CSEN1001

Computer and Network Security

Mervat AbuElkheir

Ahmad Helmy

Mohamed Abdelrazik

Lecture (6)

Public Key Cryptography

Public Key Cryptography

- Traditional **private/secret/single key** cryptography uses **one** key
- Shared by both sender and receiver
- If this key is disclosed communications are compromised
- Also is **symmetric**, parties are equal
- Hence does not protect sender from receiver forging a message & claiming it's sent by sender

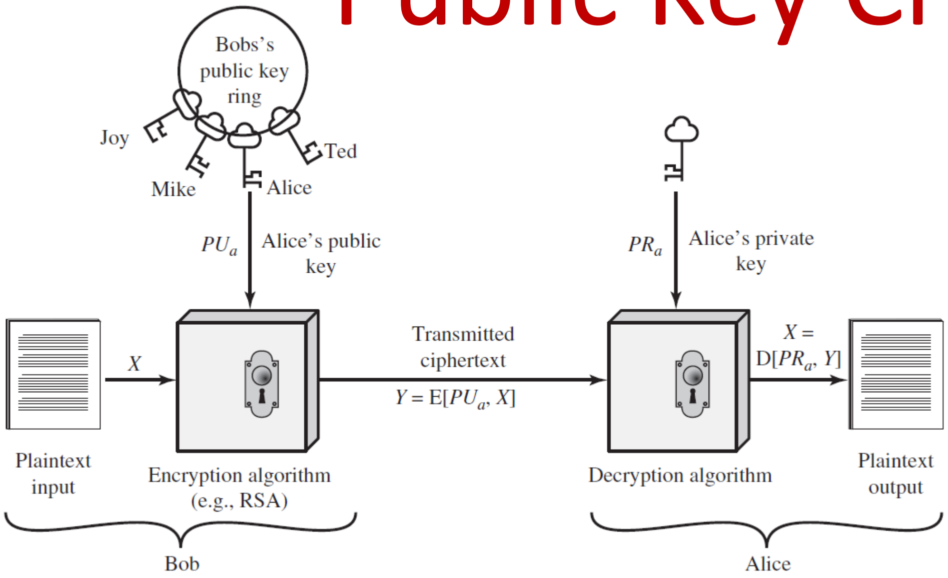
Public Key Cryptography

- Probably most significant advance in the 3000 year history of cryptography
- Uses **two** keys – a public & a private key
- **Asymmetric** since parties are **not** equal
- Uses clever application of number theoretic concepts to function
- Complements **rather than** replaces private key crypto

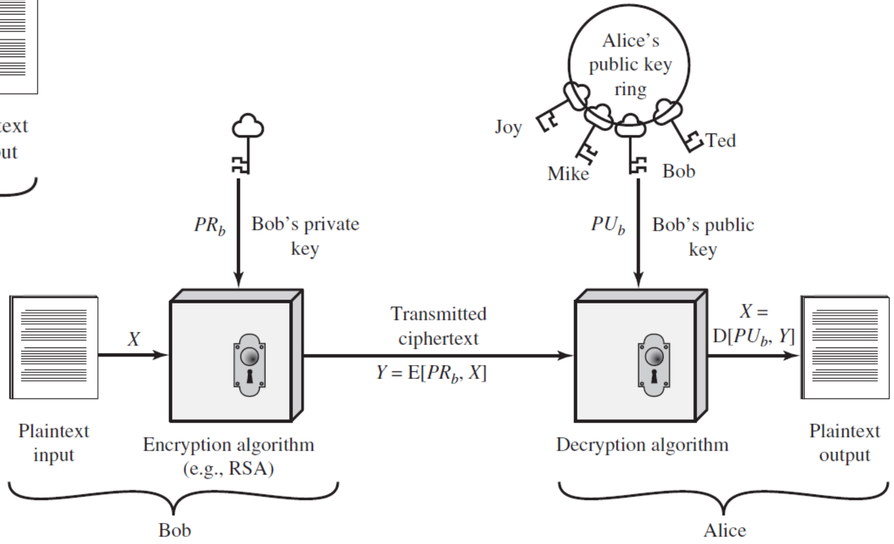
Why Public Key Cryptography

- Developed to address two key issues:
  - **Key distribution** – how to have secure communications in general without having to trust a KDC with your key
  - **Digital signatures** – how to verify that a message comes intact from the claimed sender
- Public invention due to Whitfield Diffie & Martin Hellman at Stanford University in 1976
  - known earlier in classified community

Public Key Cryptography



(a) Encryption with public key

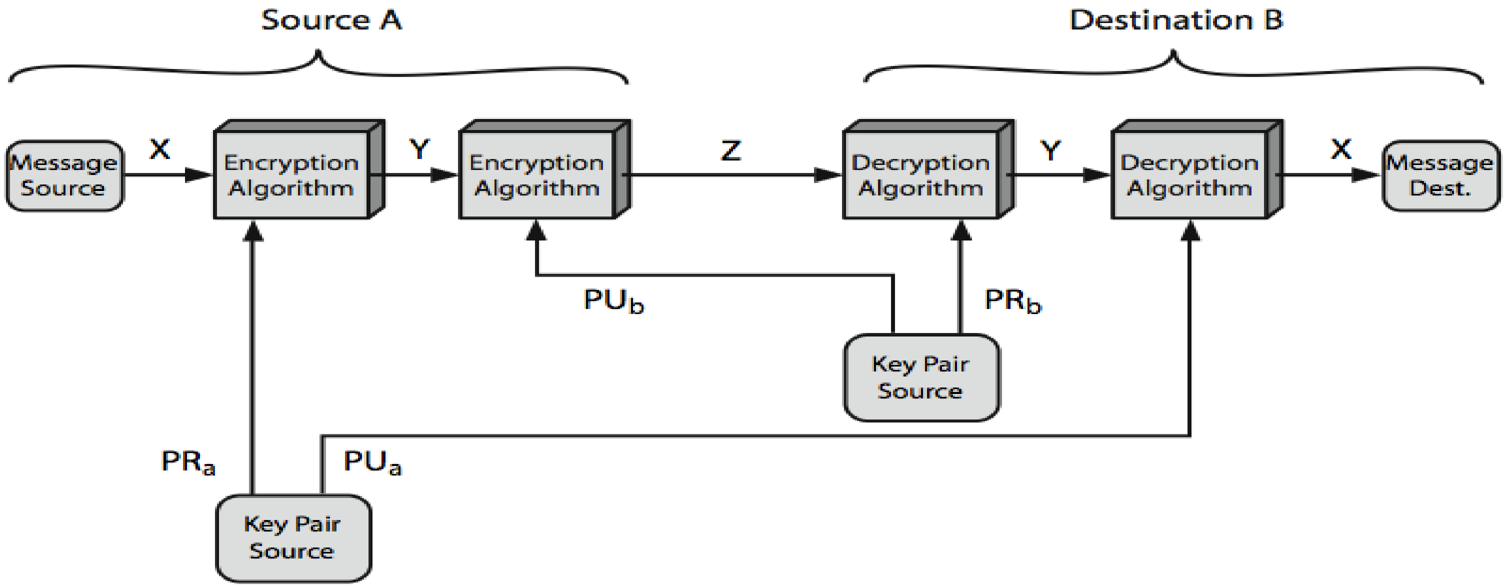


(b) Encryption with private key

Public Key Cryptography

- **Two** keys are related:
  - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - a **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
- Is **asymmetric** because
  - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

Public Key Cryptosystems



Public Key Applications

- Can classify uses into 3 categories:
  - **Encryption/decryption** (provide confidentiality)
  - **Digital signatures** (provide authentication)
  - **Key exchange** (of session keys)
- Some algorithms are suitable for all uses, others are specific to one

Public Key Algorithms

- ❑ Diffie-Hellman key exchange algorithm
  - only allows exchange of a secret key
- ❑ RSA (Rivest, Shamir, Adleman)
  - developed in 1977
  - only widely accepted public-key encryption algorithm
  - given tech advances, need 1024+ bit keys
- ❑ Digital Signature Standard (DSS)
  - provides only a digital signature function with SHA-1
- ❑ Elliptic curve cryptography (ECC)
  - new, security like RSA, but with much smaller keys

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Public Key Cryptography

- Public-Key algorithms rely on three principles:
  - It is computationally easy to generate key pairs
  - It is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - It is computationally infeasible to find private key knowing only algorithm & public key
  - Either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms)

$$C = f_k(P) \qquad easy\ if\ k\ and\ P\ are\ known$$
$$P = f_k^{-1}(C) \qquad easy\ if\ k\ and\ C\ are\ known$$
$$P = f_k^{-1}(C) \qquad infeasible\ if\ C\ is\ known\ but\ k\ is\ unknown$$

Security of Public Key Cryptography

- Like private key schemes brute force **exhaustive search** attack is always theoretically possible
- But keys used are too large (>512bits)
- Security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalysis) problems
- More generally the **hard** problem is known, but is made hard enough to be impractical to break
- Requires the use of **very large numbers**
- Hence is **slow** compared to private key schemes

Mathematical Background

$Remainder = x \ mod \ y$

1. Calculate floor(x/y) → $\left\lfloor \frac{x}{y} \right\rfloor$
2. Multiply $\left\lfloor \frac{x}{y} \right\rfloor \times y$
3. Remainder = $x - \left(\left\lfloor \frac{x}{y} \right\rfloor \times y \right)$

Ex: 53 mod 19

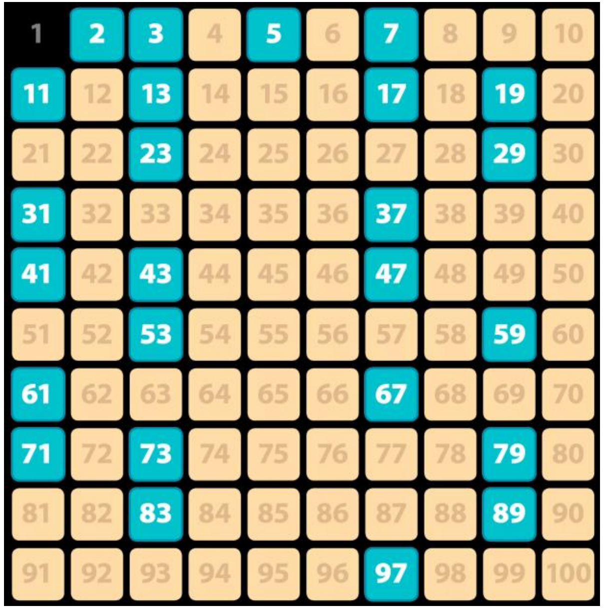
1. $\left\lfloor \frac{53}{19} \right\rfloor = 2$
2. $2 \times 19 = 38$
3. $Remainder = 53 - 38 = 15$

GCD and Multiplicative Inverse

- The greatest common divisor (gcd) between two numbers is the largest integer that will divide both numbers
  - $\gcd(4,10) = 2$
- If two numbers have a $\gcd$ of 1, then the smaller of the two numbers has a multiplicative inverse in the modulo of the larger number
  - $\gcd(4,9) = 1$ ➔ 4 has multiplicative inverse in $\boldsymbol{mod\ 9}$ ➔ 7
  - $4 \times 7 = 28 = 1\ mod\ 9$ ➔ $28 - 1 = 27$, which is dividable by 9

Prime Numbers

- A prime is a number that can only be divided without a remainder by itself and 1
- For any prime number p, every number from 1 up to $p-1$ has a $\gcd$ of 1 with p
- Therefore every number from 1 up to $p-1$ has a multiplicative inverse in $mod\ p$



Euler's Totient

- The number of elements that have a multiplicative inverse in a set of modulo integers
  - Also the number of elements with GCD = 1 with the integer
  - denoted using the Greek symbol phi ϕ
- For a prime number p, $\phi(p) = p - 1$
  - $\phi(7) = 6$

n	$\phi(n)$
1	1
2	1
3	2
4	2
5	4
6	2
7	6
8	4
9	6
10	4
11	10

RSA

- By Rivest, Shamir & Adleman of MIT in 1977
- Best known & widely used public-key scheme
- Based on exponentiation over integers modulo a prime
  - N.B. exponentiation takes $O((\log n)^3)$ operations (easy)
- Uses large integers (e.g. 1024 bits)
- Security due to cost of factoring large numbers
  - N.B. factorization takes $O(e^{\log n \log \log n})$ operations (hard)

RSA Key Setup

Each user generates a public/private key pair by:

1. Selecting two large primes at random: p, q
2. Computing their system modulus $n = p \cdot q$
   - Note that Euler's Totient $\phi(n) = (p-1) \cdot (q-1)$
3. Selecting at random the encryption key e
   - where $3 \leq e < \phi(n), \quad \gcd(e, \phi(n)) = 1$
4. Solve the following equation to find decryption key d
   - $e \times d \equiv 1\, mod\, \phi(n) and\ d < \phi(n) \quad [e \times d = 1 + k \times \phi(n) \quad for\ some\ k]$
5. Publish the public encryption key: $PU = \{e, n\}$
6. Keep secret the private decryption key: $PR = \{d, n\}$

RSA Use

- To encrypt a message M the sender:
  - obtains **public key** of recipient $PU = \{e, n\}$
  - computes: $C = M^e \bmod n$, where $0 \leq M < n$
- To decrypt the ciphertext C the owner:
  - uses their private key $PR = \{d, n\}$
  - computes: $M = C^d \bmod n$
- Note that the message M must be smaller than the modulus n (block if needed)