Network Security

Asim Rasheed

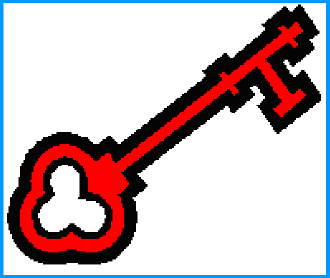
Where we are …

- Introduction to network security
- Vulnerabilities in IP
- I. CRYPTOGRAPHY
  - Symmetric Encryption and Message Confidentiality
  - Public-Key Cryptography and Message Authentication
- II. NETWORK SECURITY APPLICATIONS
  - Authentication Applications (Kerberos, X.509)
  - Electronic Mail Security (PGP, S/MIME)
  - IP Security (IPSec, AH, ESP, IKE)
  - Web Security (SSL, TLS, SET)
- III. SYSTEM SECURITY
  - Intruders and intrusion detection
  - Malicious Software (viruses)
  - Firewalls and trusted systems

Public Key Cryptography and RSA

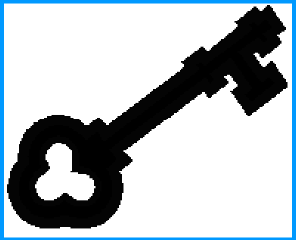
TWO TYPES OF KEYS

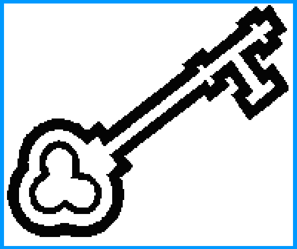
Symmetric Keys



Shared Secret Key
Session Key

Asymmetric Keys





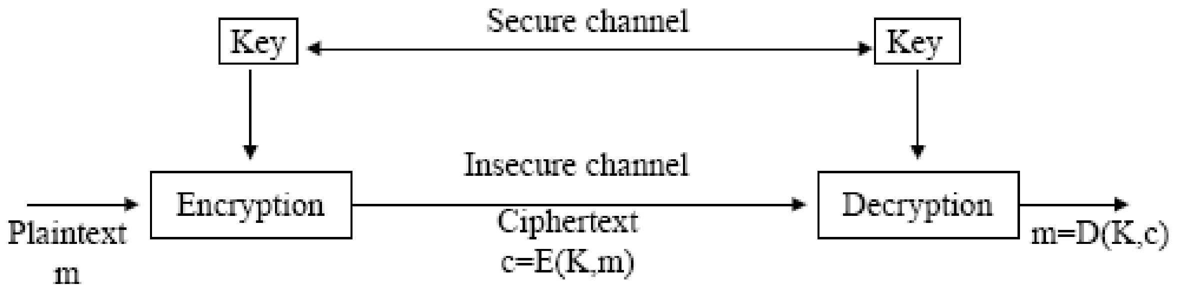
Private Key **Public Key**

Symmetric 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
- It is **symmetric**, parties are equal
- Does not protect sender
  - receiver may forge a message & claim that it is sent by the sender !!!

Symmetric Cryptosystems

- Symmetric encryption = Secret Key Encryption
- $K_E = K_D$ — secret (private) key
- Only sender S and receiver R know the key



- As long as the key remains secret, it also provides authentication (= proof of sender's identity)

Problems with Symmetric Encryption

- Ensuring security of the "**key channel**"
- Need an efficient key distribution infrastructure
  - **Separate keys needed for each communicating S-R pair**
- For **n** communicating users, need:

  n * (n -1) /2 keys

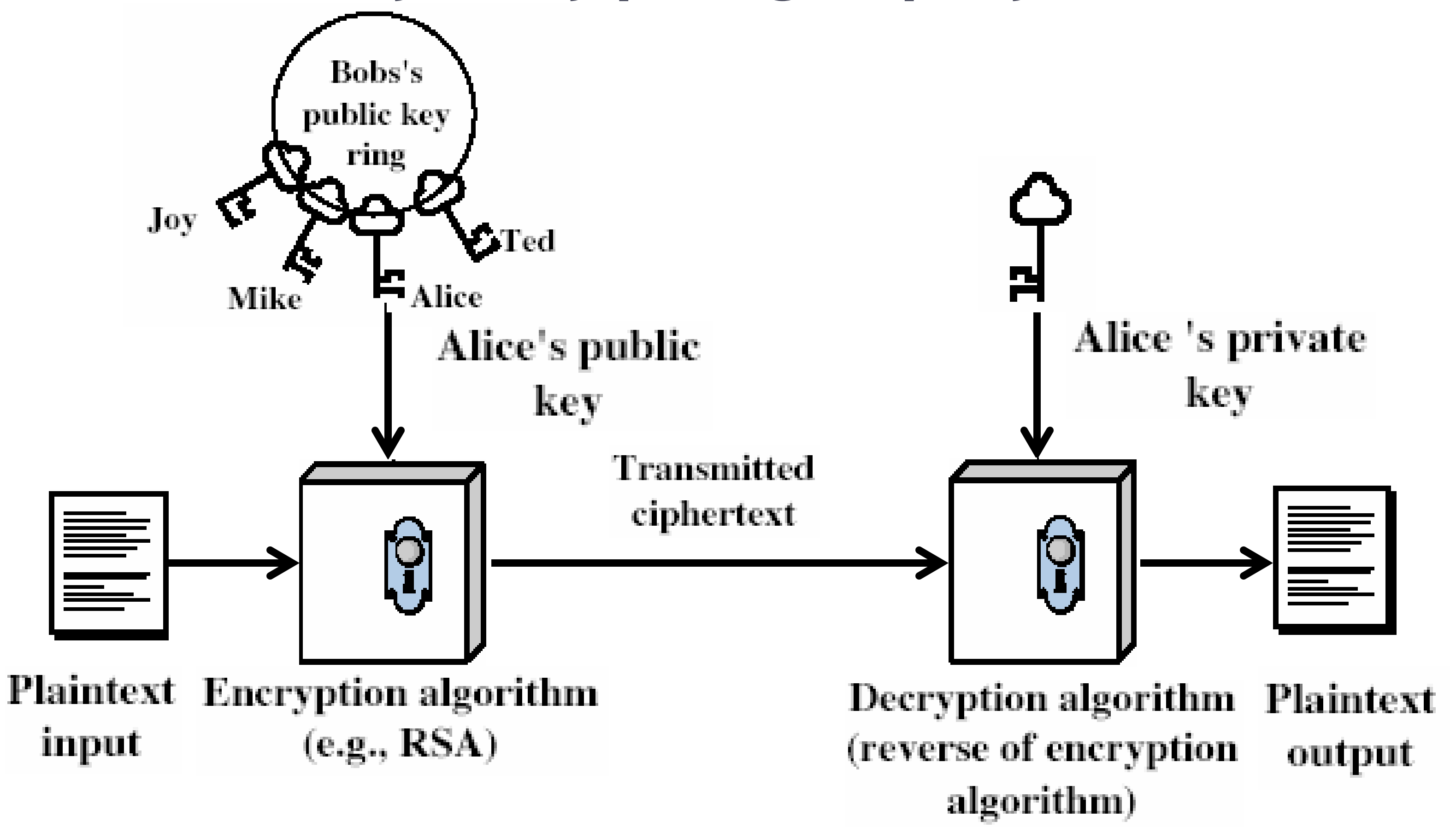
Public-Key Cryptography

- Whitfield Diffie & Martin Hellman at Stanford University in 1976 came up with a method that addressed both problems
- Probably most significant advancement in the 3000 years history of cryptography
- Uses clever application of number theory concepts to function
- Complements **rather than** replaces private key cryptography

Public-Key Cryptography

- Public-key/Two-key/Asymmetric cryptography involves the use of two keys:
  - A public-key, which may be known to 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:
  - Key used to encrypt messages or verify signatures can not be used to decrypt messages or create signatures

Public-Key Cryptography



Public-Key Cryptography

- Asymmetric encryption = Public Key Encryption (PKE)
- $K_E \neq K_D$ — public and private keys

- PKE systems eliminate symmetric encryption problems
  - Need no secure key distribution channel

    => easy key distribution

Why Public-Key Cryptography?

- Developed to address two key issues:
  - Key distribution: two important considerations were
    - Two communicants must already share a key
    - The use of Key Distribution Center (KDC): what if KDC is compromised
  - Digital Signatures: How can we know that the digital message is sent by the claimed sender??
    - we have signatures, used in paper documents

Asymmetric Cryptosystems

- One PKE approach:
  - R keeps her private key K_D
  - R can distribute the correspoding public key K_E to anybody who wants to send encrypted msgs to her
    - No need for secure channel to send K_E
    - Can even post the key on an open Web site — it is public!
- Only private K_D can decode msgs encoded with public K_E!
  - Anybody (K_E is public) can encode
  - Only owner of K_D can decode

Symm vs Asymm Keys

Symmetric	Asymmetric
•Key: D = E	•Key pair: <E, D>, D ≠ E
•K kept secret	•D kept secret E public (or known to n users)
•K agreed upon between 2 parties in advance	•E distributed to k users before first communication
•Like using a "simple" safe (with one door)	•Like using a safe with locked deposit slot
–Need safe key to deposit doc in safe	–Need *deposit slot key* to slide doc into safe
–Need safe key to get doc from safe	–Need *safe door key* to get doc from safe

Need for Key Management

- Private key must be carefully managed in both SE and PKE (asymm.) cryptosystems
  - Storing / safeguarding / activating-deactivating

Keys can expire - e.g. to take a key away from a fired employee

- Public key must be carefully distributed in PKE systems

  => Key management is a major issue

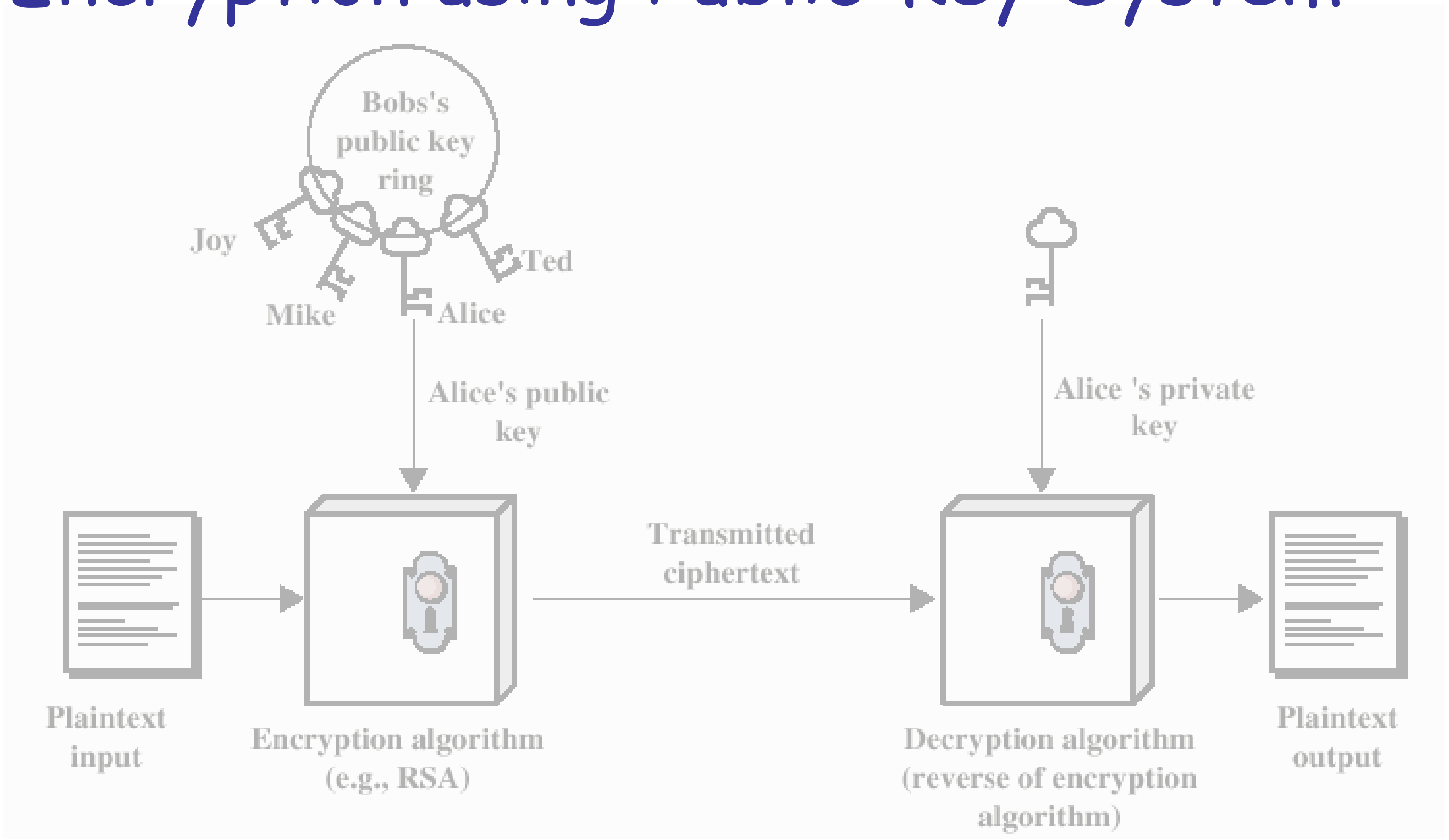
Public-Key Characteristics

- Public-Key algorithms rely on two keys with the characteristics that:
  - Computationally infeasible to find decryption key, knowing only algorithm & encryption key
  - Computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - Either of the two related keys can be used for encryption, with the other used for decryption

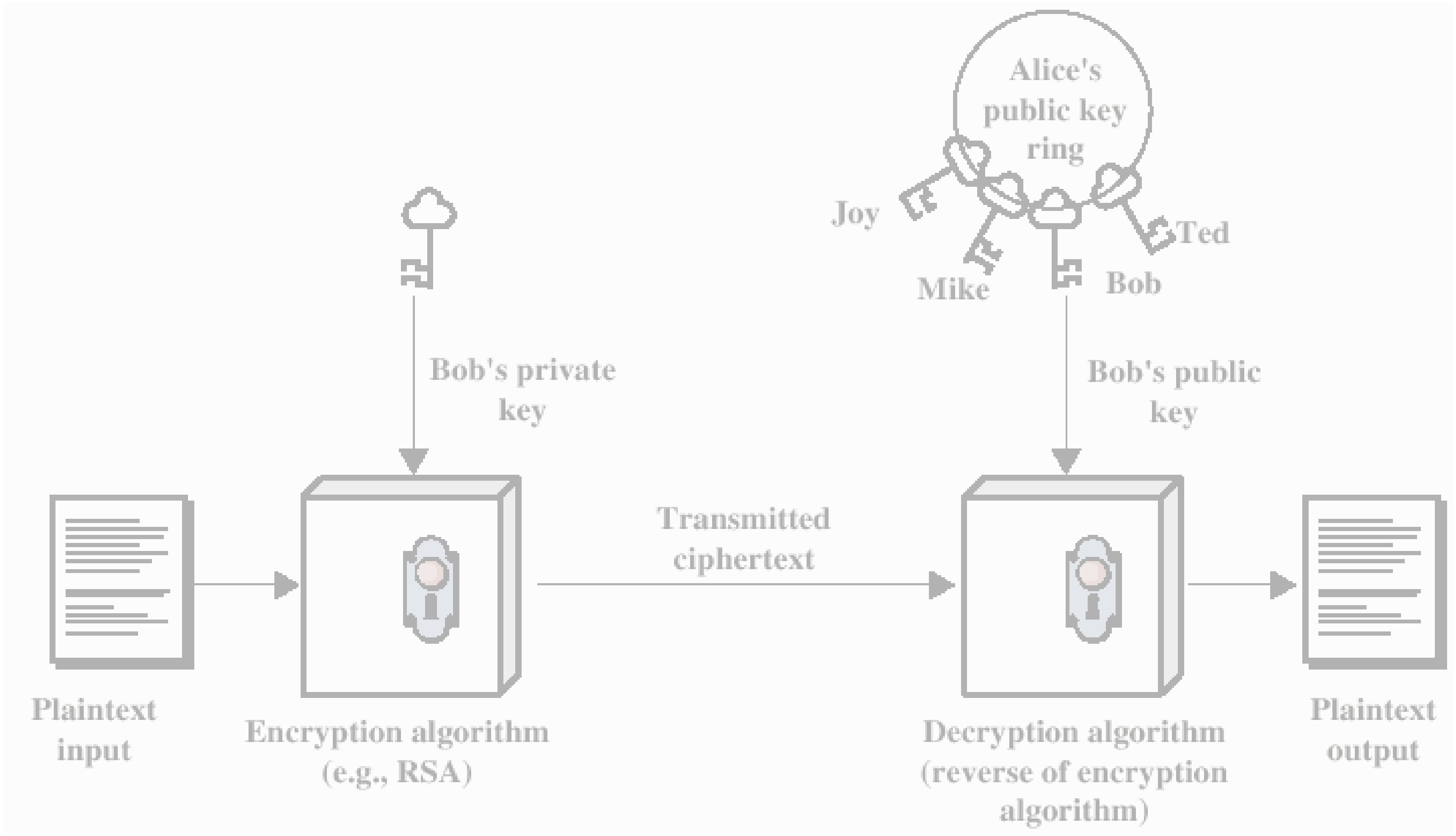
Public-Key Cryptosystems

- Each user generates a key pair
- Each user places one of the key, public key, in public register or other accessible file
- The companion key, private key, is kept secret
- If B wishes to send confidential message to A
  - Encrypt it using A's public key
- When A receives the message
  - It decrypts using its private key
  - No one else can decrypt the message

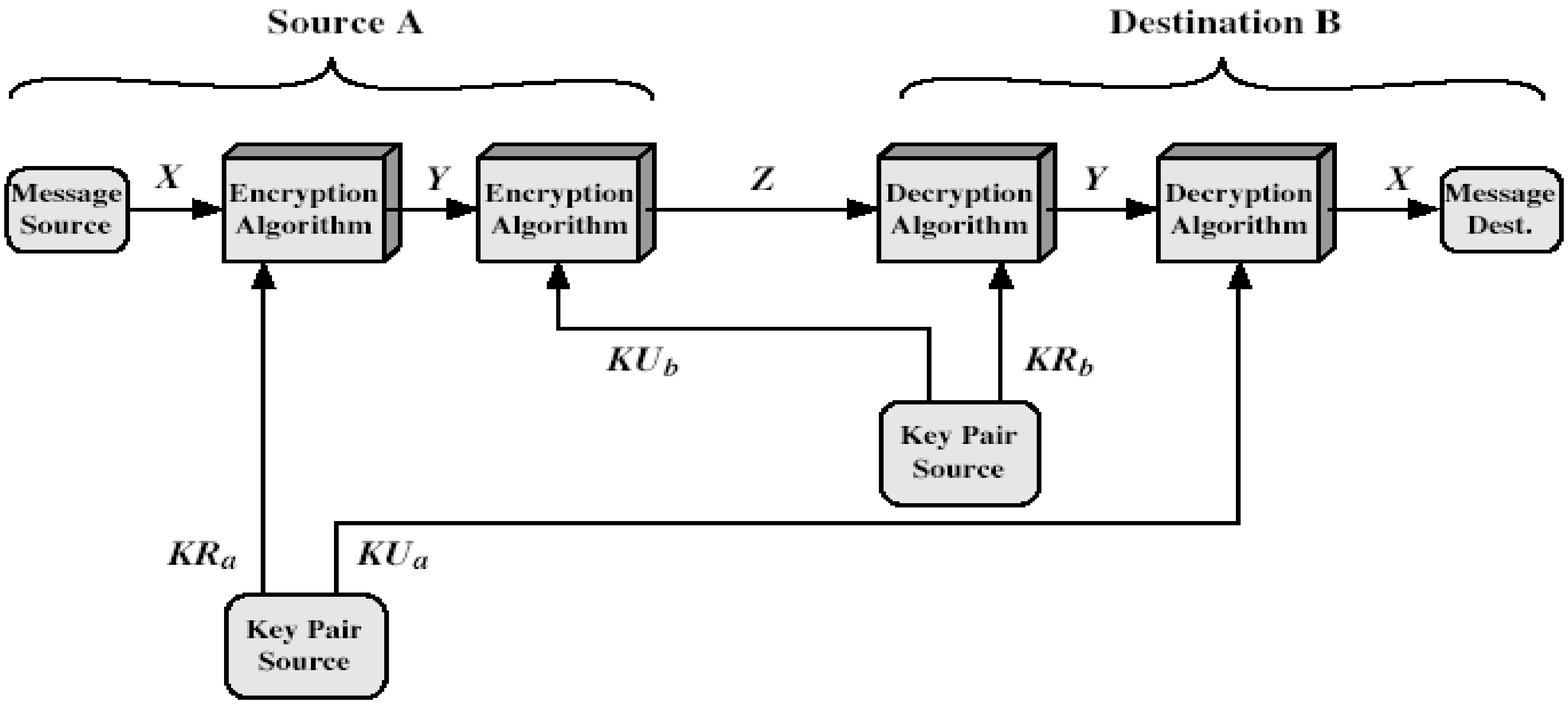
Encryption using Public-Key System



Authentication using Public-Key System



Public-Key Cryptosystems



Public key cryptosystems: Secrecy and Authentication

Public-Key Applications

- Can classify uses into 3 categories:
  - **encryption/decryption** (provide secrecy)
  - **digital signatures** (provide authentication)
  - **key exchange** (of session keys)
- Some algorithms are suitable for all uses, others are specific to one

Requirements for Public Key Cryptography

1.Computationally easy for party B to generate pair

2.Computationally easy for a sender, knowing public key and message, to generate corresponding ciphertext

3.Computationally easy for receiver B to decrypt the resulting ciphertext using the private key

Requirements…

1. Computationally infeasible for opponent, knowing the public key, to determine the private key

2. Computationally infeasible for an opponent, knowing the ciphertext and public key, to recover original message

3. Encryption and decryption functions can be applied in either order

Security of Public Key Schemes

- 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** (cryptanalyse) problems
- More generally the **hard** problem is known, its just made too hard to do in practise
- Requires the use of **very large numbers**
- Hence is **slow** compared to private key schemes

Public-Key Substitution Risk

•Molly can remove Bob’s public key and replace it with her own. Then Alice encrypts using “Bob’s” public key.

•Molly intercepts the message, decrypts it with her own private key, and modifies it.

•Molly re-encrypts it with Bob’s real public key. Bob can decrypt it with his private key, so he never detects the attack.

Forging Signatures

- Molly removes Alice's public key and replaces it with her own.

- Alice signs a message with her private key. Molly intercepts it, strips the signature, then modifies the message.

- Molly creates a new signature for the message using her own private key.

Forging Signatures

•Bob receives the signature and decrypts it with "Alice's" public key.

•Bob also runs the hash over Molly's bogus message and verifies the signature.

The Problem

- We need a way to tie a public/private key pair to a person

- A *digital signature* only ties a message to a *private key*, not to a person!

The Solution

- We need a trusted third party that can authoritatively bind a key pair to a person

- This trusted third party is called a “certification authority” (CA)

- The CA issues a *digital certificate* to each user, which contains the public key for that user

RSA

- By Rivest, Shamir & Adleman of MIT in 1977
- Best known & widely used public-key scheme
- Based on exponentiation in a finite (Galois) field over integers modulo a prime
- Uses large integers (e.g., 1024 bits)
- Security due to cost of factoring large numbers

RSA Key Setup

- Each user generates a public/private key pair by:
  - Choosing two large primes at random: p , q
  - Computing $N = p \times q$
  - Choose e (less than N) such that e and $(p-1)(q-1)$ are relatively prime
  - Choose d such that $(e \times d) \bmod [(p-1)(q-1)]$ is equal to 1

- Publish their public encryption key: $KU=\{N,e\}$
- Keep secret private decryption key: $KR=\{p,q,d\}$

RSA Use

- Sender uses the following algorithm to encrypt the message:
  - **Public key** of recipient KU={N,e}
  - Computes: $C = M^e \bmod N$, where $0 \leq M < N$
- Receiver uses the following algorithm to decrypt the ciphertext:
  - Uses their private key KR={p,q,d}
  - Computes: $M = C^d \bmod N$
- Note that the message M must be smaller than the modulus N (block if needed)

RSA Example

1. Select primes: $p=17$ & $q=7$
2. Compute $N = p \times q = 17 \times 7 = 119$
3. Compute $(p-1)(q-1) = 16 \times 6 = 96$

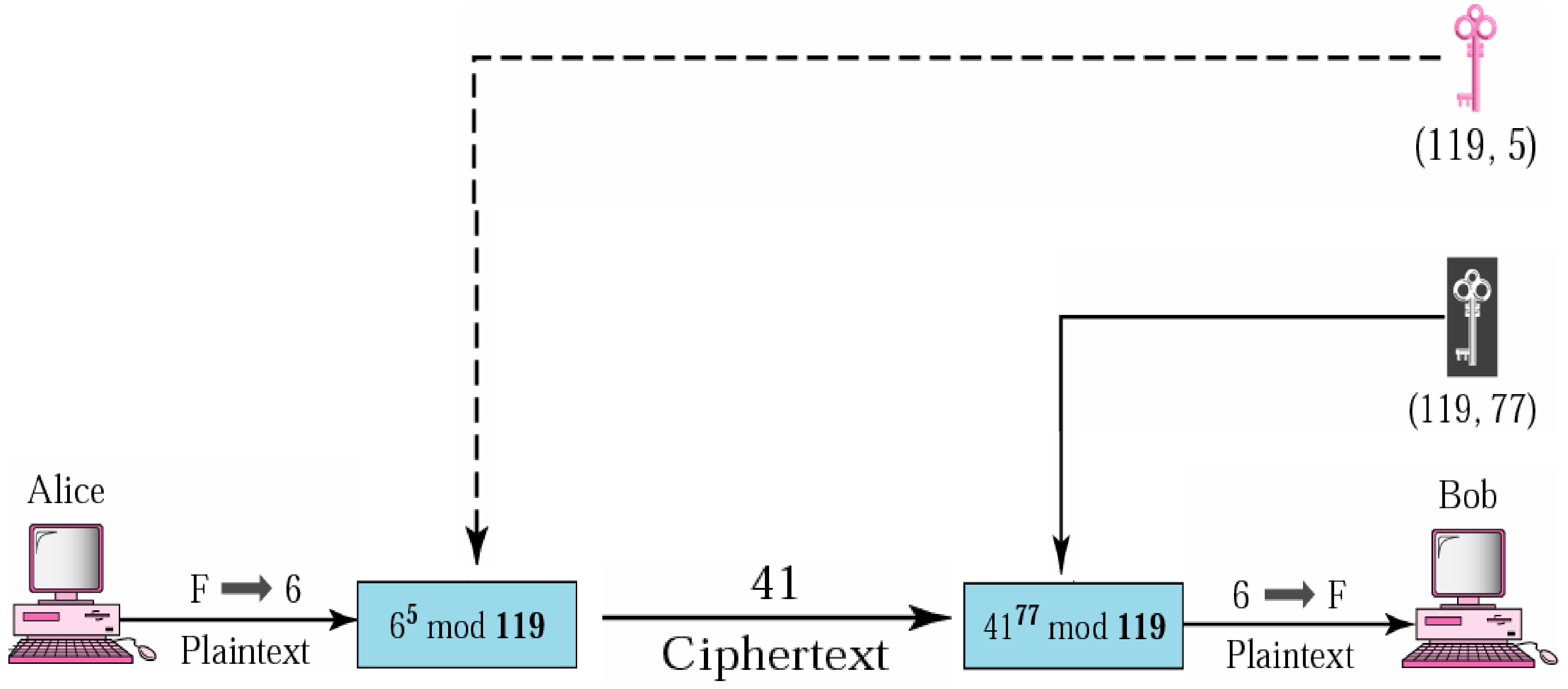
   Select e : $\gcd(e,96)=1$; choose $e=5$

1. Determine d: $d \times e = 1 \bmod 96$ and $d < 96$ Value of is $d=77$ since $77 \times 5 = 385 = 4 \times 96 + 1$

   Publish public key: $KU=\{119,5\}$

   Keep secret private key: $KR=\{17,7,77\}$

Example…



(119, 5)
(119, 77)
Alice
Bob
F ➡ 6
Plaintext
6^5 mod **119**
41
Ciphertext
41^{77} mod **119**
6 ➡ F
Plaintext

RSA Key Generation

- Users of RSA must:
  - Determine two primes at random: `p, q`
  - Select either `e` or `d` and compute the other
- Primes `p,q` must not be easily derived from modulus `N=p.q`
  - Means must be sufficiently large
  - Typically guess and use probabilistic test
- `e, d` are inverses, so use inverse algorithm to compute the other

RSA Security

- Three approaches to attack RSA:
  - Brute force key search (infeasible; given size of numbers)
  - Mathematical attacks
    - Factor N=p*q, hence find (p-1)(q-1) and then d
    - Determine (p-1)(q-1) directly and find d
    - Find d directly
  - Timing attacks (on running of decryption)

Timing Attacks

- Developed in mid-1990's
- Exploit timing variations in operations
  - e.g., multiplying by small vs large number
- Infer operand size based on time taken
- RSA exploits time taken in exponentiation
- Countermeasures
  - use constant exponentiation time
  - add random delays
  - blind values used in calculations

Summary

- Have considered:
  - principles of public-key cryptography
  - RSA algorithm, implementation, security

Any question ?

Key Management

Key Management

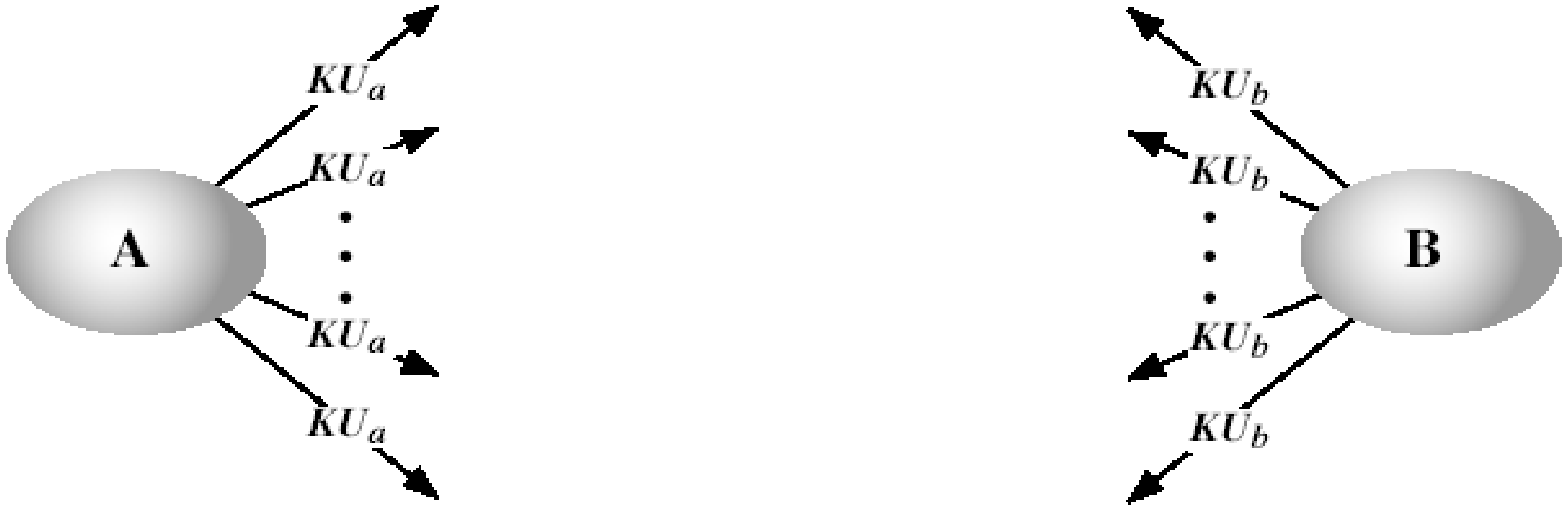
- Public-key encryption helps address key distribution problems
- Have two aspects of this:
  - Distribution of public keys
  - Use of public-key encryption to distribute secret keys

Distribution of Public Keys

- Can be considered as using one of following mechanisms:
  - Public announcement:
  - Publicly available directory
  - Public-key authority
  - Public-key certificates

Public Announcement

- Users distribute public keys to recipients or broadcast to community at large
  - e.g., append PGP keys to email messages or post to news groups or email list



Public Announcement Weakness

- Major weakness is forgery
  - –Anyone can create a key claiming to be someone else and broadcast it
  - –Until forgery is discovered, can masquerade as claimed user

Publicly Available Directory

•Can obtain greater security by registering keys with a public directory

•Directory must be trusted with properties:

–Contains {name, public-key} entries

–Participants register securely with directory

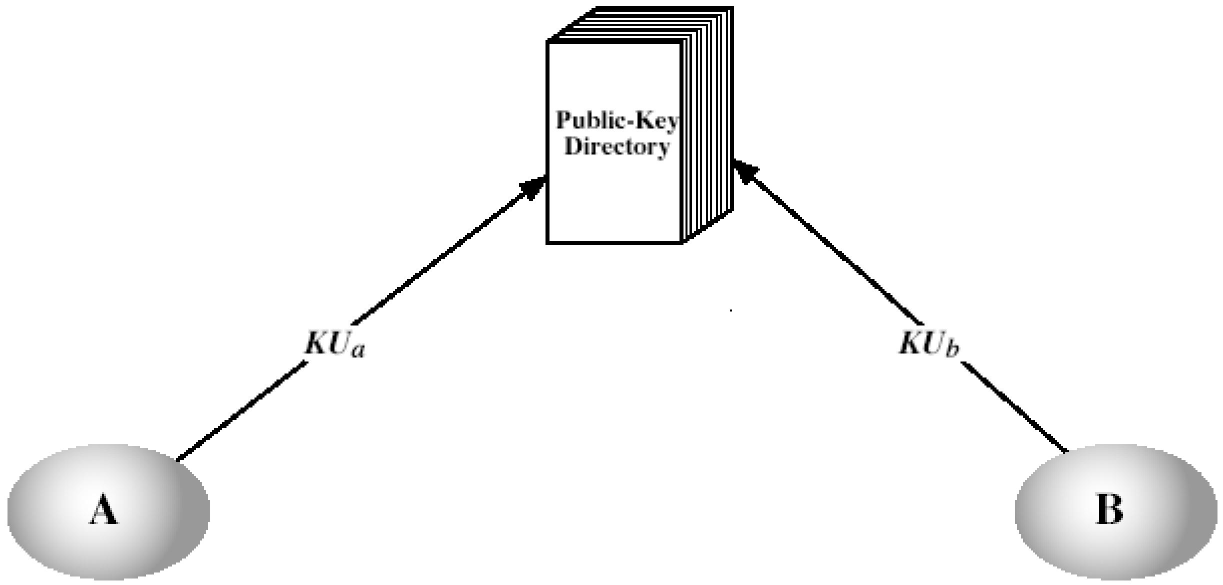
–Participants can replace key at any time

–Directory is periodically published

–Directory can be accessed electronically

Publicly Available Directory

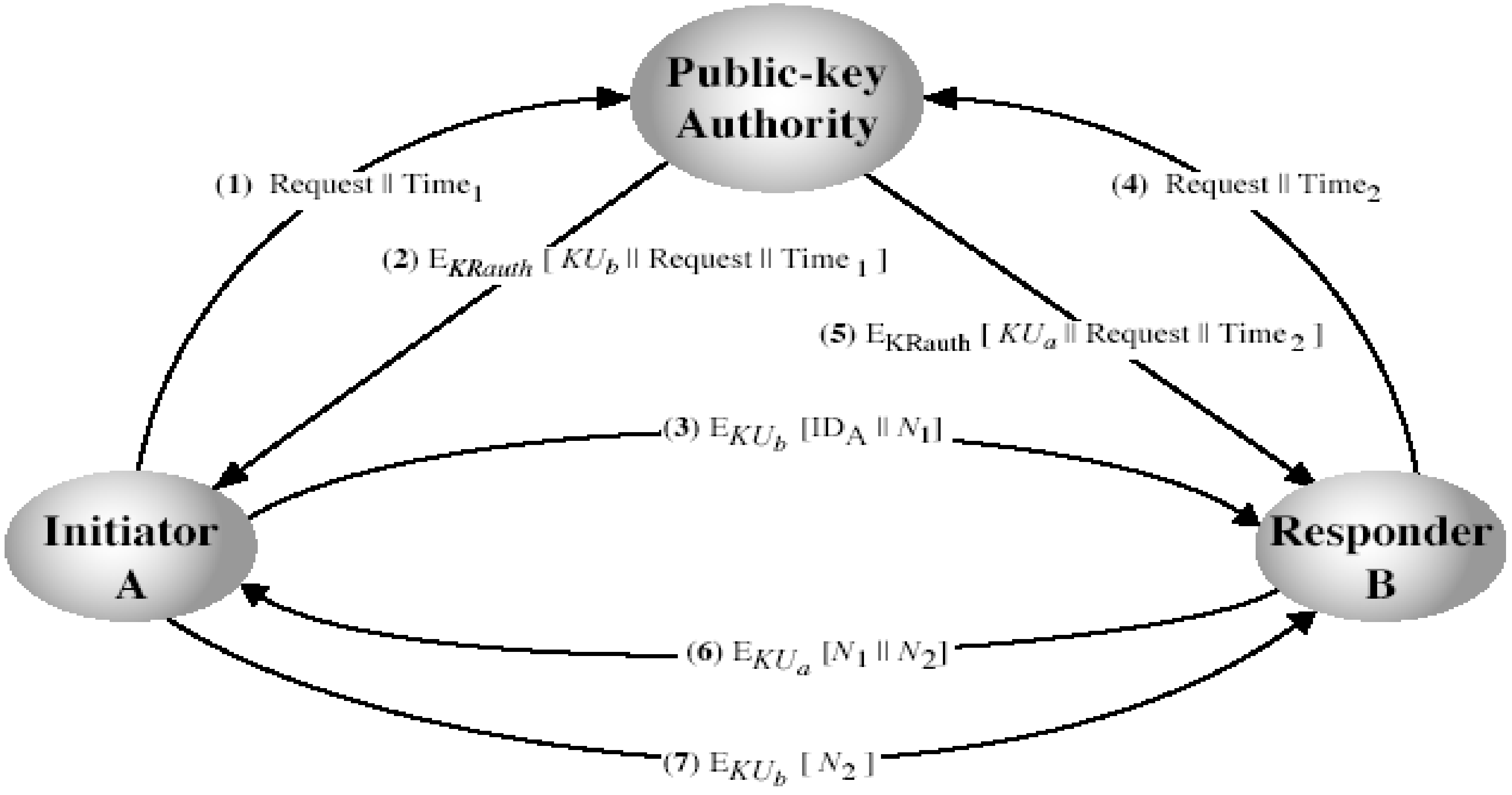
- More secure than individual public announcements
- Still vulnerable to tampering or forgery



Public-Key Authority

- Improve security by tightening control over distribution of keys from directory
- Has properties of directory
- And requires users to know public key for the directory
- Then users interact with directory to obtain any desired public key securely
  - Does require real-time access to directory when keys are needed

Public-Key Authority


Public-key Authority
Initiator A
Responder B
(1) Request ‖ $Time_1$
(2) E_{KRauth} [KU_b ‖ Request ‖ $Time_1$]
(3) E_{KU_b} [ID_A ‖ N_1]
(4) Request ‖ $Time_2$
(5) E_{KRauth} [KU_a ‖ Request ‖ $Time_2$]
(6) E_{KU_a} [N_1 ‖ N_2]
(7) E_{KU_b} [N_2]

Public-Key Authority Message Exchange

•A sends a timestamped message to the public key authority containing request for B's public key

•Authority responds with a message encrypted with its private key.

–A's assured message is generated from authority

–Message includes: B's public key, original request and original timestamp

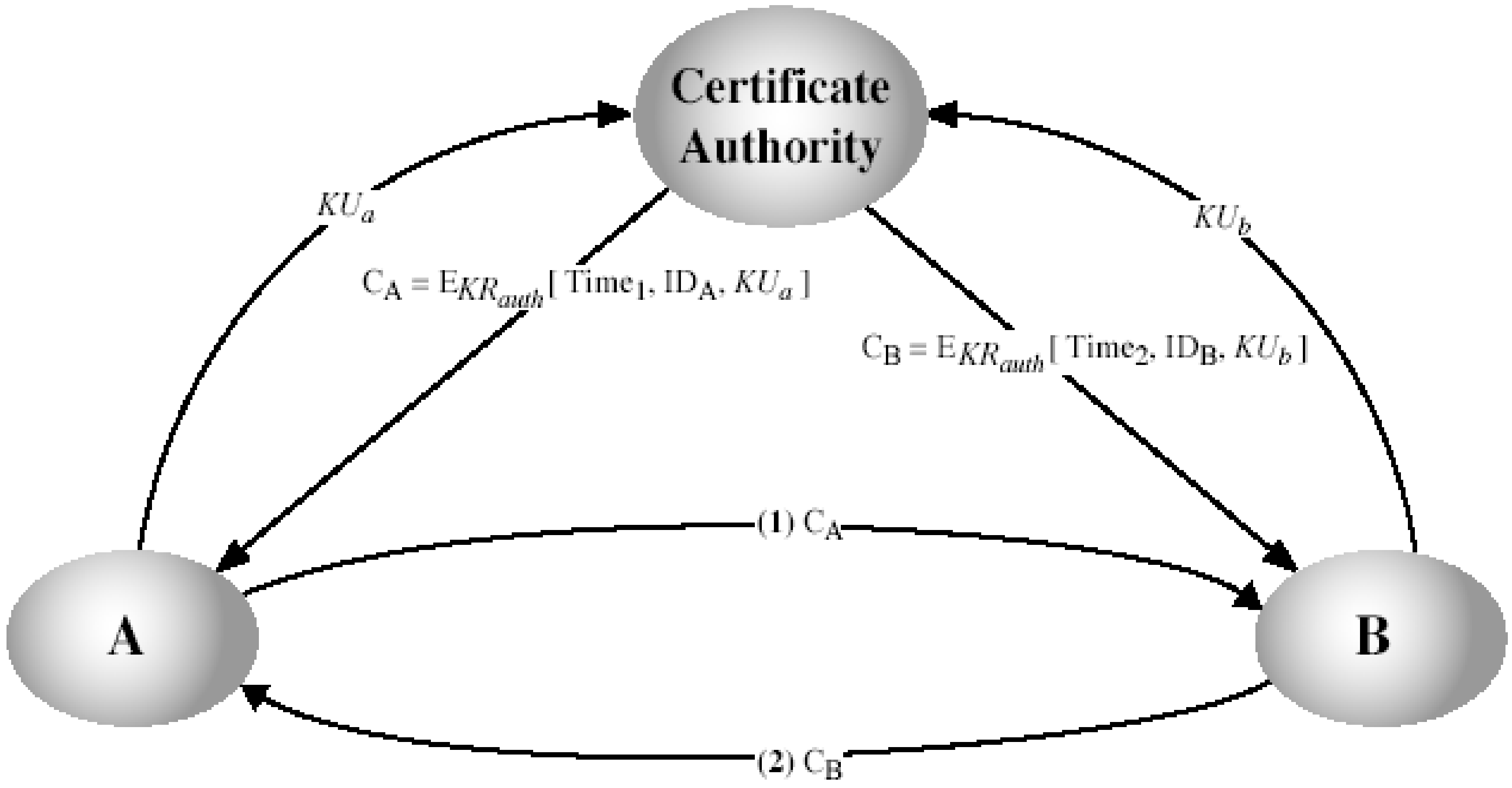
Message Exchange…

- A stores B's public key and also uses it for encryption of ID_A and a nonce
  - This identifies the session uniquely
- B gets the public key from authority
- B sends a message to A encrypted with A's public key
  - Message contains A's nonce & new nonce from B
  - A's nonce in message tells that the communicator is B
- A returns B's nonce, encrypted using B's public key

Public-Key Certificates

- Certificates allow key exchange without real-time access to public-key authority
- A certificate binds **identity** to **public key**
  - Usually with other information such as period of validity, rights of use, etc.
- With all contents **signed** by a trusted Public-Key or Certificate Authority (CA)
- Can be verified by anyone who knows the public-key authorities public-key

Public-Key Certificates



Certificate Authority
KU_a
KU_b
$C_A = E_{KR_{auth}}[\text{Time}_1, ID_A, KU_a]$
$C_B = E_{KR_{auth}}[\text{Time}_2, ID_B, KU_b]$
(1) C_A
(2) C_B
A
B

Public-Key Certificates

- Nodes acquire the certificates from the CA
- Each certificate contains a public key and other information regarding currency of the certificate, encrypted with CA's private key
  - Recipient uses CA's public key to decrypt the certificate
- Participant conveys its key information using this certificate

Public-Key Certificates Requirements

•Any participant can read a certificate

•Any participant can verify that the certificate originated from CA

•Only CA can create and update certificates

•Any participant can verify the currency of the certificate

Public Key Distribution of Secret Keys

•Use previous methods to obtain public-key

•Can use for secrecy or authentication

•But public-key algorithms are slow

•So usually want to use private-key encryption to protect message contents

–Hence need a session key

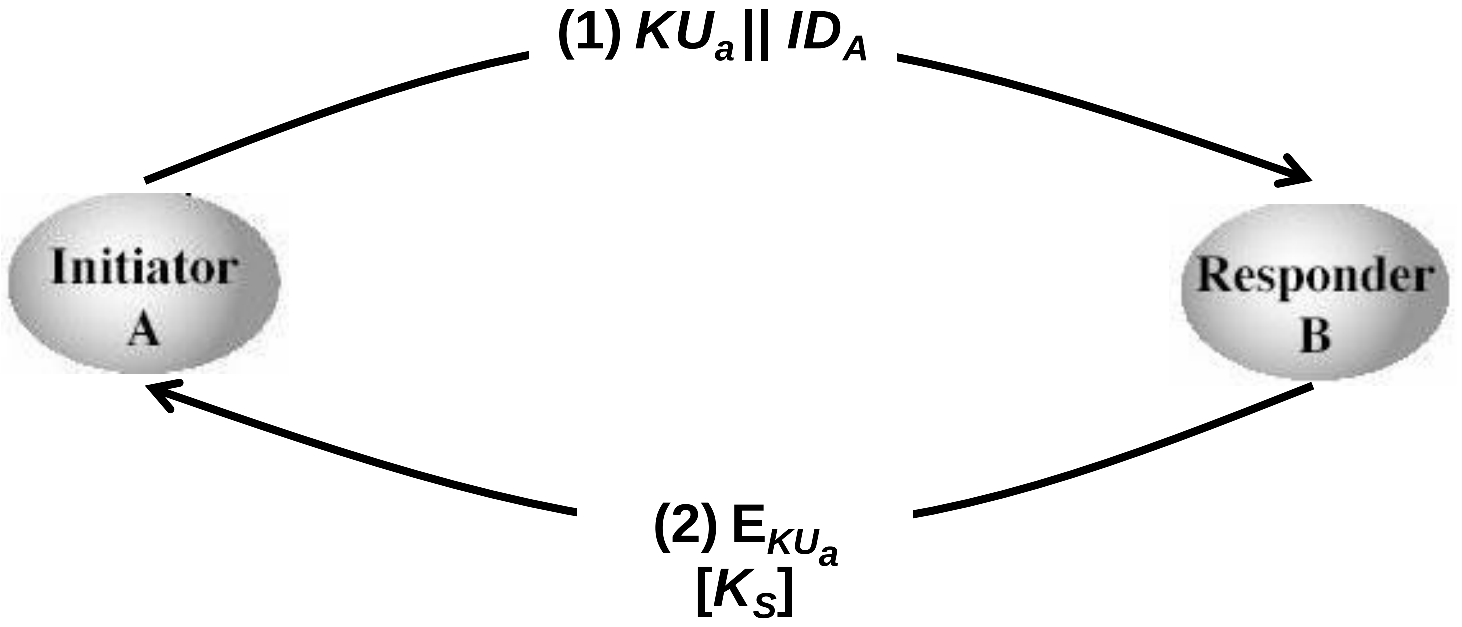
•Have several alternatives for negotiating a suitable session

Simple Secret Key Distribution

- Proposed by Merkle in 1979
  - A generates a new temporary public key pair
  - A sends B the public key and their identity
  - B generates a session key K sends it to A encrypted using the supplied public key
  - A decrypts the session key and both use

Secret Key Distribution

- Problem is that an opponent can intercept and impersonate both halves of protocol



Secret Key Distribution Problem Scenario

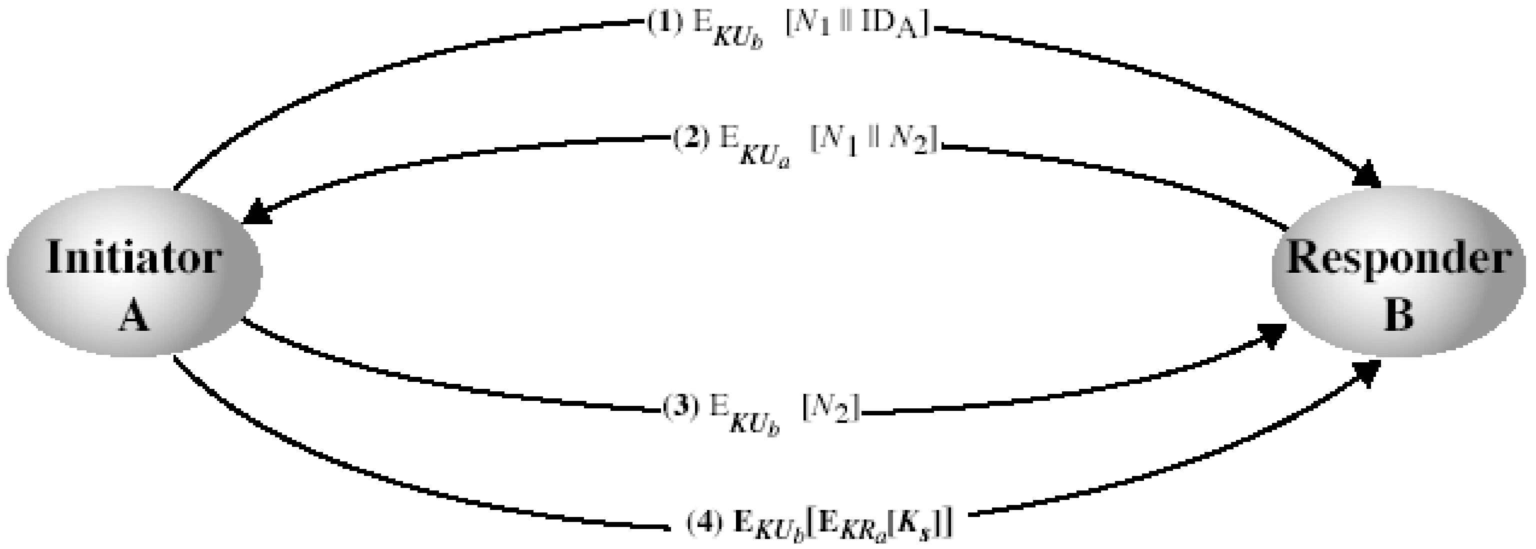
- A generates public/private key and transmits to B
- E can intercept a message, create its own public/private key pair and transmits it to B
- B generates a secret key and transmits it encrypted with E’s public key

Problem Scenario

- E intercepts message and learns the secret key
- E transmits to A the secret key encrypted with A’s public key
- Now E can intercept all the communication

Distribution of Secret Keys with Confidentiality and Authentication

- Must have securely exchanged public-keys:


(1) $E_{KU_b}[N_1 \| ID_A]$
(2) $E_{KU_a}[N_1 \| N_2]$
Initiator A
Responder B
(3) $E_{KU_b}[N_2]$
(4) $E_{KU_b}[E_{KR_a}[K_s]]$

Distribution of Secret Keys

1. A uses B's public key to encrypt ID_A and a nonce
-Used to identify the transaction
2. B sends a message to A encrypted with A' public key, containing A's nonce and B's nonce
3. A returns nonce of B
4. A selects a secret key and sends message M; first encrypting secret key with A private key and then encrypting with B's public key

Diffie-Hellman Key Exchange

- First public-key type scheme proposed
- By Diffie & Hellman in 1976 along with the exposition of public key concepts
  - Note: James Ellis (UK CESG) secretly proposed the concept in 1970
- Is a practical method for public exchange of a secret key
- Used in a number of commercial products

Diffie-Hellman Key Exchange

- A public-key distribution scheme
  - Cannot be used to exchange an arbitrary message
  - Rather it can establish a common key
  - Known only to the two participants
- Value of key depends on the participants (and their private and public key information)
- Based on exponentiation in a finite field (modulo a prime or a polynomial) - easy
- Security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

Diffie-Hellman Setup

- Both the communicating parties agree on p and g
- Alice picks S_A at random, Bob picks S_B
- Alice computes $T_A = g^{S_A} \bmod p$, Bob computes $T_B = g^{S_B} \bmod p$
- They exchange Ts
- Each raises the number they receive to their secret number
- Alice computes $T_B^{S_A} \bmod p$, Bob computes $T_A^{S_B} \bmod p$
- They come up with same number; because
- $T_B^{S_A} = (g^{S_B})^{S_A} = g^{S_A S_B} = (g^{S_A})^{S_B} = T_A^{S_B} \bmod p = K_{AB}$

Diffie-Hellman Example

- Users Alice & Bob who wish to swap keys:
- Agree on prime p=353 and g =3
- Select random secret keys:

– A chooses $S_A=97$, B chooses $S_B=233$

- Compute public keys:

-$T_A=3^{97} \bmod 353 = 40$ (Alice)

-$T_B=3^{233} \bmod 353 = 248$ (Bob)

- Compute shared session key as:

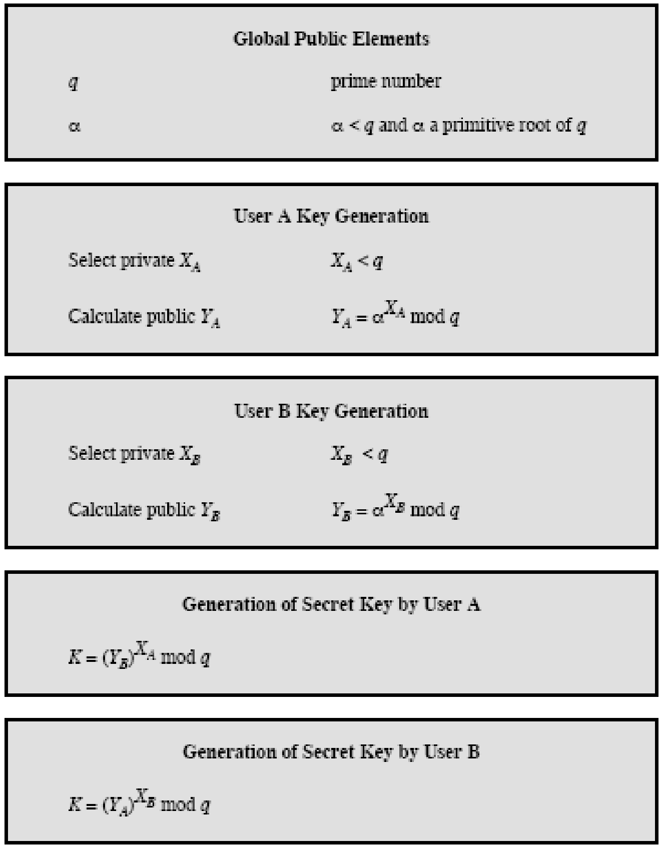
$K_{AB}= T_B^{S_A} \bmod 353 = 248^{97} = 160$ (Alice)

$K_{AB}= T_A^{S_B} \bmod 353 = 40^{233} = 160$ (Bob)

Diffie-Hellman Key Exchange

- K_{AB} is used as session key in private-key encryption scheme between Alice and Bob
- If Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- Attacker needs an S, must solve discrete log

The 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	$Y_B = \alpha^{X_B} \bmod q$

Generation of Secret Key by User A

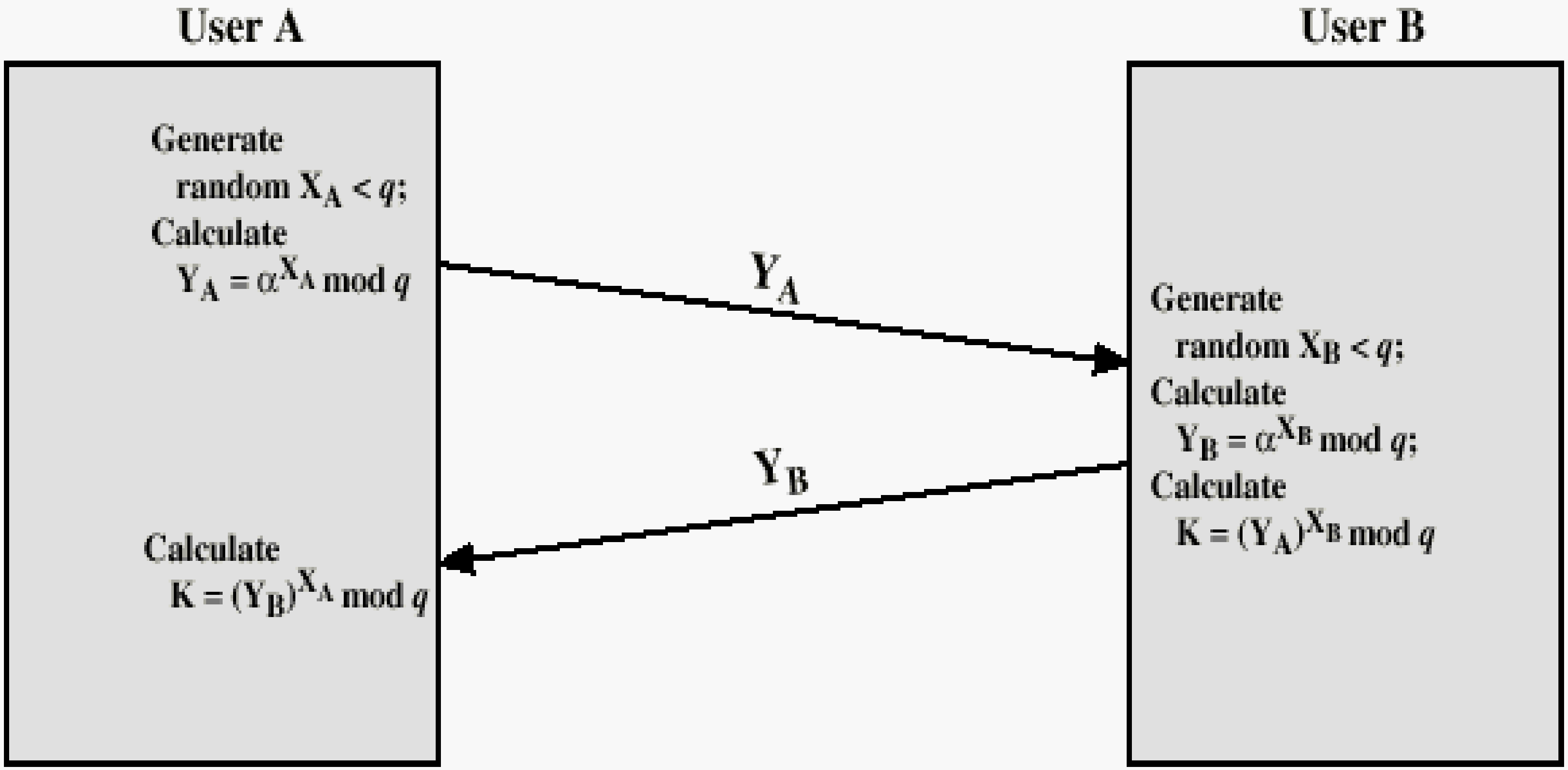
$K = (Y_B)^{X_A} \bmod q$

Generation of Secret Key by User B

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

Figure 10.7 The Diffie-Hellman Key Exchange Algorithm

Diffie-Hellman Key Exchange



User A
User B
Generate random $X_A < q$;
Calculate $Y_A = \alpha^{X_A} \bmod q$
Y_A
Generate random $X_B < q$;
Calculate $Y_B = \alpha^{X_B} \bmod q$;
Calculate $K = (Y_A)^{X_B} \bmod q$
Y_B
Calculate $K = (Y_B)^{X_A} \bmod q$

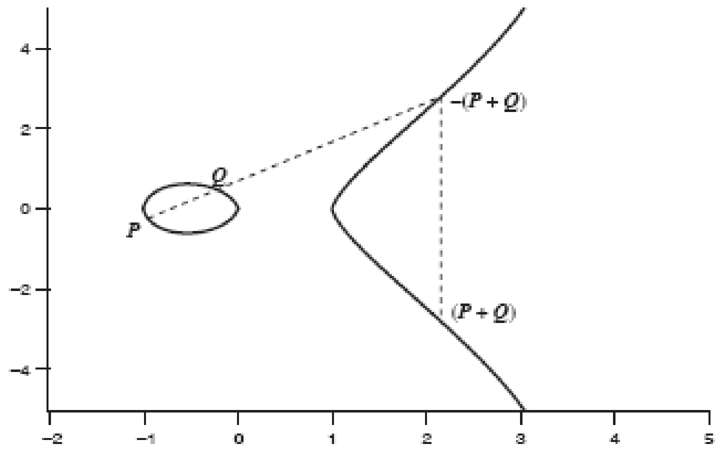
Elliptic Curve Cryptography

- Majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials
- Imposes a significant load in storing and processing keys and messages
- An alternative is to use elliptic curves
- Offers same security with smaller bit sizes

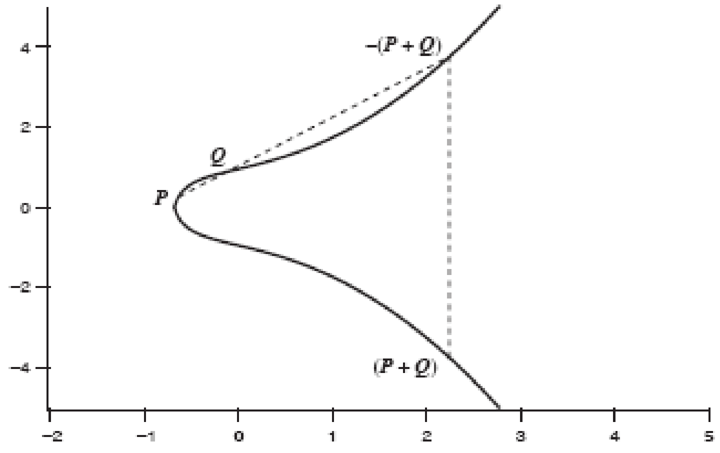
Real Elliptic Curves

- An elliptic curve is defined by an equation in two variables x & y, with coefficients
- Consider a cubic elliptic curve of form
  - $y^2 = x^3 + ax + b$
  - where x, y, a, b are all real numbers
  - also define zero point O
- Have addition operation for elliptic curve
  - Geometrically sum of Q+R is reflection of intersection R

Real Elliptic Curve Example



(a) $y^2 = x^3 - x$

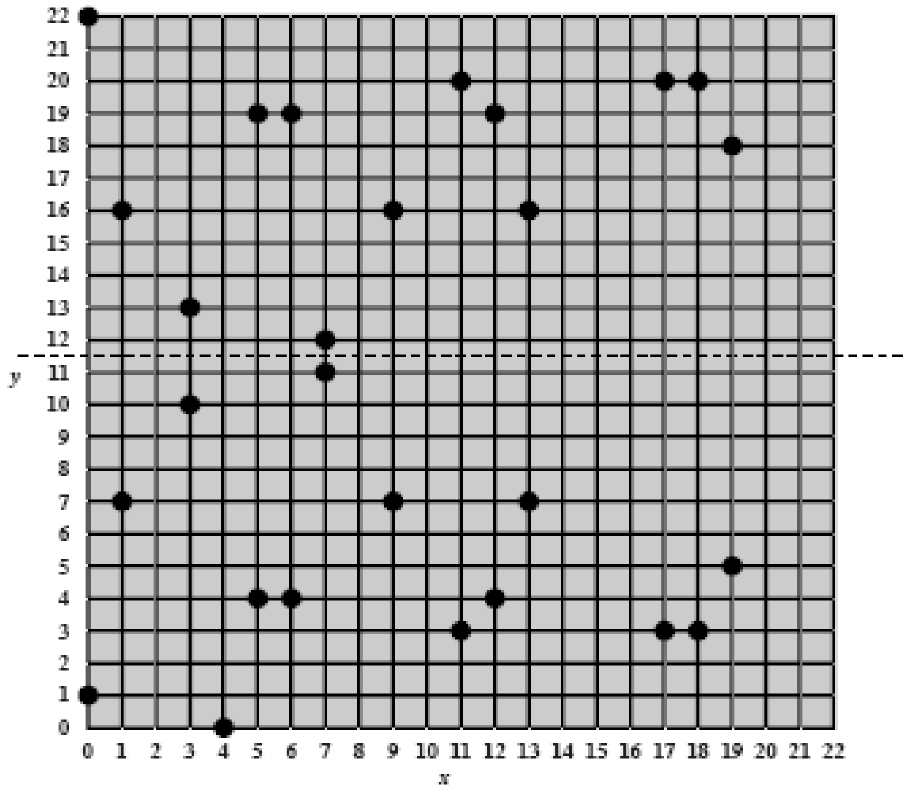


(b) $y^2 = x^3 + x + 1$

Finite Elliptic Curves

- Elliptic curve cryptography uses curves whose variables & coefficients are finite
- Have two families commonly used:
  - Prime curves $E_p(a,b)$ defined over Z_p
    - use integers modulo a prime
    - best in software
  - Binary curves $E_{2m}(a,b)$ defined over $GF(2^n)$
    - use polynomials with binary coefficients
    - best in hardware

The Elliptic Curve $E_{23}(1,1)$



Elliptic Curve Cryptography

•ECC addition is analog of modulo multiply

•ECC repeated addition is analog of modulo exponentiation

•Need "hard" problem equivalent to discrete log

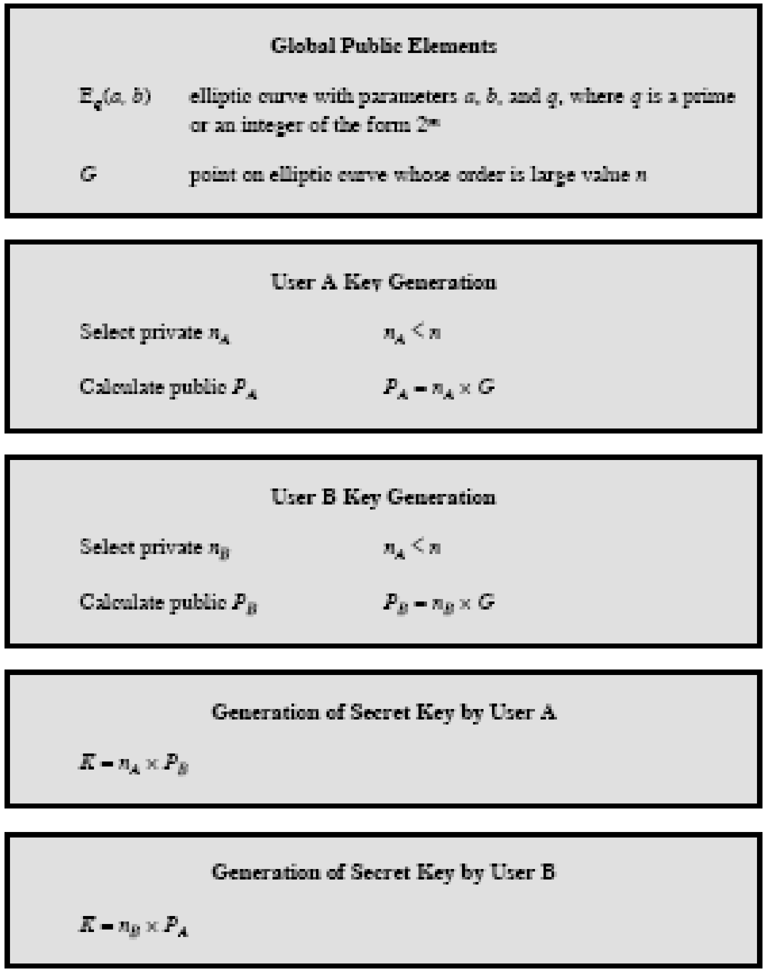
–`Q=kP`, where Q, P belong to a prime curve

–is "easy" to compute Q given k,P

–but "hard" to find k given Q, P

–known as the elliptic curve logarithm problem

ECC Key Exchange



Global Public Elements

$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation

Select private n_B	$n_A < n$
Calculate public P_B	$P_B = n_B \times G$

Generation of Secret Key by User A

$K = n_A \times P_B$

Generation of Secret Key by User B

$K = n_B \times P_A$

Summary

- Have considered:
  - Distribution of public keys
  - Public-key distribution of secret keys
  - Diffie-Hellman key exchange
  - Elliptic Curve cryptography

Any question ?