COMP412 Computer Security

Lec 07 Public Key Cryptography

Dr. Xiaochen Yuan
2021/2022

Contents

Birth of Public-Key Cryptosystems

- Beginning to 1960's: **permutations** and **substitutions** (Caesar, rotor machines, DES, . . .)
- 1960's: NSA secretly discovered public-key cryptography
- 1970: first known (secret) report on public-key cryptography by CESG, UK
- 1976: Diffie and Hellman public introduction to public-key cryptography
  - Avoid reliance on third-parties for key distribution
  - Allow digital signatures

Principles of Public-Key Cryptosystems

- Symmetric algorithms used same secret key for **encryption** and **decryption**
- Asymmetric algorithms in public-key cryptography use one key for **encryption** and different but related key for **decryption**
- Characteristics of asymmetric algorithms:
  - *Require*: Computationally infeasible to determine decryption key given only algorithm and encryption key
  - *Optional*: Either of two related keys can be used for encryption, with other used for decryption

Public and Private Keys

Public-Private Key Pair

- User A has pair of related keys, public and private: (PU_A, PR_A); similar for other users

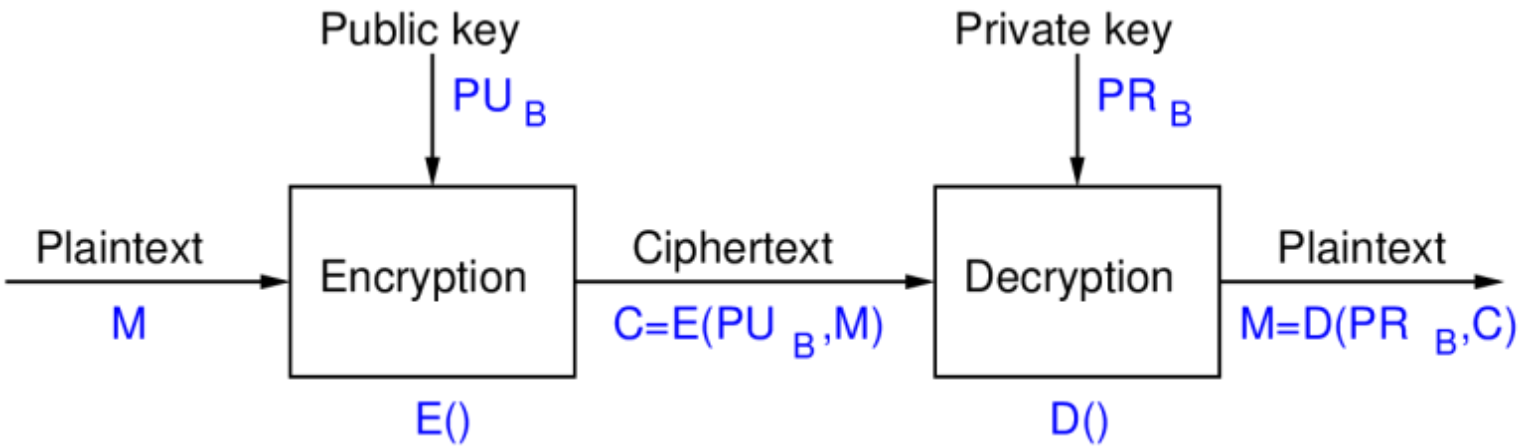
Public Key

- Public, Available to anyone
- For secrecy: used in **encryption**
- For authentication: used in **decryption**

Private Key

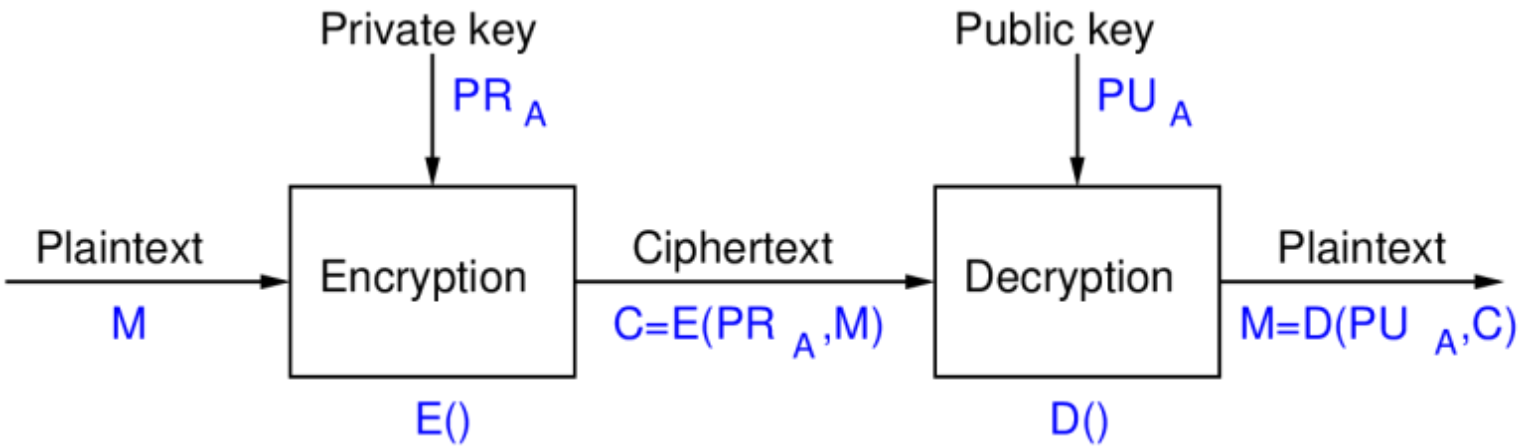
- Secret, known only by owner
- For secrecy: used in **decryption**
- For authentication: used in **encryption**

Confidentiality with Public Key Crypto



- Encrypt using **receivers** public key
- Decrypt using **receivers** private key
- Only the person with private key can successful decrypt

Authentication with Public Key Crypto



- Encrypt using **senders** private key
- Decrypt using **senders** public key
- Only the person with private key could have encrypted

Conventional vs Public-Key Encryption

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

Credit: Table 9.2 in Stallings, *Cryptography and Network Security*, 6th Ed.

Applications of Public Key Cryptosystems

- **Secrecy**, encryption/decryption of messages
- **Digital signature**, *sign* message with private key
- **Key exchange**, share secret session keys

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

Credit: Table 9.3 in Stallings, *Cryptography and Network Security*, 6th Ed., Pearson

Requirements of Public-Key Cryptography

1. Computationally easy for B to generate pair (PU_b, PR_b)
2. Computationally easy for A, knowing PU_b and message M, to generate ciphertext:

$$C = \mathrm{E}(PU_b, M)$$

3. Computationally easy for B to decrypt ciphertext using PR_b:

$$M = \mathrm{D}(PR_b, C) = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

4. Computationally infeasible for attacker, knowing PU_b and C, to determine PR_b
5. Computationally infeasible for attacker, knowing PU_b and C, to determine M
6. (Optional) Two keys can be applied in either order:

$$M = \mathrm{D}[PU_b, \mathrm{E}(PR_b, M)] = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

Requirements of Public-Key Cryptography

6 requirements lead to need for trap-door one-way function

- Every function value has **unique inverse**
- Calculation of function is **easy**
- Calculation of inverse is **infeasible**, unless certain information is known

$$
\begin{aligned}
Y &= f_k(X) && \textbf{easy}, \text{ if } k \text{ and } X \text{ are known} \\
X &= f_k^{-1}(Y) && \textbf{easy}, \text{ if } k \text{ and } Y \text{ are known} \\
X &= f_k^{-1}(Y) && \textbf{infeasible}, \text{ if } Y \text{ is known but } k \text{ is not}
\end{aligned}
$$

- What is **easy**? What is **infeasible**?
  - Computational complexity of algorithm gives an indication
  - Easy if can be solved in polynomial time as function of input

Public-Key Cryptanalysis

Brute Force Attacks

- Use large key to avoid brute force attacks
- Public-key algorithms less efficient with larger keys
- Public-key cryptography mainly used for **key management** and **signatures**

Compute Private Key from Public Key

- No known feasible methods using standard computing

Probable-Message Attack

- Encrypt all possible M^t using PU_b — for the C^t that matches C, attacker knows M
- Only feasible of M is short
- Solution for short messages: append random bits to make it longer

Contents

RSA

- Ron Rivest, Adi Shamir and Len Adleman
- Created in 1978; RSA Security sells related products
- Most widely used **public-key algorithm**
- Block cipher: plaintext and ciphertext are integers

The RSA Algorithm

Key Generation

1. Choose primes p and q *(private, chosen)*

and calculate $n = pq$ *(public, calculated)*

2. Select e: $gcd(\varphi(n), e) = 1$, $1 < e < \varphi(n)$

(public, chosen)

3. Find $d \equiv e^{-1} \pmod{\varphi(n)}$

(private, calculated)

$PU = \{e, n\}, PR = \{d, n\}, p$ and q also private

The RSA Algorithm

Encryption

Encryption of plaintext M, where $M < n$:

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

Decryption

Decryption of ciphertext C :

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

Requirements of the RSA Algorithm

1. Possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M < n$
2. Easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$
3. Infeasible to determine d given e and n

- Requirement 1 met if e and d are ***relatively prime***
- Choose primes p and q, and calculate:

$$n = pq$$
$$1 < e < \varphi(n)$$
$$ed \equiv 1 \pmod{\varphi(n)} \text{ or } d \equiv e^{-1} \pmod{\varphi(n)}$$

- $\boldsymbol{n}$ and $\boldsymbol{e}$ are public;
- $\boldsymbol{p}$, $\boldsymbol{q}$ and $\boldsymbol{d}$ are private

Example of RSA Algorithm

User selects p = 17, q = 11,

Generate the public & private keys: PU & PR.

Encrypts Plaintext: M = 88

Ciphertext C = ? (Confidentiality)

Decrypt the Ciphertext C.

Computational Efficiency of RSA

- Encryption and decryption require exponentiation
  - Very large numbers; using properties of modular arithmetic makes it easier:

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

- Choosing e
  - Values such as 3, 17 and 65537 are popular: make exponentiation faster
  - Small e vulnerable to attack: add random padding to each M
- Choosing d
  - Small d vulnerable to attack
  - Decryption using large d made faster using **Chinese Remainder Theorem** and **Fermat's Theorem**
- Choosing p and q
  - p and q must be very large primes
  - Choose random odd number and test if it's prime (probabilistic test)

Security of RSA

- **Brute-Force attack**: choose large d (but makes algorithm slower)
- **Mathematical attacks**:
  1. Factor n into its two prime factors
  2. Determine $\varphi(n)$ directly, without determining p or q
  3. Determine d directly, without determining $\varphi(n)$

  - Factoring n is considered fastest approach; hence used as measure of RSA security
- **Timing attacks**: practical, but countermeasures easy to add (e.g. random delay). 2 to 10% performance penalty
- **Chosen ciphertext attack**: countermeasure is to use padding (Optimal Asymmetric Encryption Padding)

Chinese Remainder Theorem

- 中国剩余定理, 孙子定理
  - 3rd-century book中国南北朝时期 -《孙子算经》
  - 有物不知其数，三三数之剩二，五五数之剩三，七七数之剩二。问物几何？
  - There are certain things whose number is unknown. If we count them by threes, we have two left over; by fives, we have three left over; and by sevens, two are left over. How many things are there?
  - 一个整数除以三余二，除以五余三，除以七余二，求这个整数?

Chinese Remainder Theorem

$$(S): \begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

$$M = m_1 \times m_2 \times \cdots \times m_n = \prod_{i=1}^{n} m_i$$

$$M_i = M/m_i, \quad \forall i \in \{1, 2, \cdots, n\}$$

$$t_i M_i \equiv 1 \pmod{m_i}, \quad \forall i \in \{1, 2, \cdots, n\}$$

$$x = a_1 t_1 M_1 + a_2 t_2 M_2 + \cdots + a_n t_n M_n + kM = kM + \sum_{i=1}^{n} a_i t_i M_i$$

Chinese Remainder Theorem

- 中国剩余定理, 孙子定理
  - 宋朝数学家秦九韶于1247年《数书九章》卷一、二《大衍类》对“物不知数”问题做出了完整系统的解答。明朝数学家程大位将解法编成易于上口的《孙子歌诀》：

    三人同行七十希，五树梅花廿一支，
    七子团圆正半月，除百零五使得知
  - 模数为3、5、7时候的同余方程的**秦九韶解法:** 将除以3得到的余数乘以70，将除以5得到的余数乘以21，将除以7得到的余数乘以15，全部加起来后减去105或者105的整数倍，得到的余数就是答案。

$$70 \times 2 + 21 \times 3 + 15 \times 2 = 233 = 2 \times 105 + 23.$$

  - The result was later generalized with a complete solution called Dayanshu (大衍術) in Qin Jiushao's 1247 Mathematical Treatise in Nine Sections (數書九章)

Contents

Public Key Cryptography - Procedure

Scenario:

- Alice wants to send an encrypted message to Bob

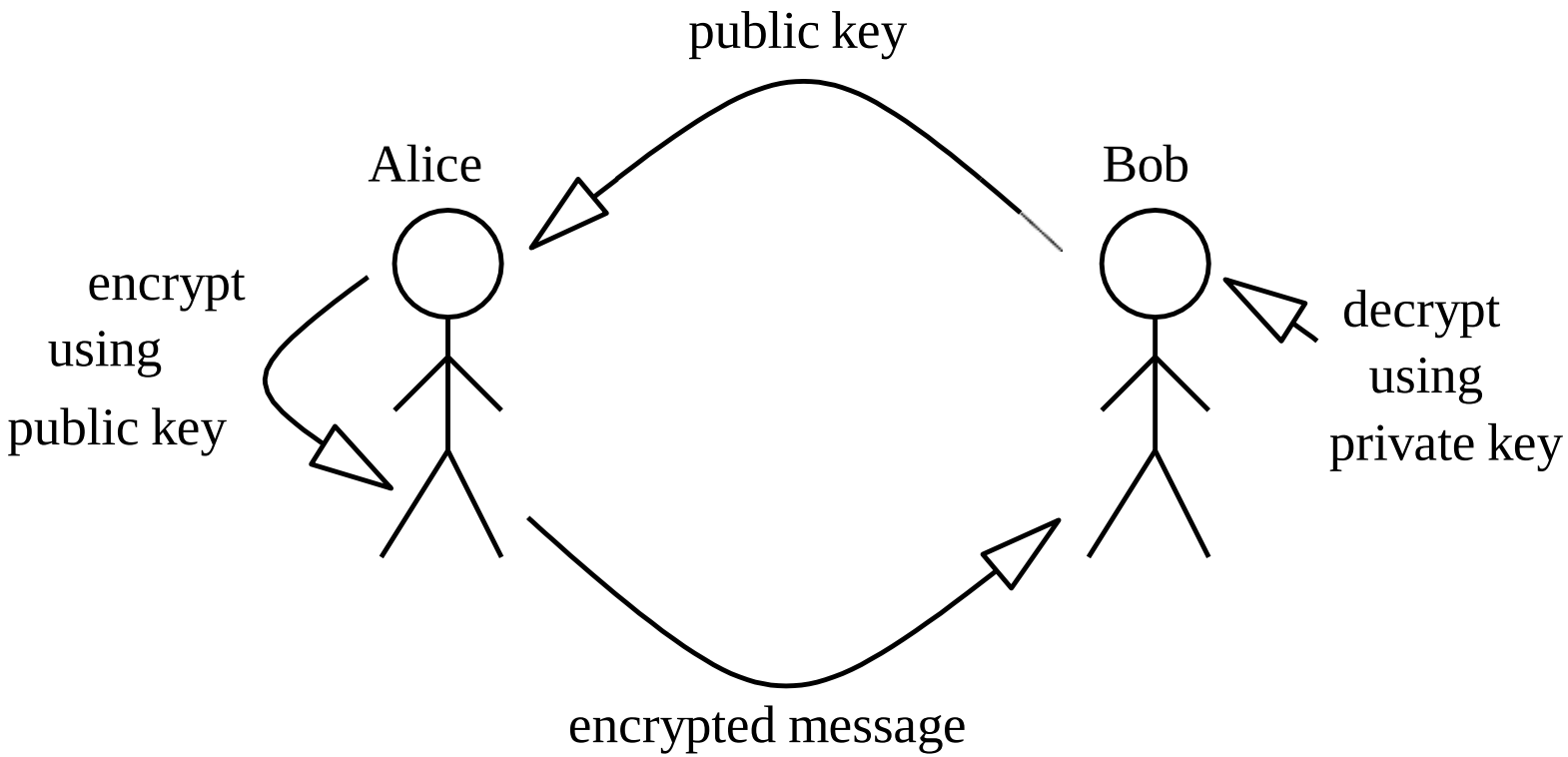
Procedure

1. Bob computes a public and a private key, the *keypair*
2. Bob publishes his public key
3. Alice encrypts the message using Bob's public key
4. Alice sends the message to Bob.
5. Bob decrypts the message using his private key

Effect:

- Nobody intercepting the message can read nor alter it unrecognized

Public Key Cryptography - Scheme



Diffie-Hellman Key Exchange

- Diffie and Hellman proposed public key crypto-system in 1976
- Algorithm for **exchanging** secret key (not for secrecy of data)
- Based on discrete logarithms
- Easy to calculate exponential modulo a prime
- Infeasible to calculate inverse, i.e. discrete logarithm

Diffie-Hellman Key Exchange Algorithm

Global Public Elements

q	prime number
α	$\alpha < q$ and α 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	$Y_B = \alpha^{X_B} \bmod q$

Calculation of Secret Key by User A

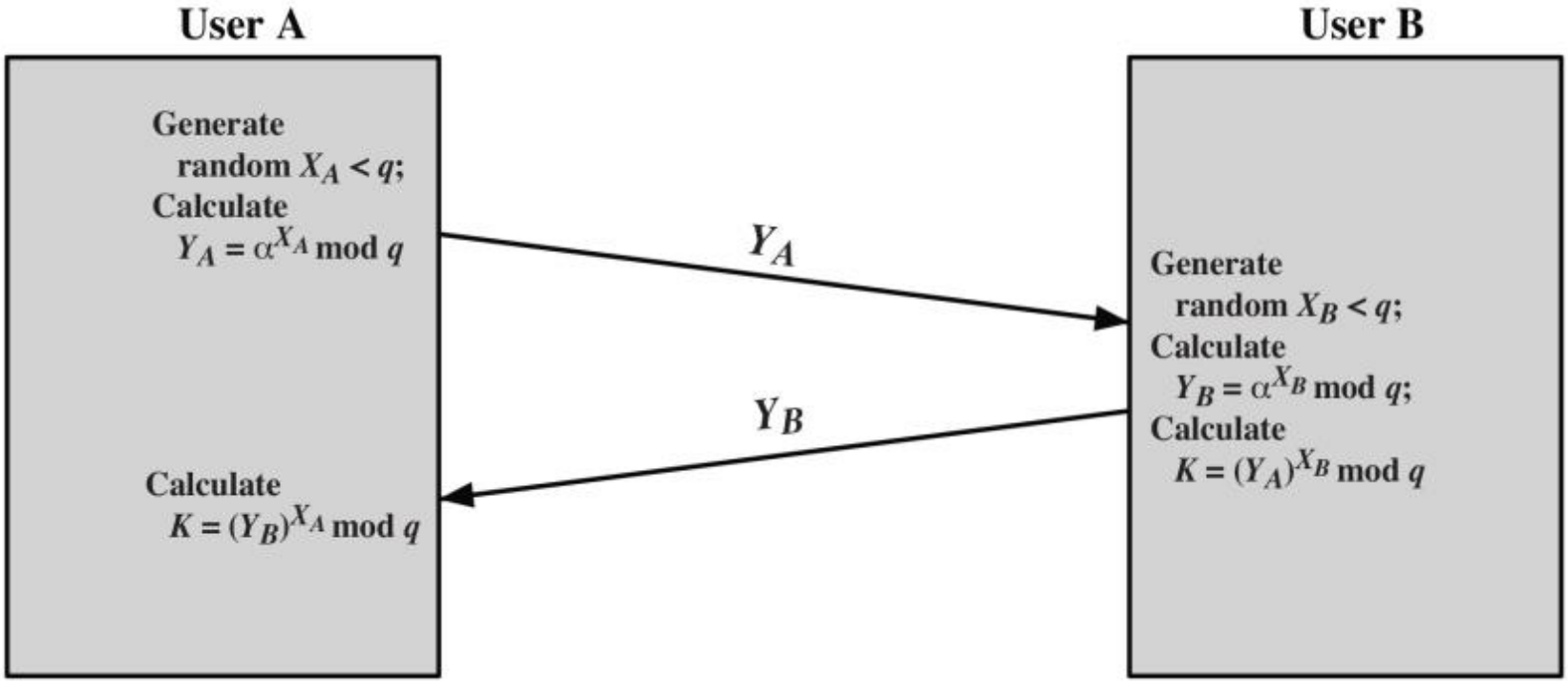
$K = (Y_B)^{X_A} \bmod q$

Calculation of Secret Key by User B

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

Credit: Figure 10.1 in Stallings, *Cryptography and Network Security*, 6th Ed., Pearson

Diffie-Hellman Key Exchange



Credit: Figure 10.2.2 in Stallings, *Cryptography and Network Security*, 6th Ed., Pearson

Diffie-Hellman Key Exchange Example

- Choose a prime number **q = 353**.
- Choose **α** a primitive root of q, **α = 3**.
  - α and q are *public*, known to both A and B (and anyone else, including attacker).

- A selects $\mathbf{X_A = 97}$ (*private*)
- B selects $\mathbf{X_B = 233}$ (*private*)

- Apply the ***Diffie-Hellman Key Exchange Algorithm*** to generate the private key.

Security of Diffie-Hellman Key Exchange

- Insecure against ***man-in-the-middle-attack***
- Countermeasure is to use digital signatures and public-key certificates

Man-in-the-middle-attack example

q=19, α = 3

User A: $\mathbf{X_A = 10}$

$\mathbf{Y_A = ?\ K_A = ?}$

User B: $\mathbf{X_B = 11}$

$\mathbf{Y_B = ?\ K_B = ?}$

Mal: $\mathbf{X_{MalA} = 2,\ Y_{MalA} = ?}$

$\mathbf{X_{MalB} = 7,\ Y_{MalB} = ?}$

After Man-in-the-middle-attack,

$\mathbf{K'_A = ?\ K'_B = ?}$

$\mathbf{K_{MalA} = ?\ K_{MalB} = ?}$

Contents

Other Public-Key Cryptosystems

ElGamal Crypto-system

- Similar concepts to Diffie-Hellman
- Used in Digital Signature Standard and secure email

Elliptic Curve Cryptography

- Uses elliptic curve arithmetic (instead of modular arithmetic in RSA)
- Equivalent security to RSA with smaller keys (better performance)
- Used for key exchange and digital signatures
- Not required in this course

ElGamal Cryptosystem

Presented in 1984 by Tather Elgamal

Key aspects:

- Based on the Discrete Logarithm problem
- Randomized encryption

Application:

- Establishing a secure channel for key sharing
- Encrypting messages

ElGamal Cryptosystem - Key Generation

Participant A generates the public/private key pair

1. Generate large prime q and generator g of the multiplicative group of the integers modulo q.
2. Select a random integer a, $1 \leq a < q - 1$, and compute $g^a \bmod q$.
3. A's Public key is $(q, g, g^a \bmod q)$;

   A's Private key is a.

ElGamal Cryptosystem - Encryption Procedure

Participant B encrypts a message m to A

1. Obtain A's authentic public key (q, g, g^a).
2. Represent the message as integers m in the range $\{0, 1, \ldots, q-1\}$.
3. Select a random integer k, $1 \leq k \leq q-1$.
4. Compute $\gamma = g^k \bmod q$ and $\delta = (m\ (g^a)^k) \bmod q$.
5. Send ciphertext $c = (\gamma, \delta)$ to A

ElGamal Cryptosystem - Decryption Procedure

Participant A receives encrypted message m from B

1. Use private key a to compute (γ^{q-1-a}) mod q.
   Note: $\gamma^{q-1-a} = \gamma^{-a} \pmod{q}$
2. Recover m by computing $((\gamma^{-a}) * \delta)$mod q.

ElGamal Cryptosystem - Example

Encryption:

Alice choses her public key:

- Prime $q = 17$
- Generator $g = 6$
- Given private key $a = 5$
- Public key ?

Bob encrypts his message $m = 13$:

- Given random $k = 10$
- *ciphertext* c = ?

Bob sends *ciphertext* to Alice.

ElGamal Cryptosystem - Example

Decryption:

Alice receives $ciphertext$ from Bob.

- Alice's public key is (q, g, g^a) =
- Her private key is $a = 5$

Alice now decrypts the message using her private key:

m' = $((\gamma^{-a})\ \delta)$ mod q

ElGamal Cryptosystem - Summary

Features:

- use of a random factor k for encryption
- variant of DH: shared secret is g^{ak}

Problems:

- Duplicates message length
- Depends on intractability of DL and DH