# Information Security -- Part II
# Asymmetric Ciphers

Frank Yeong-Sung Lin
Information Management Department
National Taiwan University

# Outline

- Introduction to information security
- Introduction to public-key cryptosystems
- RSA
- Diffie-Hellman key exchange
- ECC
- Mutual trust
  - Key management
  - User authentication

# Areas Considered by Info. Security

- Secrecy (Confidentiality): keep information unrevealed
- Authentication: determine the identity of whom you are talking to
- Nonrepudiation: make sure that someone cannot deny the things he/she had done
- Integrity control: make sure the message you received has not been modified
- Availability: make sure the resource be available for authorized personnel when needed

# Essential Concepts for Info. Security

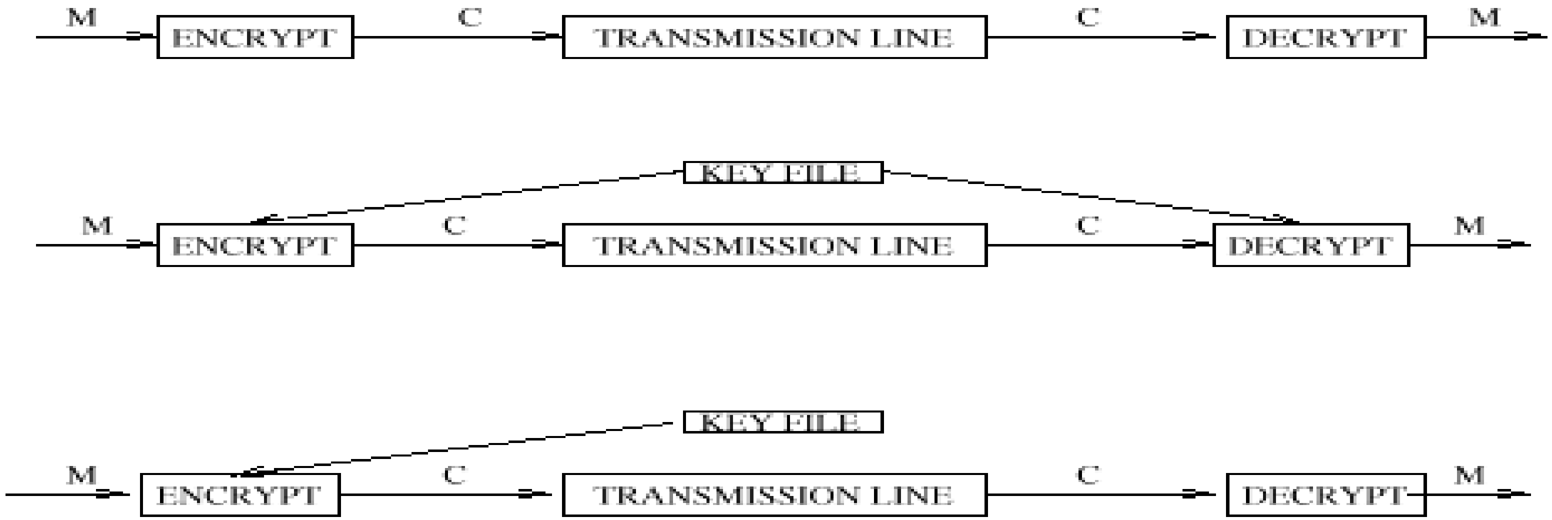
- Risk management
  - threats, vulnerabilities, assets, damages and probabilities
  - balancing acts
  - all cryptosystems may be compromised (trade-off between overhead and expected time span of protection)
- Notion of chains (Achilles' heel)
- Notion of buckets (products, policies, processes and people)
- Defense in-depth
- Average vs. worst cases
- Backup, restoration and contingency plans

# A Number of Interesting Ciphers

- Chinese poems
- Clubs and leather stripes
- Invisible ink (steganography in general)
- Books
- Code books
- Enigma
- XOR (can be considered as an example of symmetric cryptosystems)
- Ej/vu3z8h96
- Scramblers (physical and application layers)

# Principles of Public-Key Cryptosystems

**Public vs Nonpublic** Unlike Private key cryptography, there is no need to share keys. Instead, there is a public "phone number" available to any potential user and a private key.



# Principles of Public-Key Cryptosystems (cont'd)

- Requirements for PKC
  - easy for B (receiver) to generate $KU_b$ and $KR_b$
  - easy for A (sender) to calculate $C = E_{KUb}(M)$
  - easy for B to calculate $M = D_{KRb}(C) = D_{KRb}(E_{KUb}(M))$
  - infeasible for an opponent to calculate $KR_b$ from $KU_b$
  - infeasible for an opponent to calculate M from C and $KU_b$
  - (useful but not necessary) $M = D_{KRb}(E_{KUb}(M)) = E_{KUb}(D_{KRb}(M))$ (true for RSA and good for authentication)

# Principles of Public-Key Cryptosystems (cont'd)

## TRAPDOOR

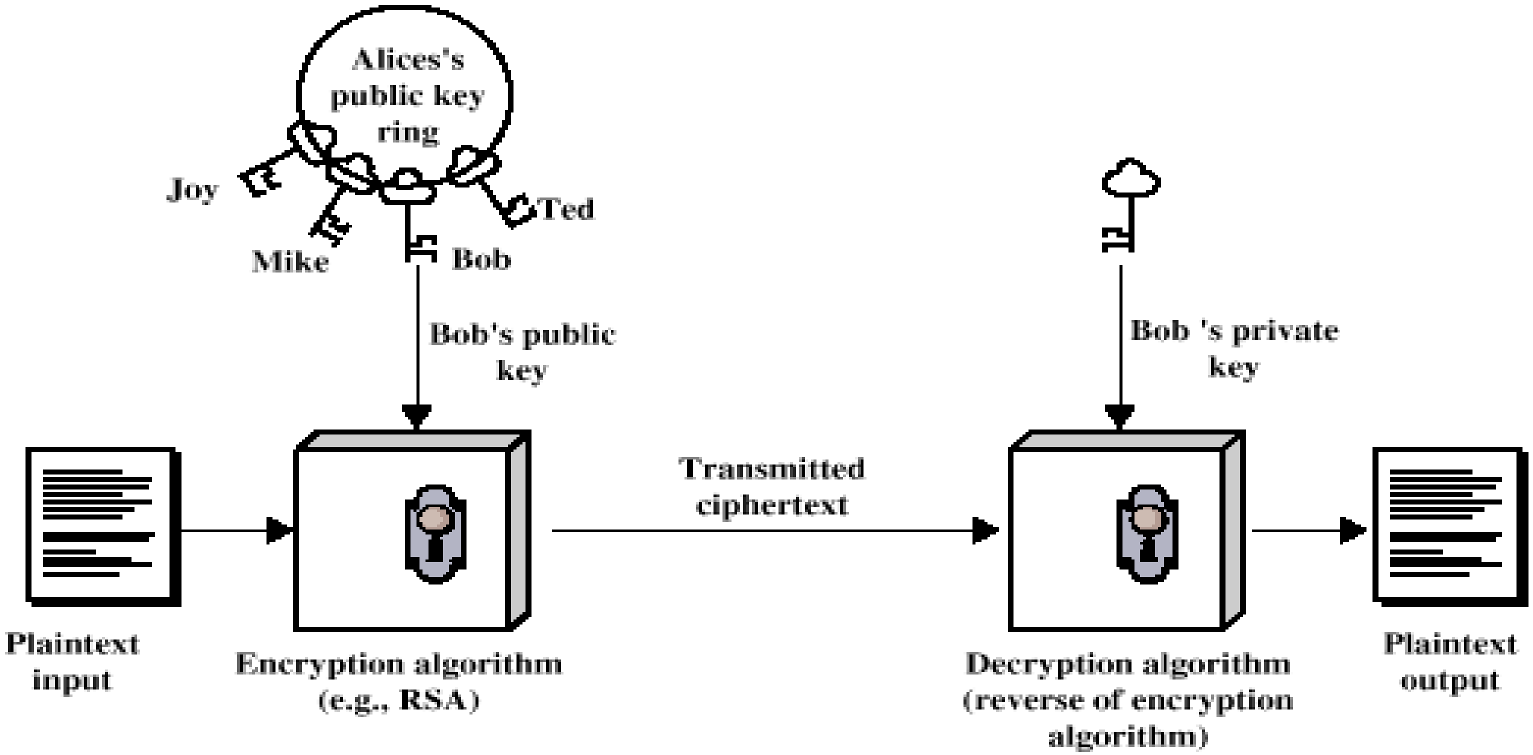
Public Key Cryptography (PKC) is based on the idea of a **trapdoor** function $f : X \longrightarrow Y$, i.e.,

- $f$ is one-to-one,
- $f$ is easy to compute,
- $f$ is public,
- $f^{-1}$ is difficult to compute,
- $f^{-1}$ becomes easy to compute if a trapdoor is known.

# Principles of Public-Key Cryptosystems (cont'd)

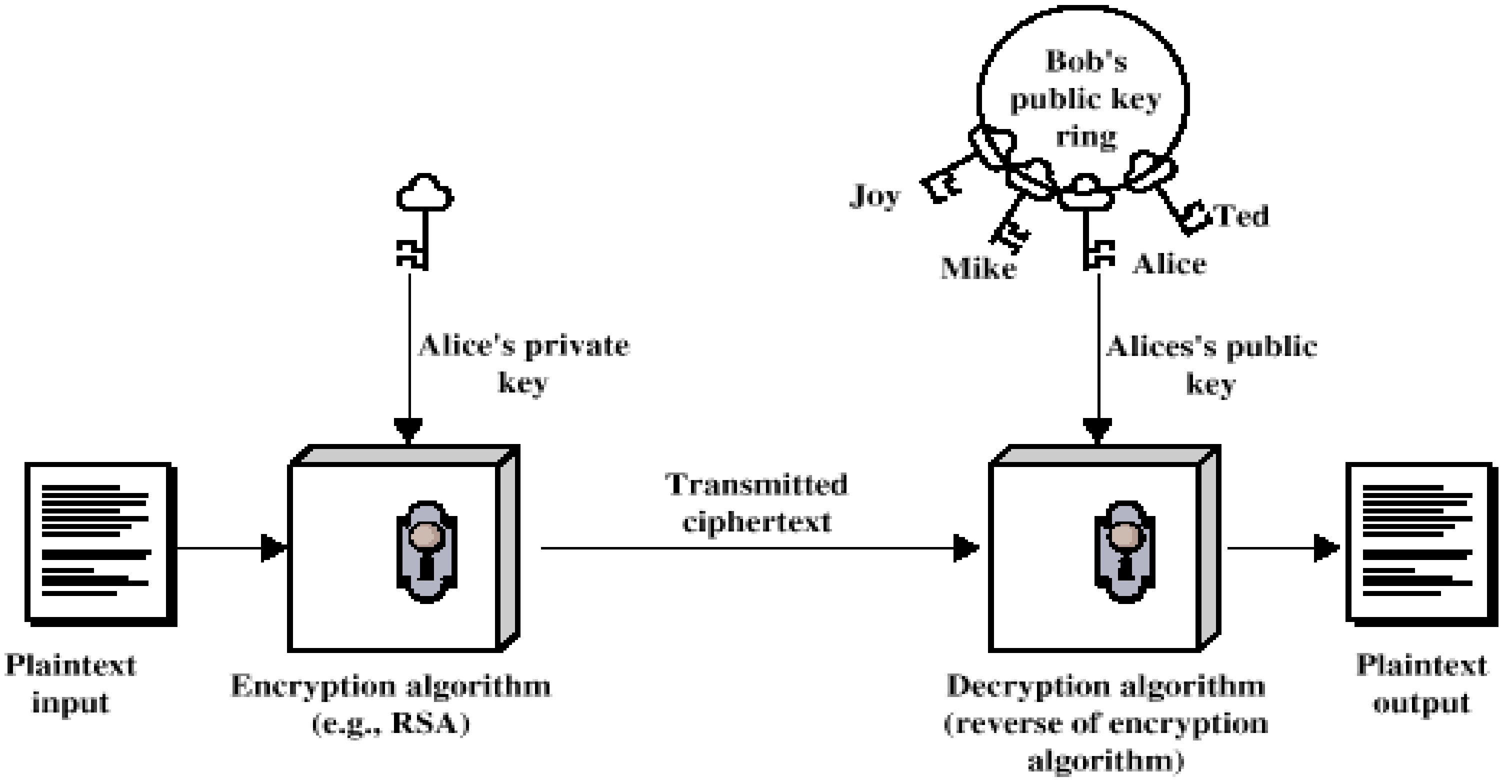
- The idea of PKC was first proposed by Diffie and Hellman in 1976.
- Two keys (public and private) are needed.
- The difficulty of calculating $f^{-1}$ is typically facilitated by
  - factorization of large numbers
  - resolution of NP-completeness
  - calculation of discrete logarithms
- High complexity confines PKC to key management and signature applications

# Principles of Public-Key Cryptosystems (cont'd)




(a) Encryption

# Principles of Public-Key Cryptosystems (cont'd)



(b) Authentication

# Principles of Public-Key Cryptosystems (cont'd)

- Comparison between conventional (symmetric) and public-key (asymmetric) encryption

| Conventional Encryption | Public-Key Encryption |
|---|---|
| **Needed to Work:** | **Needed to Work:** |
| 1. The same algorithm with the same key is used for encryption and decryption.<br>2. The sender and receiver must share the algorithm and the key. | 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption.<br>2. The sender and receiver must each have one of the matched pair of keys (not the same one). |
| **Needed for Security:** | **Needed for Security:** |
| 1. The key must be kept secret.<br>2. It must be impossible or at least impractical to decipher a message if no other information is available.<br>3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key. | 1. One of the two keys must be kept secret.<br>2. It must be impossible or at least impractical to decipher a message if no other information is available.<br>3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key. |

# Principles of Public-Key Cryptosystems (cont'd)

- Applications for PKC
  - encryption/decryption
  - digital signature
  - key exchange

| **Algorithm** | **Encryption/Decryption** | **Digital Signature** | **Key Exchange** |
|---|---|---|---|
| RSA | Yes | Yes | Yes |
| Diffie-Hellman | No | No | Yes |
| DSS | No | Yes | No |

# Principles of Public-Key Cryptosystems (cont'd)

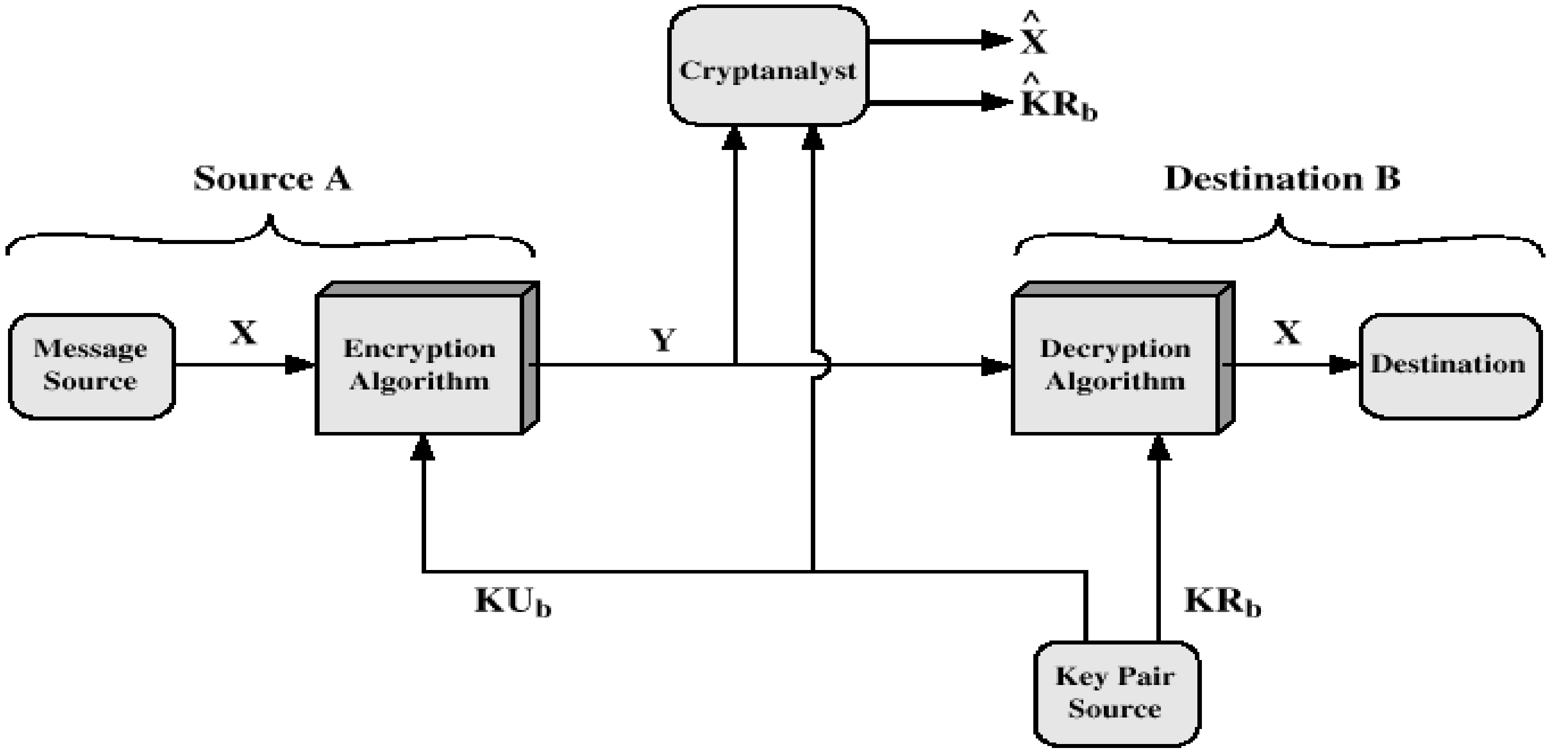



**Figure 6.2 Public-Key Cryptosystem: Secrecy**

# Principles of Public-Key Cryptosystems (cont'd)

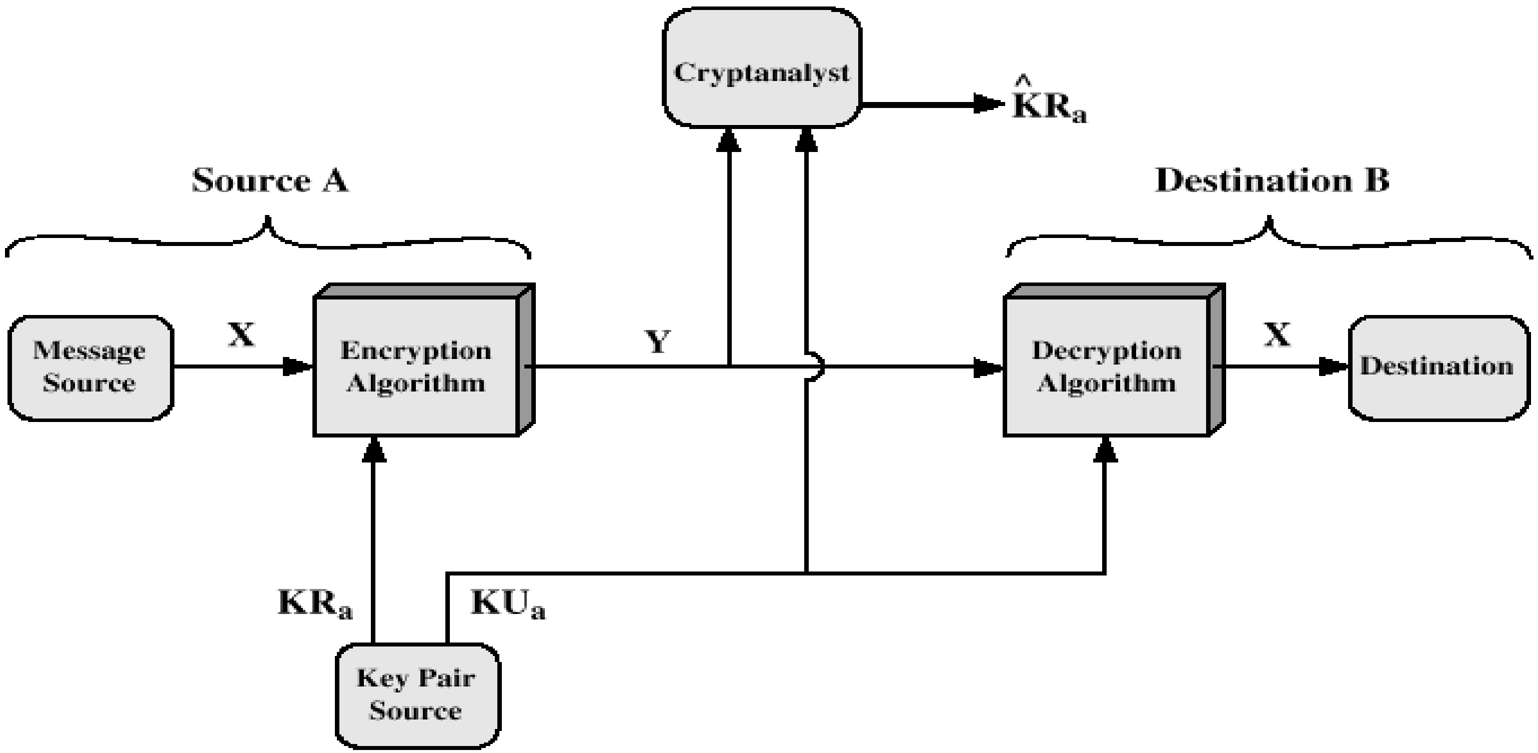



**Figure 6.3 Public-Key Cryptosystem: Authentication**

# Principles of Public-Key Cryptosystems (cont'd)

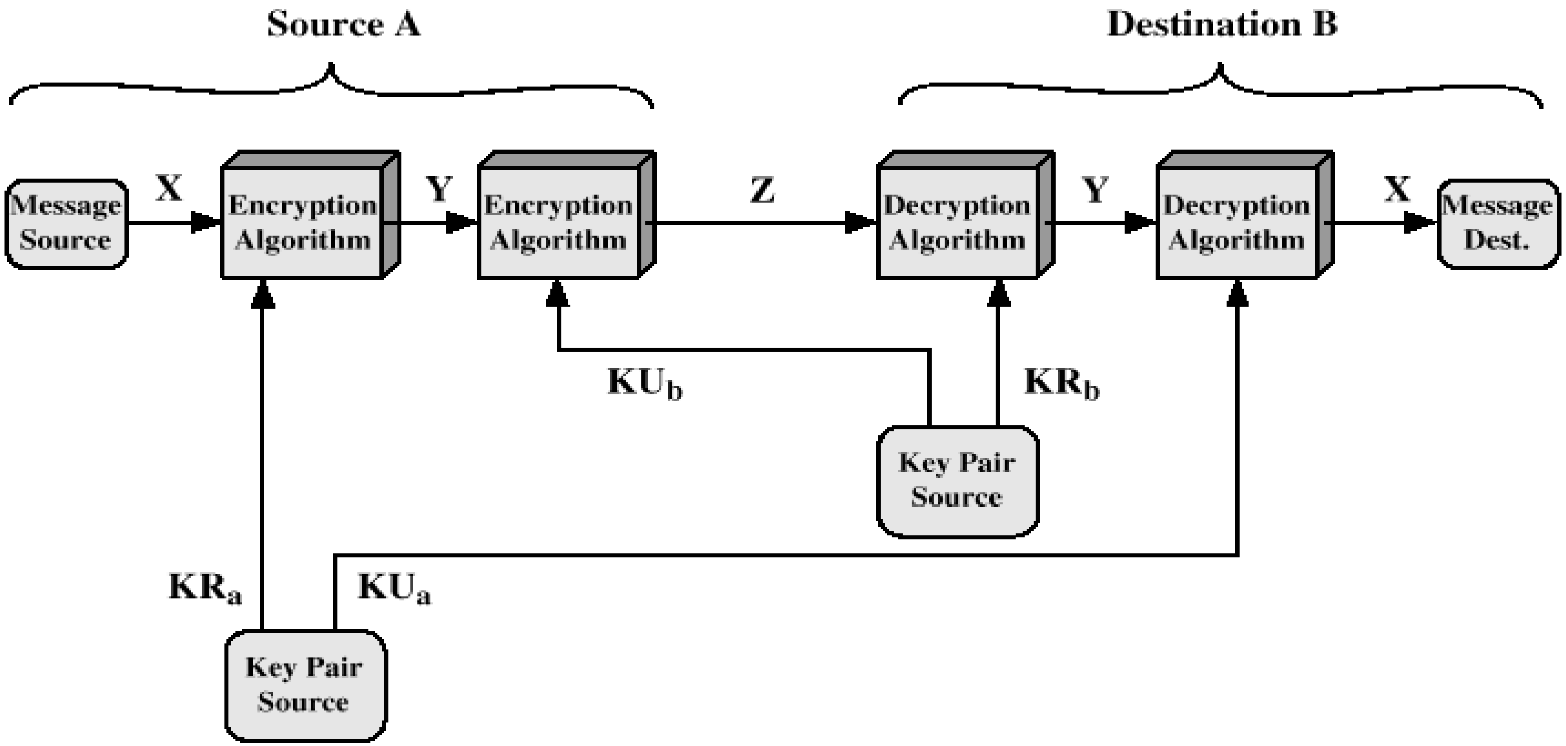



**Figure 6.4 Public-Key Cryptosystem: Secrecy and Authentication**

# The RSA Algorithm

- Developed by Rivest, Shamir, and Adleman at MIT in 1978
- First well accepted and widely adopted PKC algorithm
- Security based on the difficulty of factoring large numbers
- Patent expired in 2001

# The RSA Algorithm (cont'd)

## EULER'S TOTIENT FUNCTION

$\phi(n)$ is the number of non-negative integers less than $n$ which are relatively prime* to $n$.

| $n$ | $\phi(n)$ | $n$ | $\phi(n)$ | $n$ | $\phi(n)$ |
|---|---|---|---|---|---|
| 1 | 0 | 10 | 4 | 19 | 18 |
| 2 | 1 | 11 | 10 | 20 | 8 |
| 3 | 2 | 12 | 4 | 21 | 12 |
| 4 | 2 | 13 | 12 | 22 | 10 |
| 5 | 4 | 14 | 6 | 23 | 22 |
| 6 | 2 | 15 | 8 | 24 | 8 |
| 7 | 6 | 16 | 8 | 25 | 20 |
| 8 | 4 | 17 | 16 | 26 | 12 |
| 9 | 4 | 18 | 6 | 27 | 18 |

**Some Important Values of $\phi(n)$:**

| $n$ | $\phi(n) =$ | Conditions |
|---|---|---|
| $p$ | $p-1$ | $p$ prime |
| $p^n$ | $p^n - p^{n-1}$ | $p$ prime |
| $s \cdot t$ | $\phi(s) \cdot \phi(t)$ | $\gcd(s,t) = 1$ |
| $p \cdot q$ | $(p-1) \cdot (q-1)$ | $p, q$ prime |

***互質**，又稱**互素**。若 $N$ 個整數的最大公因數是1，則稱這 $N$ 個整數互質。

# The RSA Algorithm (cont'd)

## RSA CRYPTOSYSTEM

$n$,$p$,$q$: Define $n = pq$ where $p$ and $q$ are large primes.

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

$M$: $M$ is the number representing the message to be encrypted.

$C$: $C$ is the number representing the "Cypher-text" (i.e., the encrypted text).

**Public Information:** $n, e$.

**Private Information:** $d$.

# The RSA Algorithm (cont'd)

| Key Generation | |
|---|---|
| Select $p$, $q$ | $p$ and $q$ both prime |
| Calculate $n = p \times q$ | |
| Calculate $\phi(n) = (p - 1)(q - 1)$ | |
| Select integer $e$ | $\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$ |
| Calculate $d$ | $d = e^{-1} \bmod \phi(n)$ |
| Public key | $KU = \{e, n\}$ |
| Private key | $KR = \{d, n\}$ |

| Encryption | |
|---|---|
| Plaintext: | $M < n$ |
| Ciphertext: | $C = M^e \pmod{n}$ |

| Decryption | |
|---|---|
| Ciphertext: | $C$ |
| Plaintext: | $M = C^d \pmod{n}$ |

**The RSA Algorithm**

# The RSA Algorithm (cont'd)

## PRIMES

An integer $n > 1$ is prime if 1 and $n$ are its only divisors.

**Euclid:** There are infinitely many primes.
If $p_1 < p_2 < \cdots < p_n$ are the first $n$ primes then any prime divisor of the integer $1 + p_1 p_2 \cdots p_n$ must be larger than $p_n$.

The number $\pi(n)$ of primes $\leq n$ is asymptotically equal to $\frac{n}{\ln n}$.

# The RSA Algorithm (cont'd)

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199
211 223 227 229 233 239 241 251 257 263 269 271 281 283 293
307 311 313 317 331 337 347 349 449 457 461 463 467 479 487 491 499
401 409 419 421 431 433 439 443 449 457 461 463 467 479 487 491 499
503 509 521 523 541 547 557 563 569 571 577 587 593 599
601 607 613 617 619 631 641 643 647 653 659 661 673 677 683 691
701 709 719 727 733 739 743 751 757 761 769 773 787 797
809 811 821 823 827 829 839 853 857 859 863 877 881 883 887
907 911 919 929 937 941 947 953 967 971 977 983 991 997
1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063 1069 1087 1091 1093 1097
1103 1109 1117 1123 1129 1151 1153 1163 1171 1181 1187 1193
1201 1213 1217 1223 1229 1231 1237 1249 1259 1277 1279 1283 1289 1291 1297
1301 1303 1307 1319 1321 1327 1361 1367 1373 1381 1399
1409 1423 1427 1429 1433 1439 1447 1451 1453 1459 1471 1481 1483 1487 1489 1493 1499
1511 1523 1531 1543 1549 1553 1559 1567 1571 1579 1583 1597
1601 1607 1609 1613 1619 1621 1627 1637 1657 1663 1667 1669 1693 1697 1699
1709 1721 1723 1733 1741 1747 1753 1759 1777 1783 1787 1789
1801 1811 1823 1831 1847 1861 1867 1871 1873 1877 1879 1889
1901 1907 1913 1931 1933 1949 1951 1973 1979 1987 1993 199

Primes under 2000

# The RSA Algorithm (cont'd)

- The above statement is referred to as the *prime number theorem*, which was proven in 1896 by Hadaward and Poussin.

| $x$ | $\pi(x)$ | $x/\ln x$ | $(\pi(x) \times \ln x)/x$ |
|---|---|---|---|
| $10^3$ | 168 | 144.8 | 1.160 |
| $10^4$ | 1229 | 1085.7 | 1.132 |
| $10^5$ | 9592 | 8685.9 | 1.104 |
| $10^6$ | 78498 | 74382.4 | 1.085 |
| $10^7$ | 664579 | 620420.7 | 1.071 |
| $10^8$ | 5761455 | 5428681.0 | 1.061 |
| $10^9$ | 50847534 | 48254942.4 | 1.054 |
| $10^{10}$ | 455052512 | 434294481.9 | 1.048 |

# The RSA Algorithm (cont'd)

- Whether there exists a simple formula to generate prime numbers?
- An ancient Chinese mathematician conjectured that if $n$ divides $2^n - 2$ then $n$ is prime. For $n = 3$, 3 divides 6 and $n$ is prime. However, for $n = 341 = 11 \times 31$, $n$ dives $2^{341} - 2$.
- Mersenne suggested that if $p$ is **prime** then $M_p = 2^p - 1$ is prime. This type of primes are referred to as Mersenne primes*. Unfortunately, for $p = 11$, $M_{11} = 2^{11} - 1 = 2047 = 23 \times 89$.

# The RSA Algorithm (cont’d)

*In mathematics, a **Mersenne number** is a positive integer that is one less than a power of two:

$M_n = 2^n - 1.$

Some definitions of Mersenne numbers require that the exponent *n* be prime.

A **Mersenne prime** is a Mersenne number that is prime. As of September 2008, only 46 Mersenne primes are known; the largest known prime number ($2^{43,112,609} - 1$) is a Mersenne prime, and in modern times, the largest known prime has almost always been a Mersenne prime. Like several previously-discovered Mersenne primes, it was discovered by a distributed computing project on the Internet, known as the *Great Internet Mersenne Prime Search* (GIMPS). It was the first known prime number with more than 10 million digits.

# The RSA Algorithm (cont'd)

- Fermat conjectured that if $F_n = 2^{2^n} + 1$, where $n$ is a non-negative integer, then $F_n$ is prime. When $n$ is less than or equal to 4, $F_0 = 3$, $F_1 = 5$, $F_2 = 17$, $F_3 = 257$ and $F_4 = 65537$ are all primes. However, $F_5 = 4294967297 = 641 \times 6700417$ is not a prime number.
- $n^2 - 79n + 1601$ is valid only for $n < 80$.
- There are an infinite number of primes of the form $4n + 1$ or $4n + 3$.
- There is no simple way so far to gererate prime numbers.

# The RSA Algorithm (cont'd)

**Bertrand's Postulate** For any integer there is always a prime between $n+1$ and $2n$. A beautiful elementary proof is due to Erdös.

**Open problem of Hardy and Wright:** Is there a prime between $n^2$ and $(n+1)^2$?

# The RSA Algorithm (cont'd)

- Prime gap: displacement between two consecutive prime numbers
  - 0 the smallest
  - unbounded from above
  - $n!+2$ (devisable by 2), $n!+3$ (devisable by 3, $n!+4$ (devisable by 4),…, $n!+n$ (devisable by $n$) are not prime

# The RSA Algorithm (cont'd)

- Format's Little Theorem (to be proven later): If $p$ is prime and $a$ is a positive integer not divisible by $p$, then

$$a^{p\text{-}1} \equiv 1 \bmod p.$$

Example: $a = 7$, $p = 19$

$$7^2 = 49 \equiv 11 \bmod 19$$

$$7^4 = 121 \equiv 7 \bmod 19$$

$$7^8 = 49 \equiv 11 \bmod 19$$

$$7^{16} = 121 \equiv 7 \bmod 19$$

$$a^{p\text{-}1} = 7^{18} = 7^{16+2} \equiv 7{\times}11 \equiv 1 \bmod 19$$

# The RSA Algorithm (cont'd)

## HOW IT WORKS

**RSA Encryption:** $M \longrightarrow E(M) := M^e \equiv C \bmod n$

**RSA Decryption:** $C \longrightarrow D(C) := C^d \equiv M \bmod n$.

**When and Why it Works:** Recall that $\phi(n) = (p-1)(q-1)$. For RSA to work $M < n$, $\gcd(e, (p-1)(q-1)) = 1$, $p$ and $q$ are prime and $de \equiv 1(\bmod (p-1)(q-1))$.

**RSA works because:** $C^d \equiv (M^e)^d \equiv M^{ed} \equiv M^{1+k(p-1)(q-1)} (\bmod n)$

Assume that $\gcd(M, q) = \gcd(M, p) = 1$. Then by Fermat's Little Theorem:

$$C^d \equiv M(M^{p-1})^{k(q-1)} \equiv M(1)^{k(p-1)} \equiv M(\bmod p)$$
$$C^d \equiv M(M^{q-1})^{k(p-1)} \equiv M(1)^{k(q-1)} \equiv M(\bmod q)$$

Therefore $C^d \equiv M(\bmod n)$.

# The RSA Algorithm (cont'd)

- $A = M+ip$ for a non-negative integer $i$.
- $A = M+jq$ for a non-negative integer $j$.
- From the above two equations, $ip = jq$.
- Then, $i = kq$. ($p$ and $q$ are primes.)
- Consequently, $A = M+ip = M+kpq$. *Q.E.D. (quod erat demonstrandum)*

# The RSA Algorithm (cont'd)

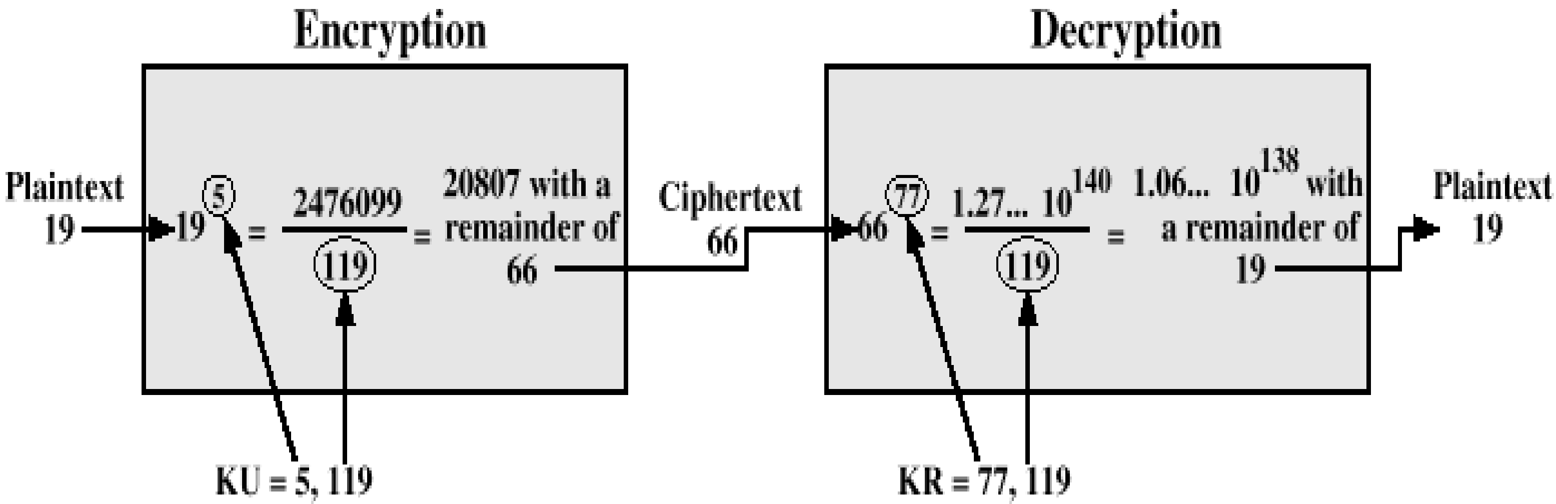



**Figure 6.6 Example of RSA Algorithm**

# The RSA Algorithm (cont'd)

- Example 1
  - Select two prime numbers, $p = 7$ and $q = 17$.
  - Calculate $n = p \times q = 7 \times 17 = 119$.
  - Calculate $\Phi(n) = (p-1)(q-1) = 96$.
  - Select $e$ such that $e$ is relatively prime to $\Phi(n) = 96$ and less than $\Phi(n)$; in this case, $e = 5$.
  - Determine $d$ such that $d \times e \equiv 1 \bmod 96$ and $d < 96$.The correct value is $d = 77$, because $77 \times 5 = 385 = 4 \times 96 + 1$.

# The RSA Algorithm (cont'd)

- **Example 2:** $p = 101, q = 113, n = 11413$. Then $\phi(n) = (p-1)(q-1) = 11200 = 2^6 5^2 7$. So any integer not divisible by $2, 5, 7$ can be used as a public key. We can choose $e = 3533$. Using the Euclidean algorithm we easily compute $e^{-1} \bmod 11200 = 6597$.

# The RSA Algorithm (cont'd)

## OPERATIONS ON NUMBERS

**Addition** of two $k$-bit numbers can be done in time $O(k)$.

$$\begin{array}{r} 010110101 \\ 11010010 \\ \hline 110000111 \end{array}$$

**Multiplication** of two $k$-bit numbers can be done in time $O(k^2)$.

$$\begin{array}{r} 1011 \\ 110 \\ \hline 0000 \\ 1011\phantom{0} \\ 1011\phantom{00} \\ \hline 100010 \end{array}$$

Both are well-known algorithms. Of course there are "faster" algorithms (see Knuth's: "Art of Computer Programming").

**Exponentiation** of two $k$-bit numbers can be done in time $O(k^3)$.

# The RSA Algorithm (cont'd)

**Example:** $p = 5, q = 7, n = 35$.
Can choose $e = 11$. Let the message be $M = 12$. To compute $12^{11} \bmod 35$.
First write $(11)_{10} = (1011)_2$. Then calculate

$$
\begin{aligned}
M^{11} &= M^{1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0} \\
&= (M^{1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0})^2 M \\
&= ((M^{1\cdot 2^1 + 0\cdot 2^0})^2 M)^2 M \\
&= ((M^2)^2 M)^2 M
\end{aligned}
$$

The formal algorithm is as follows: Compute the binary representation of $e = \sum_{i=0}^{k-1} e_i 2^i$, where $k = \lceil \log_2 \rceil$ and perform the following algorithm:

**Procedure** *exponentiation* $(x, e, n)$
$z := 1$
**for** $i = k - 1$ **downto** 0 **do**
  $z := z^2 \bmod n$
  **if** $e_i = 1$ **then** $z := z \cdot x \bmod n$
**return** $x^e \bmod n$

# The RSA Algorithm (cont'd)

- Key generation
  - determining two large prime numbers, $p$ and $q$
  - selecting either $e$ or $d$ and calculating the other
- Probabilistic algorithm to generate primes
  - [1] Pick an odd integer $n$ at random.
  - [2] Pick an integer $a < n$ ($a$ is clearly not divisible by $n$) at random.
  - [3] Perform the probabilistic primality test, such as Miller-Rabin. If $n$ fails the test, reject the value $n$ and go to [1].
  - [4] If $n$ has passed a sufficient number of tests, accept $n$; otherwise, go to [2].

# The RSA Algorithm (cont'd)

- How may trials on the average are required to find a prime?
  - from the prime number theory, primes near $n$ are spaced on the average one every ($\ln n$) integers
  - even numbers can be immediately rejected
  - for a prime on the order of $2^{200}$, about $(\ln 2^{200})/2 =$ 70 trials are required
- To calculate $e$, what is the probability that a random number is relatively prime to $\Phi(n)$? About 0.6.

# The RSA Algorithm (cont'd)

- For fixed length keys, how many primes can be chosen?
  - for 64-bit keys, $2^{64}/\ln 2^{64} - 2^{63}/\ln 2^{63} \approx 2.05 \times 10^{17}$
  - for 128- and 256-bit keys, $1.9 \times 10^{36}$ and $3.25 \times 10^{74}$, respectively, are available
- For fixed length keys, what is the probability that a randomly selected odd number *a* is prime?
  - for 64-bit keys, $2.05 \times 10^{17}/(0.5 \times (2^{64} - 2^{63})) \approx 0.044$

    (expectation value: $1/0.044 \approx 23$)
  - for 128- and 256-bit keys, 0.022 and 0.011, respectively

# The RSA Algorithm (cont'd)

- The security of RSA
  - brute force: This involves trying all possible private keys.
  - mathematical attacks: There are several approaches, all equivalent in effect to factoring the product of two primes.
  - timing attacks: These depend on the running time of the decryption algorithm.

# The RSA Algorithm (cont'd)

- To avoid brute force attacks, a large key space is required.
- To make $n$ difficult to factor
  - $p$ and $q$ should differ in length by only a few digits (both in the range of $10^{75}$ to $10^{100}$)
  - both $(p$-1) and ($q$-1) should contain a large prime factor
  - gcd($p$-1,$q$-1) should be small
  - should avoid $e << n$ and $d < n^{1/4}$

# The RSA Algorithm (cont'd)

- To make $n$ difficult to factor (cont'd)
  - $p$ and $q$ should best be strong primes, where $p$ is a strong prime if
    - there exist two large primes $p_1$ and $p_2$ such that $p_1|p$-1 and $p_2|p$+1
    - there exist four large primes $r_1$, $s_1$, $r_2$ and $s_2$ such that $r_1|p_1$-1, $s_1|p_1$+1, $r_2|p_2$-1 and $s_2|p_2$+1
  - $e$ should not be too small, e.g. for $e$ = 3 and C = $M^3$ mod $n$, if $M^3 < n$ then M can be easily calculated

# The RSA Algorithm (cont'd)

## FACTORING ALGORITHMS

**Problem:** Factor a given $n$.

This is a very important problem. No efficient algorithm (i.e., running in time polylogarithmic in $n$) is known.

The 1996 challenge referred to an RSA challenge with a key length of 130 decimal digits. Implementation was done on the Internet.

| *Decimal Digits* | *Year Achieved* | *MIPS Years* | *Algorithm* |
|---|---|---|---|
| 100 | 1991 | 7 | *Q Sieve* |
| 110 | 1992 | 75 | *Q Sieve* |
| 120 | 1993 | 830 | *Q Sieve* |
| 130 | 1996 | 500 | *Gen. Num. Field* |

MIPS-Years is Millions of Instructions Per Second counted in Years, e.g. a Pentium 200 is a 50 MIPS machine.

# The RSA Algorithm (cont’d)

- Major threats
  - the continuing increase in computing power (100 or even 1000 MIPS machines are easily available)
  - continuing refinement of factoring algorithms (from QS to GNFS and to SNFS)

# The RSA Algorithm (cont'd)

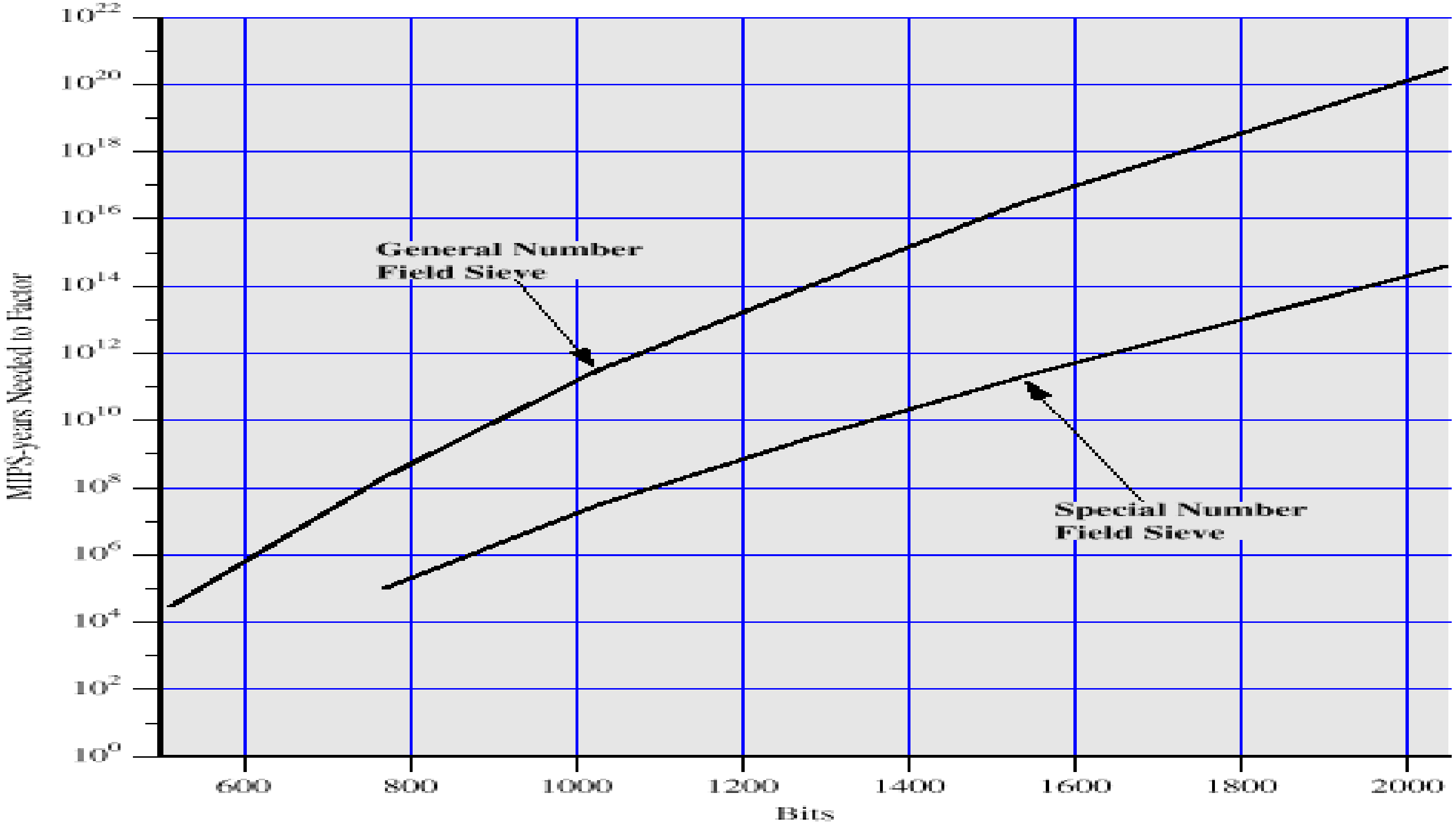



**Figure 6.9 MIPS-years Needed to Factor**

# The RSA Algorithm (cont'd)

**Experimental Running Times**

Key length selection for RSA depends on intended security and expected key lifetime. E.g., if you want your keys to remain secure for 20 years a key 1,024 bits long is too short!

Table for factoring times in NFS and SNFS.

| # of Bits | NFS-MIPS | SNFS-MIPS |
|---|---|---|
| 512 | $3 \cdot 10^4$ | $< 200$ |
| 768 | $2 \cdot 10^8$ | $1 \cdot 10^5$ |
| 1024 | $3 \cdot 10^{11}$ | $3 \cdot 10^7$ |
| 1280 | $1 \cdot 10^{14}$ | $3 \cdot 10^9$ |
| 1536 | $3 \cdot 10^{16}$ | $2 \cdot 10^{11}$ |
| 2048 | $3 \cdot 10^{20}$ | $4 \cdot 10^{14}$ |

To be sure, certainly you can use very large keys, but remember your computation time will become unreasonable! Here are some predictions in bit lengths:

| Year | Individual | Corporation | Government |
|---|---|---|---|
| 2000 | 1024 | 1280 | 1538 |
| 2005 | 1280 | 1538 | 2048 |
| 2010 | 1280 | 1538 | 2048 |
| 2015 | 1538 | 2048 | 2048 |

# The RSA Algorithm (cont'd)

## TIMING ATTACKS ON RSA

This is similar to a burglar observing how long it takes for someone to turn the dial of a safe. It is applicable to other cryptosystems as well.

A cryptanalyst can compute a private key by keeping track of how long it takes the computer to decipher messages. The exponent is computed bit-by-bit starting with the low-end bit.

For a given ciphertext it is possible to time how long it takes to perform modular exponentiation. We can therefore determine unknown bits by exploiting timing differences in responses. (This attack was implemented by Koeher in 1996.)

The problem is eliminated by using any of the following remedies: (a) constant exponentiation time, (b) random delay, or (c) blinding by multiplying the ciphertext with random number prior to exponentiation.

# Diffie-Hellman Key Exchange

- First public-key algorithm published
- Limited to key exchange
- Dependent for its effectiveness on the difficulty of computing discrete logarithm

# Diffie-Hellman Key Exchange (cont'd)

- Define a primitive root of of a prime number $p$ as one whose powers generate all the integers from 1 to $p$-1.
- If $a$ is a primitive root of the prime number $p$, then the numbers

  $a \bmod p,\ a^2 \bmod p,\ \ldots,\ a^{p-1} \bmod p$

  are distinct and consist of the integers from 1 to $p$-1 in some permutation.
- Not every number has a primitive root.
- For example, 2 is a primitive root of 5, but 4 is not.

# Diffie-Hellman Key Exchange (cont'd)

- For any integer $b$ and a primitive root $a$ of prime number $p$, one can find a unique exponent $i$ such that

  $b = a^i \bmod p$, where $0 \le i \le (p\text{-}1)$.

- The exponent $i$ is referred to as the discrete logarithm, or index, of $b$ for the base $a$, mod $p$.
- This value is denoted as $\mathrm{ind}_{a,p}(b)$ $(\mathrm{dlog}_{a,p}(b))$.

# Diffie-Hellman Key Exchange (cont'd)

| Global Public Elements | |
|---|---|
| $q$ | prime number |
| $\alpha$ | $\alpha < q$ and $\alpha$ a primitive root of $q$ |

| User A Key Generation | |
|---|---|
| Select private $X_A$ | $X_A < q$ |
| Calculate public $Y_A$ | $Y_A = \alpha^{X_A} \bmod q$ |

| User B Key Generation | |
|---|---|
| Select private $X_B$ | $X_B < q$ |
| Calculate public $Y_B$ | $Y_B = \alpha^{X_B} \bmod q$ |

**Generation of Secret Key by User A**

$K = (Y_B)^{X_A} \bmod q$

**Generation of Secret Key by User B**

$K = (Y_A)^{X_B} \bmod q$

**The Diffie-Hellman Key Exchange Algorithm**

# Diffie-Hellman Key Exchange (cont'd)

- Example:

  $q = 97$ and a primitive root $a = 5$ is selected.

  $X_A = 36$ and $X_B = 58$ (both < 97).

  $Y_A = 5^{36} = 50 \bmod 97$ and

  $Y_B = 5^{58} = 44 \bmod 97$.

  $K = (Y_B)^{X_A} \bmod 97 = 44^{36} \bmod 97 = 75 \bmod 97$.

  $K = (Y_A)^{X_B} \bmod 97 = 50^{58} \bmod 97 = 75 \bmod 97$.

  75 cannot easily be computed by the opponent.

# Diffie-Hellman Key Exchange (cont'd)

- How the algorithm works

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

# Diffie-Hellman Key Exchange (cont'd)

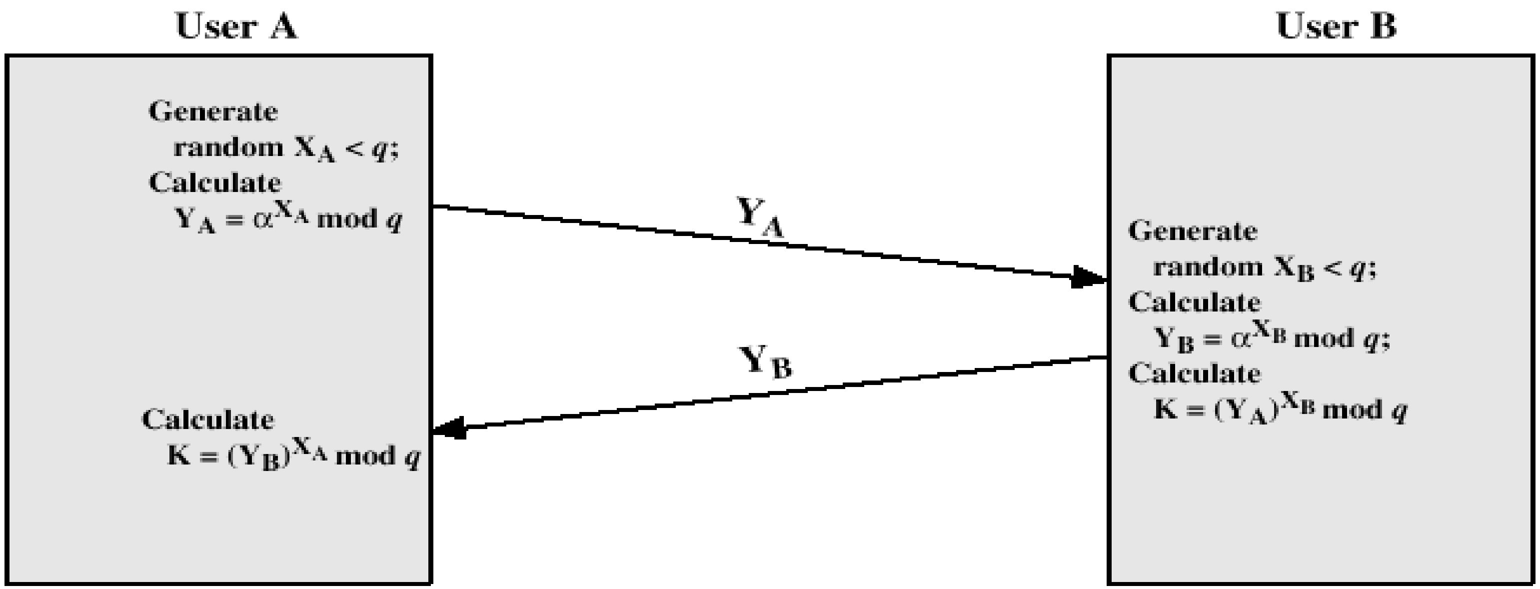



**Figure 6.17 Diffie-Hellman Key Exchange**

# Diffie-Hellman Key Exchange (cont'd)

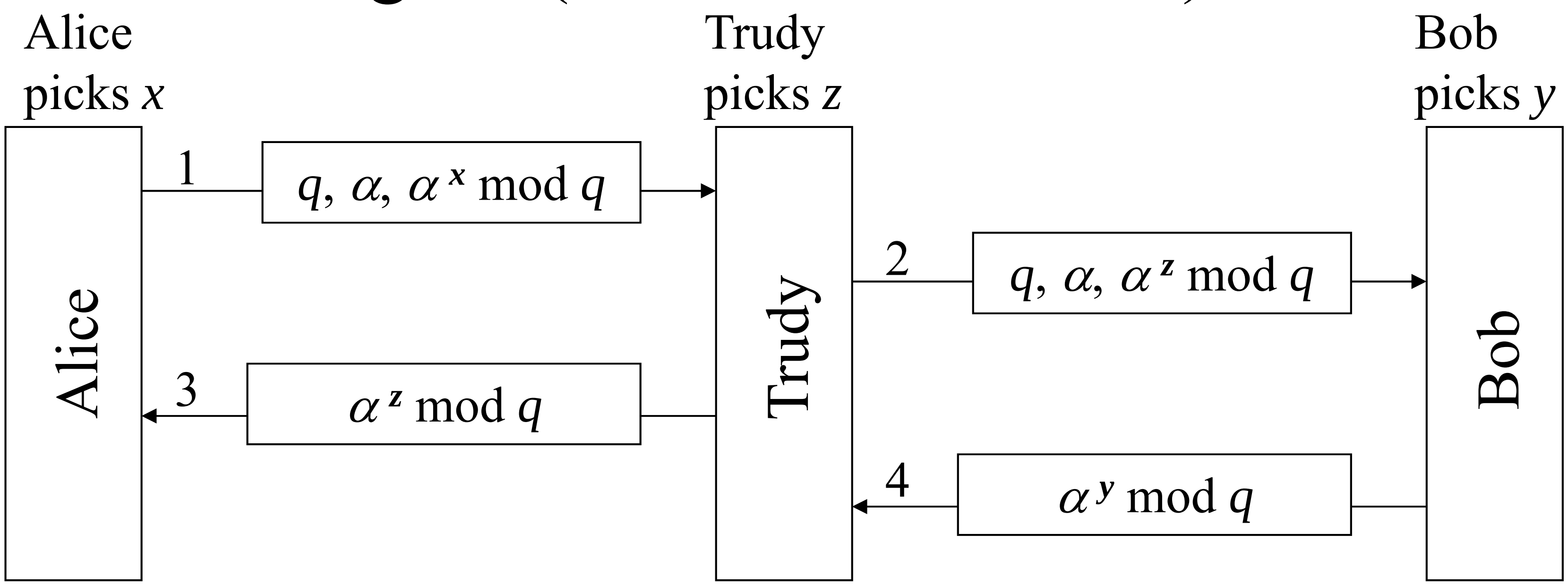
- $q$, $a$, $Y_{\mathrm{A}}$ and $Y_{\mathrm{B}}$ are public.
- To attack the secrete key of user B, the opponent must compute

  $$X_{\mathrm{B}} = \mathrm{ind}_{a,q}(Y_{\mathrm{B}}).\ [Y_{\mathrm{B}} = a^{X_{\mathrm{B}}} \bmod q.]$$

- The effectiveness of this algorithm therefore depends on the difficulty of solving discrete logarithm.

# Diffie-Hellman Key Exchange (cont'd)

- Bucket brigade (Man-in-the-middle) attack




  – ($\alpha^{xz}$ mod $q$) becomes the secret key between Alice and Trudy, while ($\alpha^{yz}$ mod $q$) becomes the secret key between Trudy and Bob.