key can be used to encrypt the message.
- The other is used to decrypt the message.



PR: Private key

PU: Public key

Public-Key Cryptography

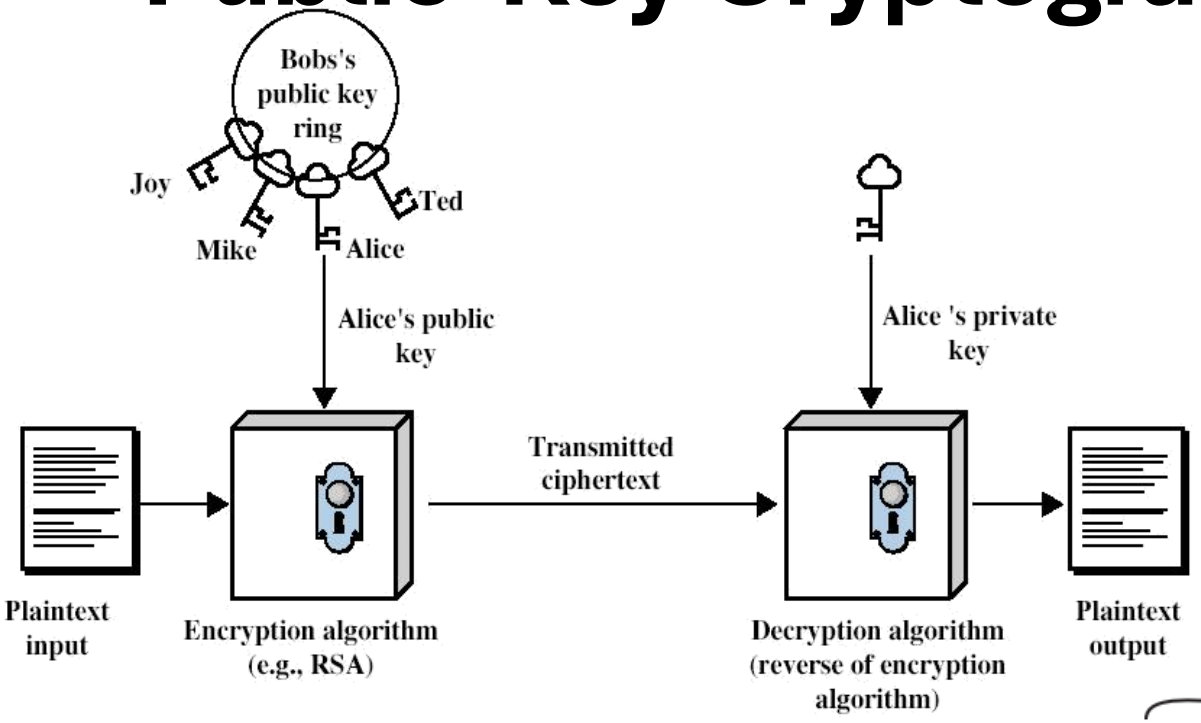
- 1976: Whitfield Diffie & Martin Hellman at Stanford University. Developed to address two key issues:
  - **Key distribution** – how to have secure communications in general without having to trust a key distribution center with your key
  - **Digital signatures** – how to verify a message from the claimed sender

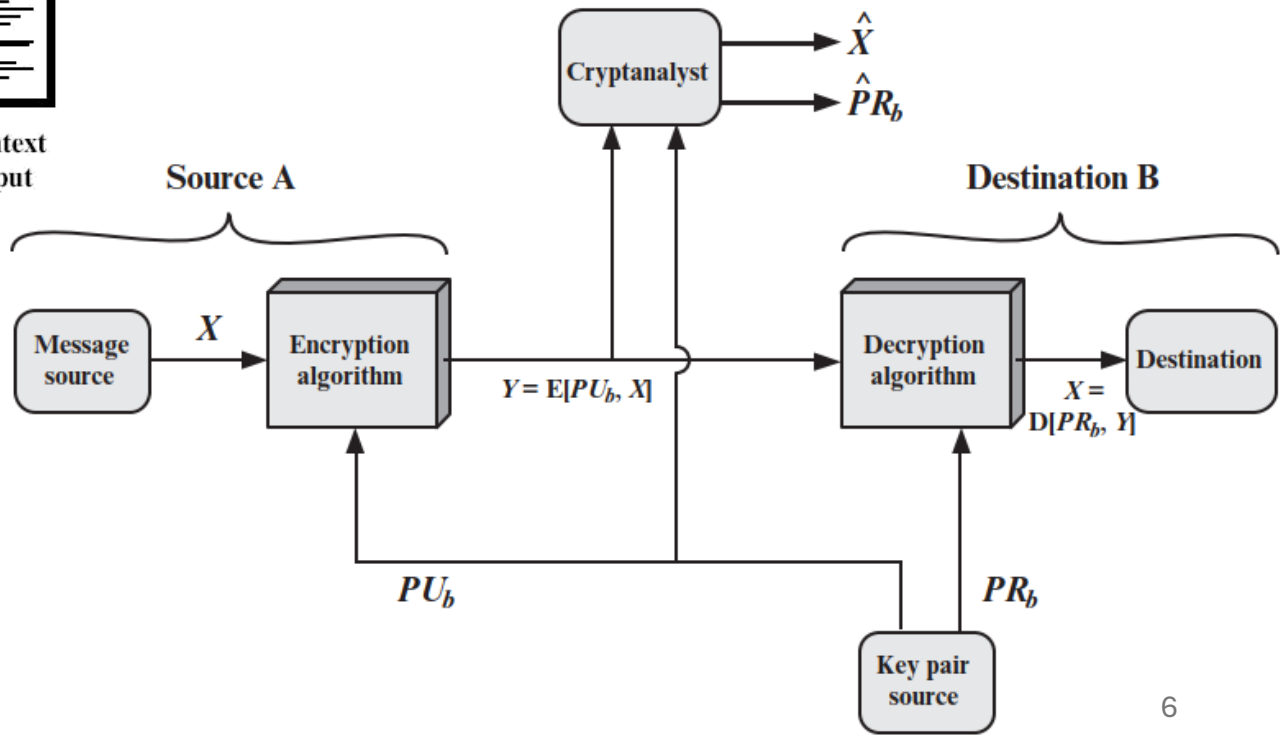
- Uses **two** keys:
  - **Public key** (*KU*): to encrypt messages and verify signatures.
  - **Private key** (*KR*): to decrypt messages, and sign (create) signatures.

- It uses clever application of number theoretic concepts to function

- It complements **rather than** replaces secret key crypto
  - Public key crypto is NOT more secure than secret key crypto
  - Key distribution does NOT disappear

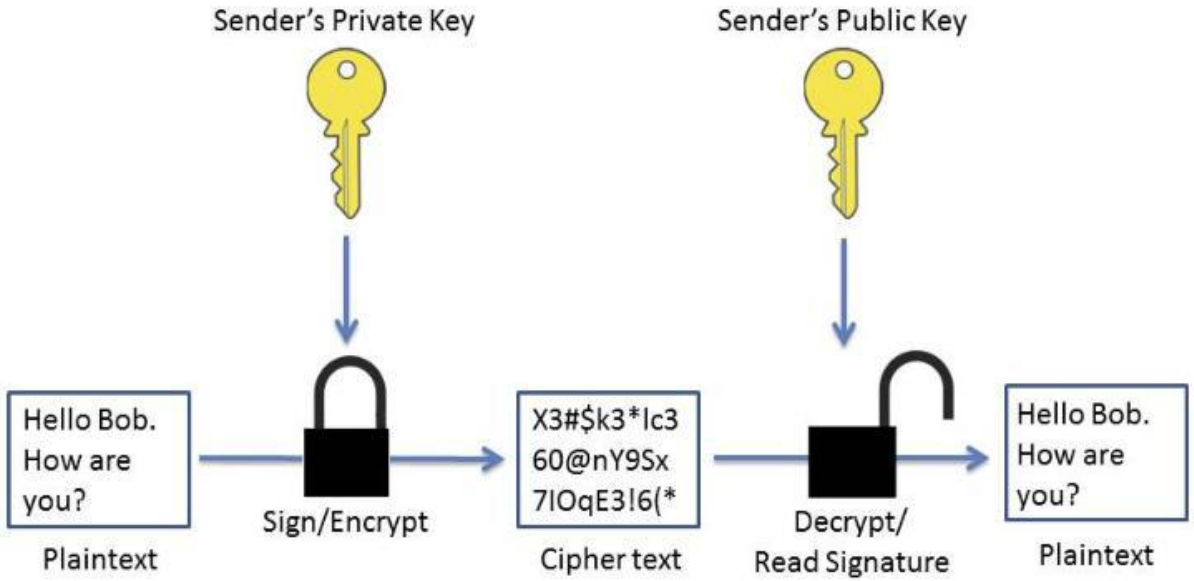
Public-Key Cryptography (Confidentiality)

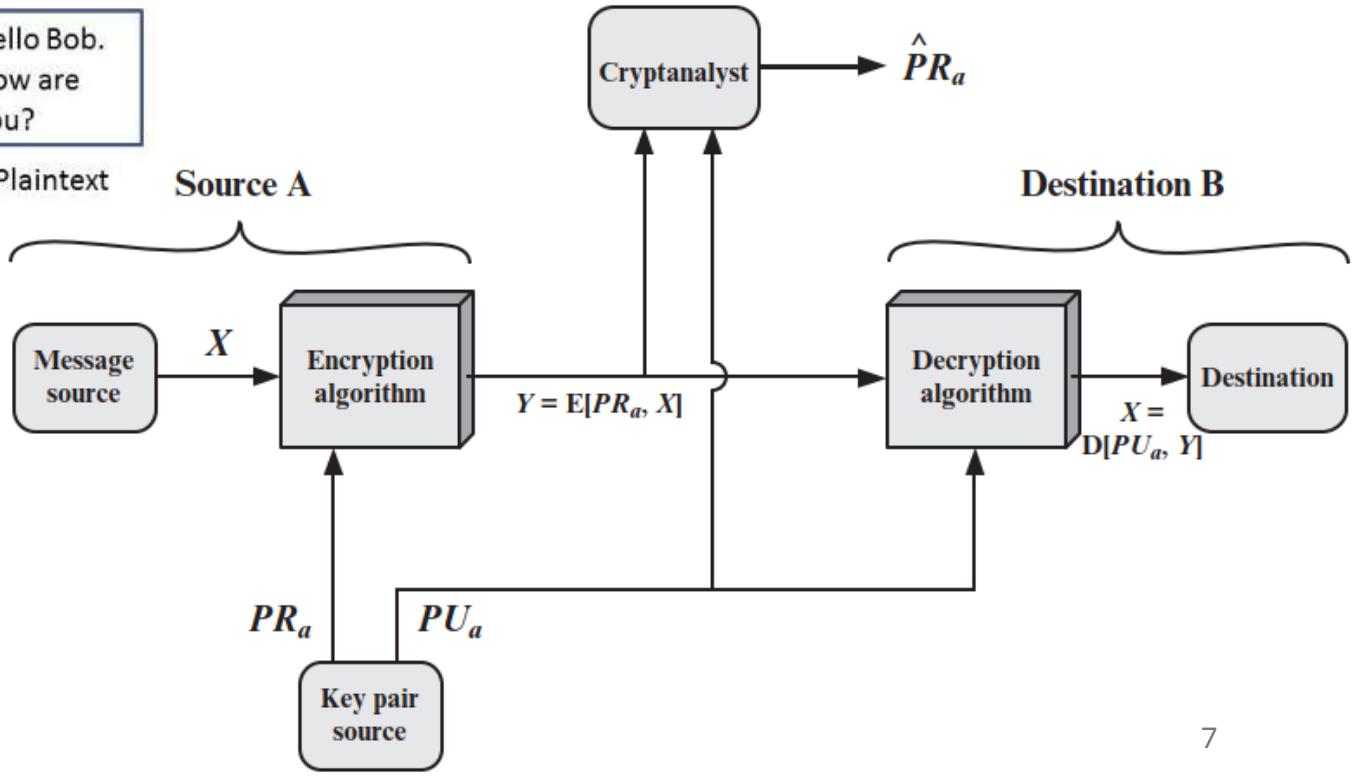




PR: Private key
PU: Public key

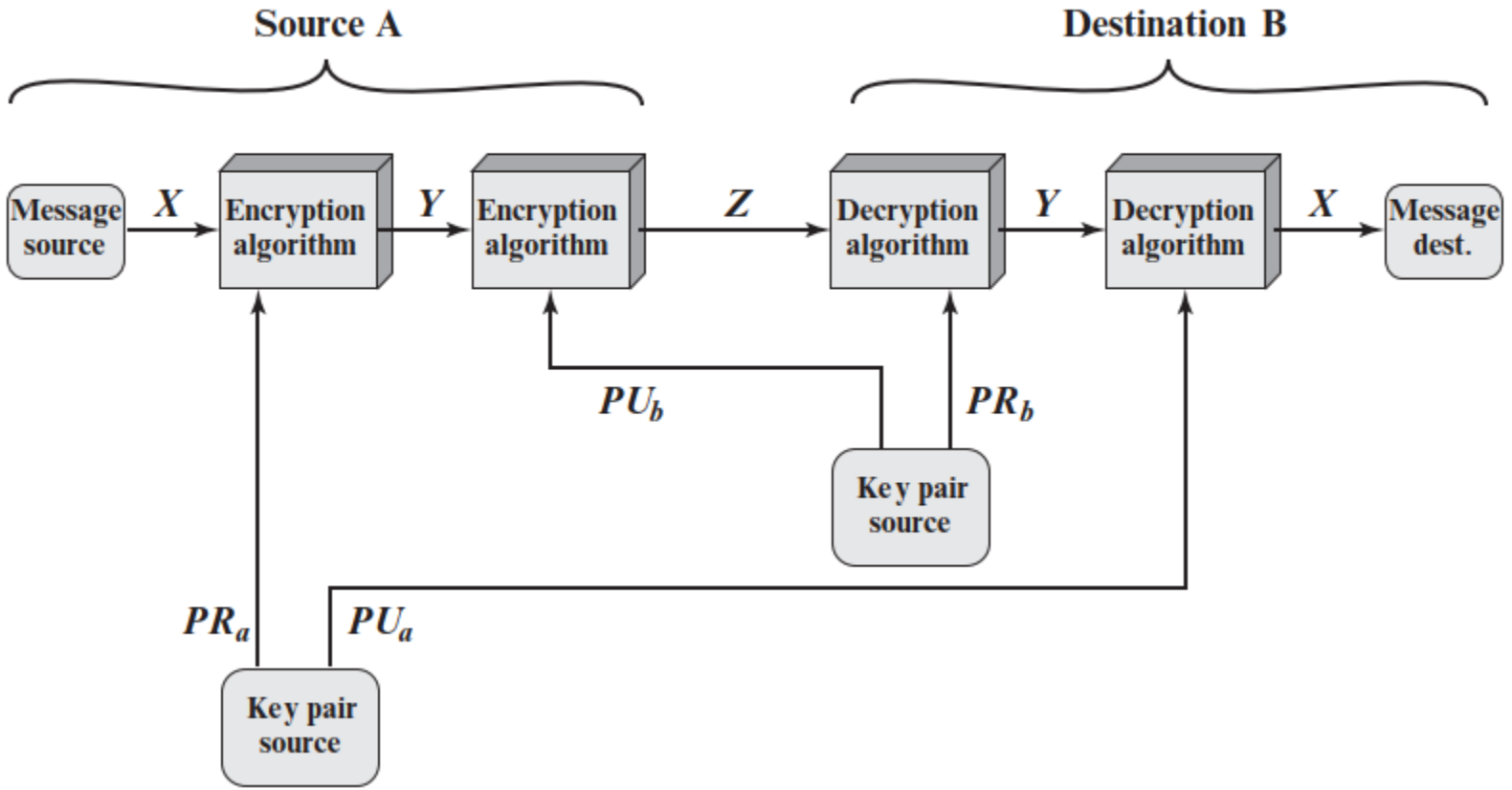
Public-Key Cryptography (Authentication)





PR: Private key
PU: Public key

Public-Key Cryptography (Confidentiality and Authentication)



PR: Private key
PU: Public key

Public-Key Characteristics

- Public-Key algorithms rely on two keys with these characteristics:
  - Computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - Computationally infeasible to find decryption key knowing only algorithm & encryption key
    - Unique factorization
      - For any n: Find its factorization into two primes p and q
    - Discrete logarithms
      - For any b, n, y: Find x such that $b^x \bmod n = y$
  - Either of the two related keys can be used for encryption, with the other used for decryption (in some schemes)

Public-Key Applications

- They can be classified into 3 categories:
  - **Encryption/decryption** (provide secrecy)
  - **Digital signatures** (provide authentication)
  - **Key exchange** (provide key distribution of session keys)

- Some algorithms are suitable for all uses, others are specific to one

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

PGP, GPG, Open PGP

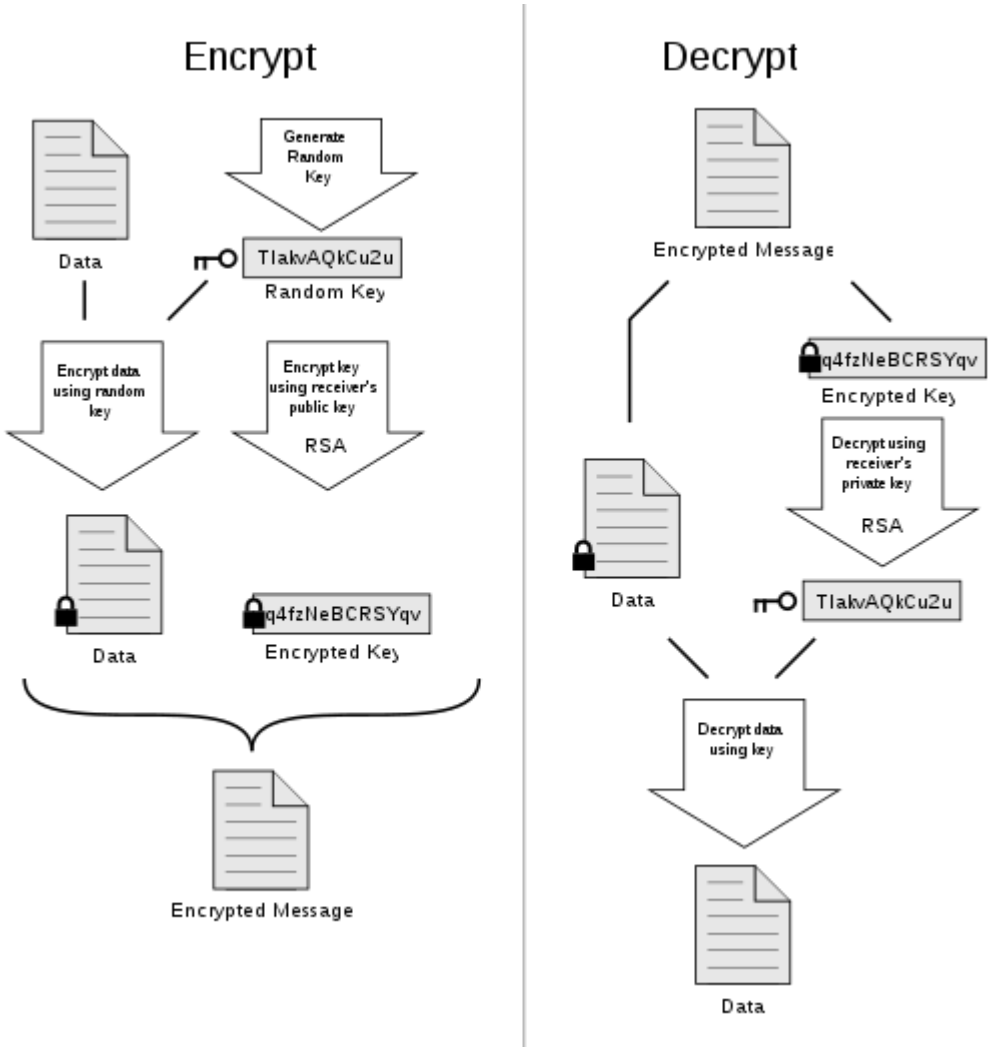
Informally:

- **Pretty Good Privacy (PGP)** is proprietary software written by Phil Zimmerman and released in 199. It uses RSA and IDEA.

- **Gnu Privacy Guard (GPG)** is similar software released in 1999 under the GPL open source license. It uses (RSA, ElGammal, DSA) and NIST AES.

- **OpenPGP** is an IETF standard with which both pieces of software are compliant.

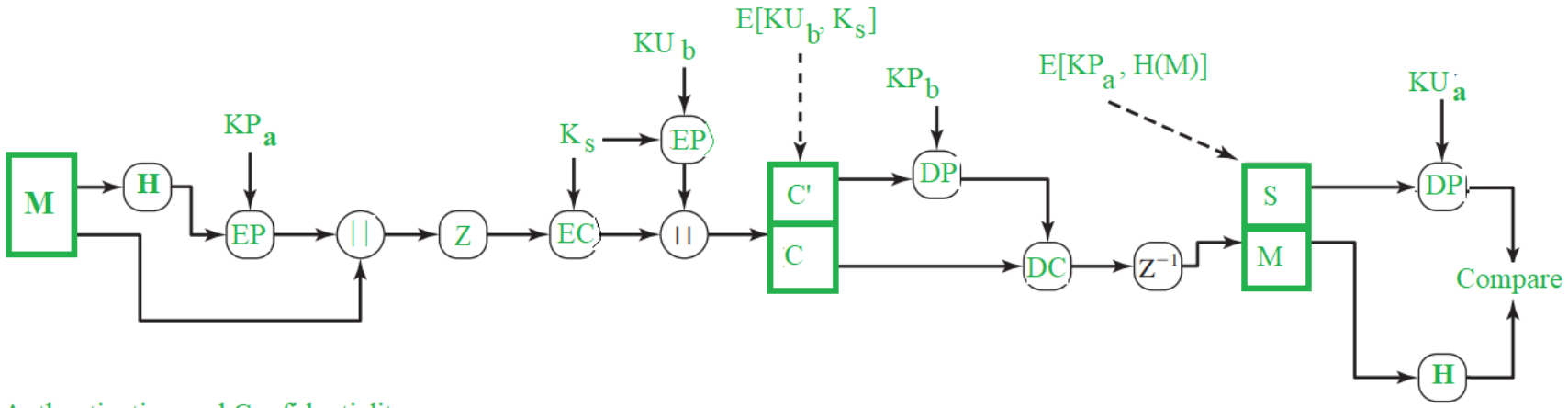
PGP ENCRYPT SCHEME



SOURCE: https://wikiimg.tojsiabtv.com/wikipedia/commons/thumb/4/4d/PGP_diagram.svg/500px-PGP_diagram.svg.png

PGP, GPG, Open PGP

PGP AUTHENTICATION AND CONFIDENTIALITY



Authentication and Confidentiality

Symbol	Meaning
M	Message
H	Hash Function
K_s	A random Session Key created for Symmetric Encryption purpose
DP	Public-Key Decryption Algorithm
EP	Public-Key Encryption Algorithm
DC	Asymmetric Encryption Algorithm
EC	Symmetric Encryption Algorithm
KP_b	A private key of user B used in Public-key encryption process
KP_a	A private key of user A used in Public-key encryption process
KU_a	A public key of user A used in Public-key encryption process
KU_b	A public key of user B used in Public-key encryption process
||	Concatenation
Z	Compression Function
Z^{-1}	Decompression Function

Security of Public Key Schemes

- Like secret key schemes, brute force **exhaustive search** attack is always theoretically possible

- Keys used are too large (>512bits) and require the use of **very large numbers**
  - Hence it is **slow** compared to private key schemes

- Security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalysis) problems

Key management

Symm. Encr.	Asymm. Encr.
It's mandatory memorize a high number of keys: *n*-1	Only it's needed memorize the endpoint's public key.

Concerning the key management, it will be much more efficient the public key systems because the symmetric key encryption doesn't enable an efficient management.

Key space and size

Symm. Encr.	Asymm. Encr.
Because the kind of encryption, the key will be about hundreds of bits: ≥ 128 bits.	Because the kind of encryption, the key will be about thousands of bits: ≥ 1024 bits.

Concerning the key space, it can not be compared both symmetric and asymmetric encryptions.

Key lifetime

Symm. Encr.	Asymm. Encr.
Short lifetime. Usually it only uses like a session key (seconds or minutes).	Long lifetime. It's managed by a third party (months or years).

The secret key lifetime is shorter than public key. The session key is randomly choosen but in asymmetric encryption the user is the person who choose the keys.

Authentication problem

- Authentication conditions:
  - a) User A must protect itself against sent messages to user B by an unknown third user. This is called masquerade or sender authentication problem.
  - b) User A must protect itself against forged messages by user B that assures he received them with a digital signature from user A. This is called message forgery or message authentication problem (Digital Signature problem).

Authentication

Authentication	
Symm. Encr.	Asymm. Encr.
It provides confidentiality but it's difficult to autheticate the sender.	It provides confidentiality and authentication thanks to both the public and private keys.

https://crypto.stackexchange.com/questions/66225/message-authentication-vs-entity-authentication

The symmetric encryption needs a third party to obtain an authentication. The asymmetric encryption achieves authentication in a easy and clear way.

Encryption performance

Encryption performance	
Symm. Encr.	Asymm. Encr.
The encryption speed is very fast (hundreds of Mbytes/sec in hw).	The encryption speed is very slow. It is used for key exchange and digital signature (hundreds of Kbytes/sec in hw).

Symmetric encryption is 100 or 1000 times faster than asymmetric encryption. In software, the performance is lower.

Summary of symmetric vs asymmetric encryption

Symmetric encryption

- ✈Confidentiality
- ✈Partial Authentication
- ✈Without digital signature
- ✈Keys:
  - Small size
  - Short lifetime
  - Many number of keys
- ✈Fast encryption

Asymmetric encryption

- ✈Confidentiality
- ✈Total Authentication
- ✈With digital signature
- ✈Keys:
  - Big size
  - Long lifetime
  - Few number of keys
- ✈Slow encryption

RSA

- By Rivest, Shamir & Adleman of MIT in 1977
- Best known & widely used public-key scheme
- Based on modular exponentiation over integers modulo a number
  - Exponentiation takes $O((\log n)^3)$ operations (easy)
- Uses large integers (ex. 1024 bits ≈ 308 digits)
- Security due to cost of factoring large numbers
  - Factorization takes $O(e^{\log n \log \log n})$ operations (hard)
- Encryption and decryption are of the following form:

$$C = M^e \bmod n$$

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

Foundation: discrete logarithm is very difficult

M-plaintext, C-ciphertext,
e- private key, d & n-public key

RSA Key Setup

- Each user generates a public/private key pair by:

1. Selecting two large primes at random - `p, q`
2. Computing the system modulus `n=p.q`
   - Note `ø(n)=(p-1)(q-1)`
3. Selecting at random the encryption key `e`
   - Where `1<e<ø(n), gcd(e,ø(n))=1`
4. Solve following equation to find decryption key `d`
   - e.d = 1 mod ø(n) and 0≤d≤n
5. Build the keys
   - Publish their public encryption key: KU={*e*,*n*}
   - Keep secret private decryption key: KR={*d*,*p*,*q*}

RSA Algorithm

Key Generation by Alice	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Choosing e

Let’s consider p=3 and q=7. What choices of e are acceptable?

In this case (p-1)(q-1) = 2 x 6 = 12. Any suitable choice of e must have the property that there are no numbers that neatly divide into e and 12 except for 1. Let’s just try them all out:

- **e=2**: this is no good, since 2 divides both e and 12. In fact this will be true for all multiples of 2 as well, so **e=4**, **e=6**, **e=8** and **e=10** are also not possible.
- **e=3**: this is no good, since 3 divides both e and 12. In fact this will be true for all multiples of 3 as well, so **e=6** and **e=9** are also not possible.
- The remaining choices are **e=5, e=7** and **e=11**. Since in each case there is no number that divides into them and 12 other than 1, all these choices of e are possible.

https://es.wikipedia.org/wiki/RSA

RSA Use

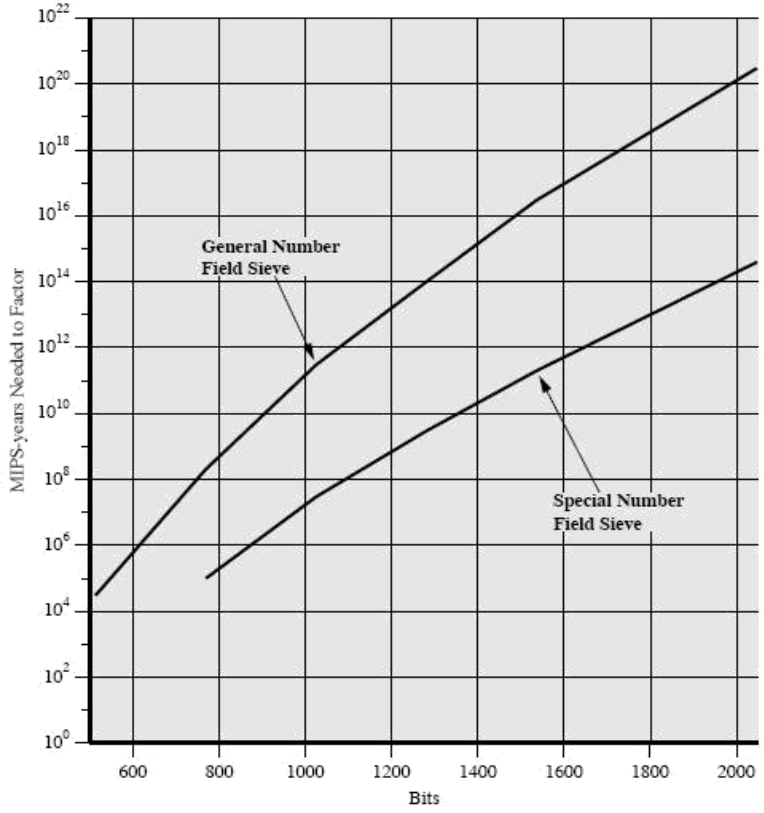
- To encrypt a message M the sender:
  - Obtains **public key** of recipient KU={e,n}
  - Computes: $C=M^e \bmod n$, where $0 \leq M < n$
  - Note that the message *M* must be smaller than the modulus *n*
- To decrypt the ciphertext C the owner:
  - Uses their private key KR={d,p,q}
  - Computes: $M=C^d \bmod n$

- Easy to calculate M^e and C^d
- Infeasible to determine *d* given *C* and *{e,n}*

Factoring Problem

- Mathematical approach takes 3 forms:
  - Factor n=p.q, hence find ø(N) and then *d*
  - Determine ø(N) (given *n*) directly and find *d*
  - Find *d (*given *e* and *n)* directly

- Currently believe all equivalent to factoring
- http://www.rsa.com/rsalabs/node.asp?id=2092



Number of Decimal Digits	Approximate Number of Bits	Date Achieved	MIPS-years	Algorithm
100	332	April 1991	7	quadratic sieve
110	365	April 1992	75	quadratic sieve
120	398	June 1993	830	quadratic sieve
129	428	April 1994	5000	quadratic sieve
130	431	April 1996	1000	generalized number field sieve
140	465	February 1999	2000	generalized number field sieve
155	512	August 1999	8000	generalized number field sieve

Other Public Cryptosystems

- Elliptic curve cryptosystems
  - Candidate replacement for public key cryptography
    - TLS 1.3 uses ECC instead RSA
  - It allows for shorter keys and greater efficiency
- ElGamal
  - Choose a prime p, and two random numbers g, $x < p$
  - Public key is $\{g, p, y\}$ where $y = g^x \bmod p$
  - Private key is x. To obtain from public key requires extracting discrete log
    - For any g, p, y: Find x such that $g^x \bmod p = y$
  - Mostly used for signatures but it can be used to encrypt/decrypt
- Diffie-Hellman
  - Key exchange
- DSS
  - Digital Signature Service

Security of RSA

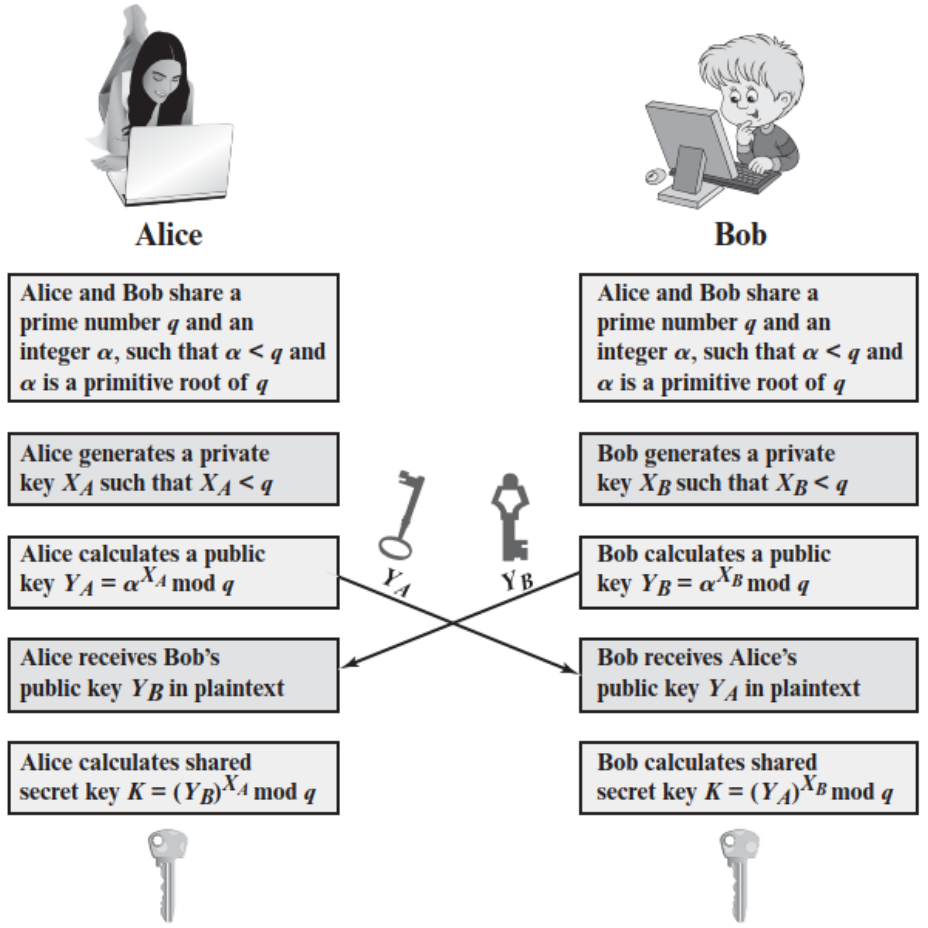
- Since 2015, NIST recommends a minimum of 2048-bit keys for RSA

Number of Decimal Digits	Number of Bits	Date Achieved
100	332	April 1991
110	365	April 1992
120	398	June 1993
129	428	April 1994
130	431	April 1996
140	465	February 1999
155	512	August 1999
160	530	April 2003
174	576	December 2003
200	663	May 2005
193	640	November 2005
232	768	December 2009
240	795	Dec, 2019
250	829	Feb, 2020

Progress in RSA Factorization

Key Exchange

- Diffie–Hellman key exchange



$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

LABORATORY

- Laboratory_02: Python hashing
- Laboratory_03: Hashcat