COMP2121: Principles of Distributed Systems and Networks

Security

Unit coordinator Dr. Vincent Gramoli
School of Information Technologies

THE UNIVERSITY OF SYDNEY

Introduction

- Previous lectures:
  - Byzantine or arbitrary failures are the hardest failures to deal with
  - Tolerance to byzantine failures is needed to have a system coping with malicious behaviors (e.g., hackers, viruses)

- Today's lecture:
  - Security is precisely the mean used by a system to be resistant against arbitrary failures
  - Let's focus on the mechanisms to guarantee integrity (no changes) and confidentiality (no interception) of the communication among distributed entities

Outline

- Definitions
- Basic Operations
- Secret Key System
- Public Key System
- Hash Function
- Authentication
- Integrity

Definitions

THE UNIVERSITY OF SYDNEY

Definitions

Goals

- *Confidentiality*: property of a computer system whereby its information is disclosed only to authorized parties
- *Integrity*: alterations of a system's assets (hardware, software, data) can be made only in an authorized way
- Security threats
  - *Interception*: unauthorized party has access to a service or data
    e.g., communication between two parties was overheard by someone else
  - *Interruption*: service of data becomes unavailable
    e.g., denial of service attack, where a service is made inaccessible to others
  - *Modification*: unauthorized changes of data or tampering with a service that no longer adheres to its specification
    e.g., changing a program so that it secretly logs activities of its user
  - *Fabrication*: generating additional data or activity that would normally not exist
    e.g., an intruder adds an entry into a password file or databases

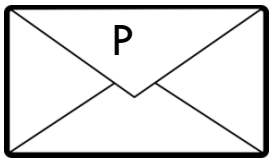
Definitions

Means

- *Security policy*: actions the entities of the system are allowed to take and which ones are prohibited
- Security mechanisms: to ensure the secure policy:
  - *Encryption*: action of transforming data into something an attacker cannot understand
  - *Authentication*: action of verifying the claimed identity of an entity (e.g., client, server, host, user, etc.)
  - *Authorization*: action of verifying whether the entity has the rights to perform the action it requests
  - *Auditing*: action of monitoring which entity access what and how

Definitions

Cryptography example

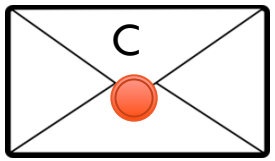


plaintext message P

- Bob wants to transmit message P protected from security threats to Alice:

Definitions

Cryptography example (con't)



sealed message C

- Bob wants to transmit message P protected from security threats to Alice:
  1. Bob encrypts P into an intelligible message C

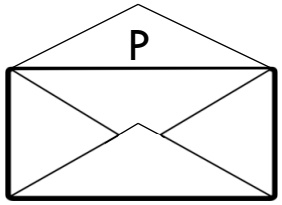
Definitions

Cryptography example (con't)

- Bob wants to transmit message P protected from security threats to Alice:
  1. Bob encrypts P into an intelligible message C
  2. Bob sends C to Alice

Definitions

Cryptography example (con't)

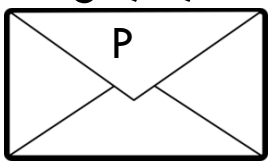


- Bob wants to transmit message P protected from security threats to Alice:
  1. Bob encrypts P into an intelligible message C
  2. Bob sends C to Alice
  3. Upon reception, Alice must decrypt C into its original form P

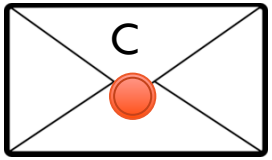
Definitions

Cryptography example (con't)

- The original form of the message, P, is called the *plaintext*



- The encrypted form, C, is called the *ciphertext*

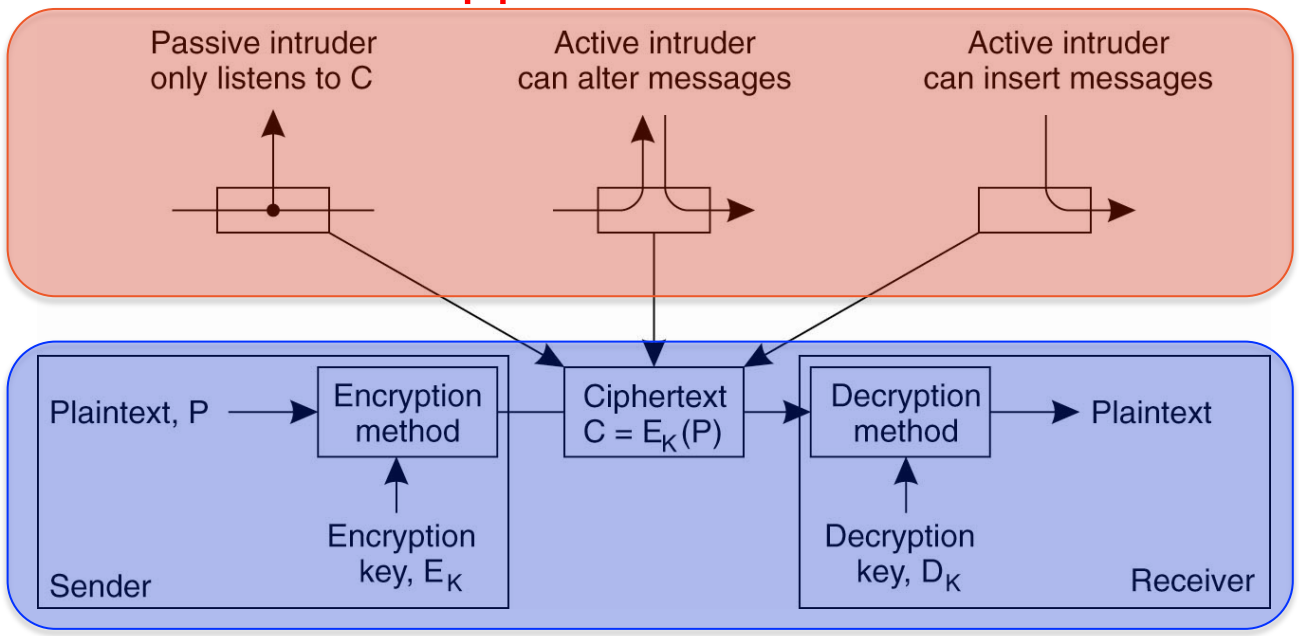


- $C = E_K(P)$: ciphertext C obtained by encrypting P using key K
- $P = D_K(C)$: plaintext P obtained by decrypting C using key K

Definitions

Cryptography example (con't)

- Intruders and eavesdroppers in communication



- Cryptography protects from interruption, modification and fabrication

Definitions

Keys

Cryptographic methods are parameterized by keys:

1. Secret key (a.k.a., symmetric) cryptosystem:
   - The same key K is used to both encrypt and decrypt the message

     $P = D_K(E_K(P))$
   - The key is kept secret, no one except the sender and receiver must know it
2. Public key (a.k.a., asymmetric) cryptosystem:
   - There are two separate keys, K_E for encryption and K_D for decryption

     $P = D_{K_D}(E_{K_E}(P))$
   - One of the key of a public key system is public while the other is private

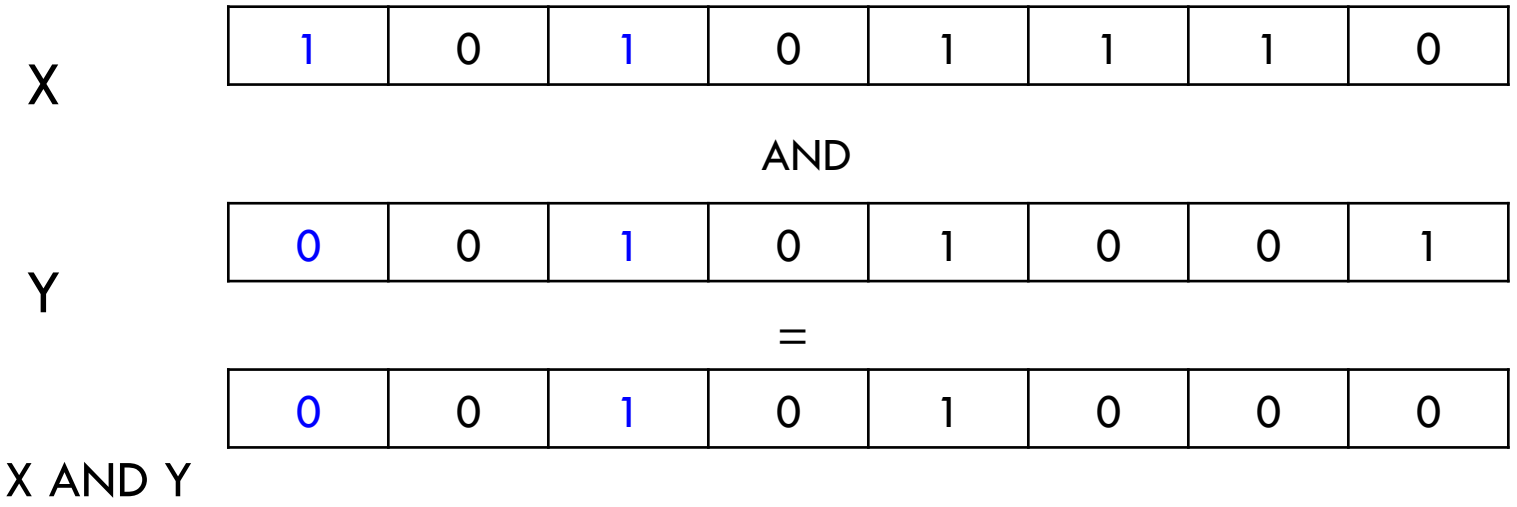
Notation	Description
$K_{A,B}$	Secret key shared by A and B
K_A^+	Public key of A
K_A^-	Private key of A

Bit-wise Operations

THE UNIVERSITY OF SYDNEY

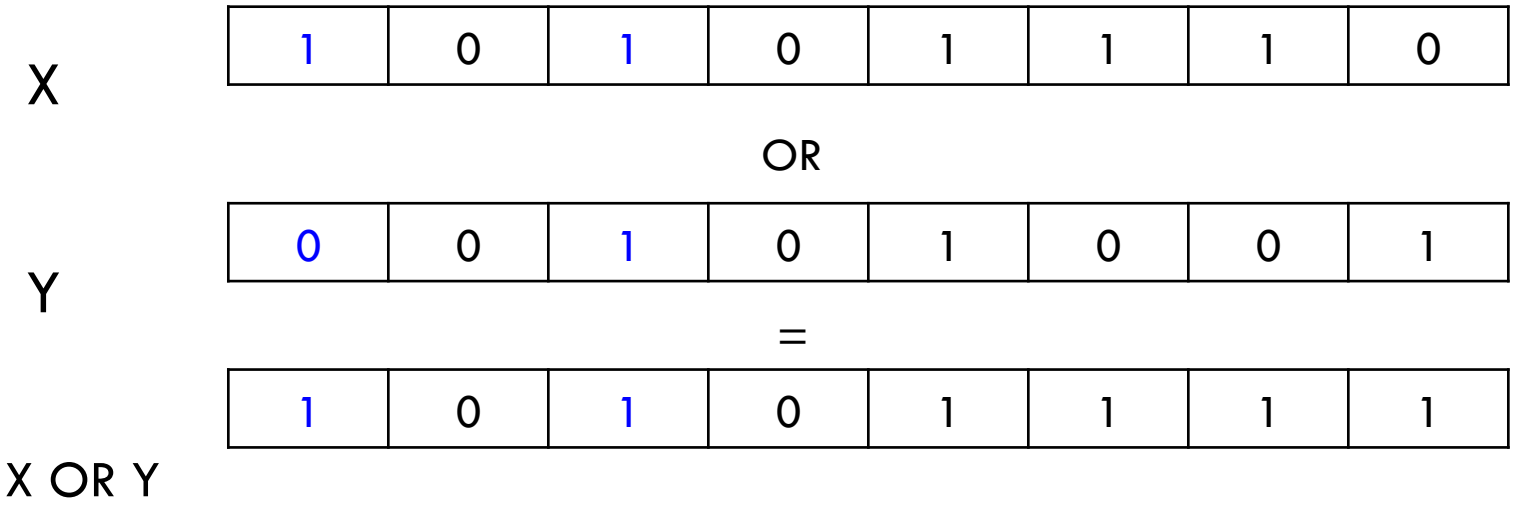
Bit-wise operations

AND Operation



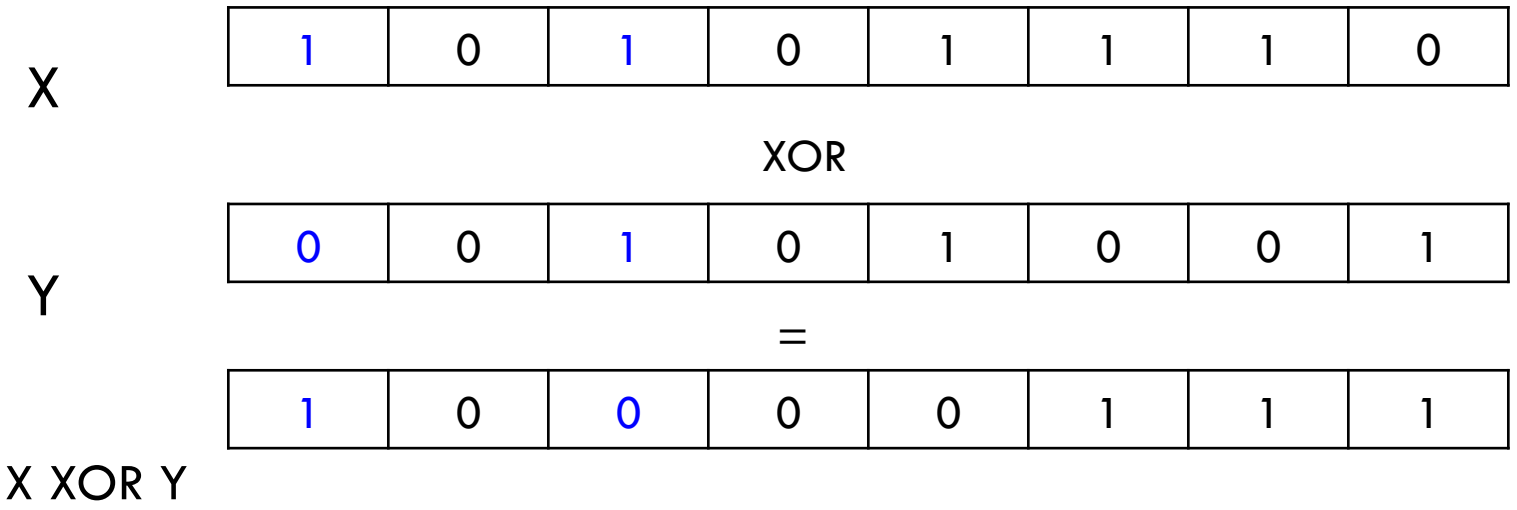
Bit-wise operations

OR operation



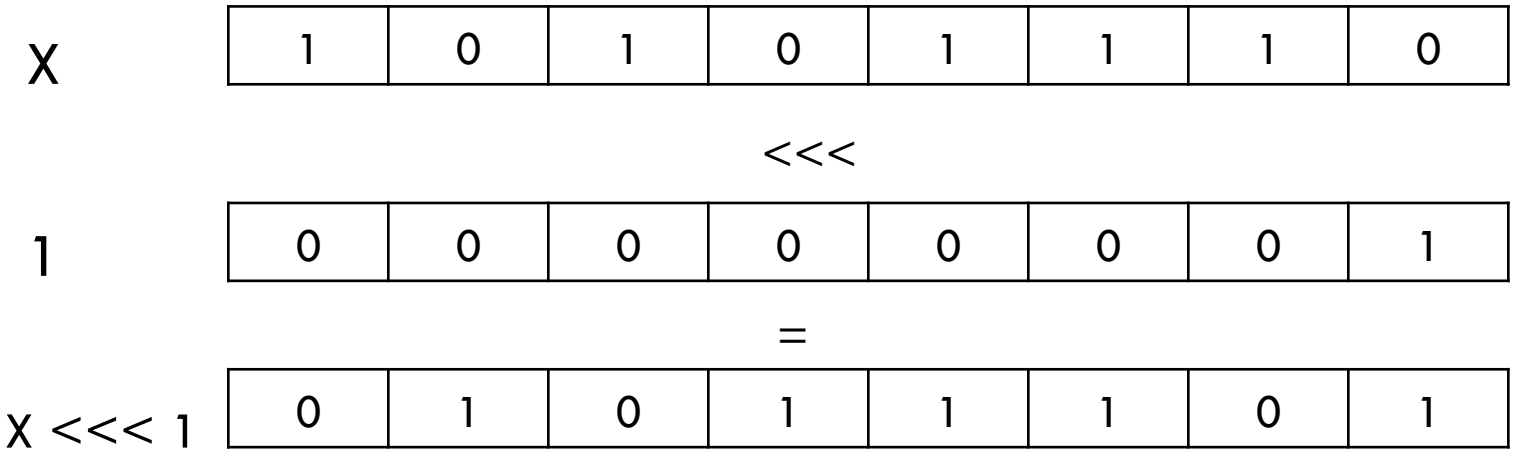
Bit-wise operations

XOR operation



Bit-wise operations

Left rotation



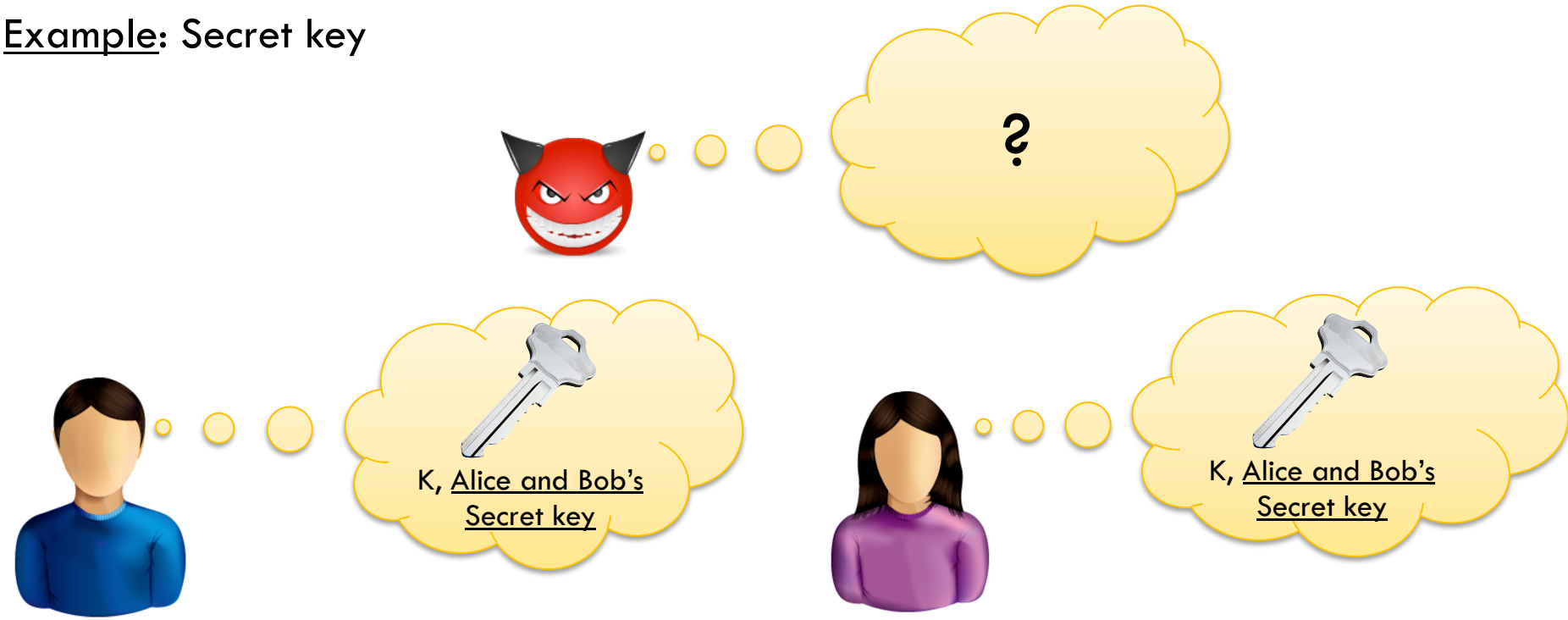
- X <<< 2 = (X <<< 1) <<< 1
- Right rotate: same in the opposite direction

Secret Key System

THE UNIVERSITY OF SYDNEY

Secret key system

Example: Secret key

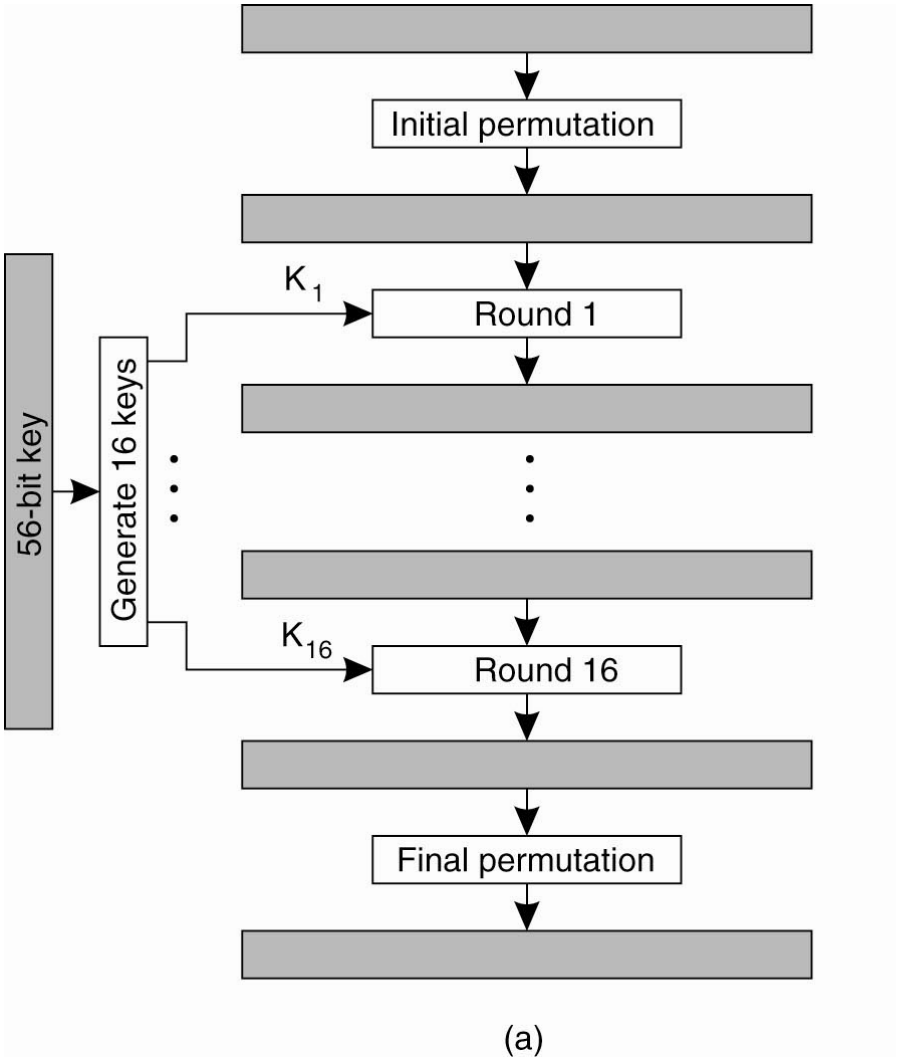


- Consider that Bob wants to sends a protected message to Alice:
  - Bob encrypts the message P using the secret key K into C
  - Upon reception, Alice decrypts C using the secret key K to read P

Secret key system

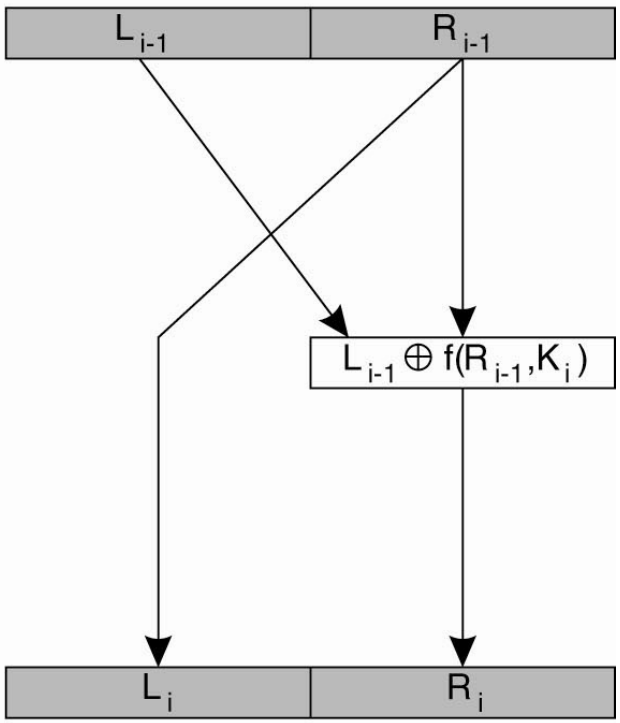
Data Encryption Standard

- Used to operate on 64-bit blocks of data
- A block is transformed into a 64-bit block in 16 encryption rounds
- Each round uses a different 48-bit key for encryption
- Each of these 16 keys is derived from a 56-bit master key
- The inversion of a permutation on the initial value is applied to the result leading to the output



Secret key system

DES (con't)



One encryption round

Encryption in a single round

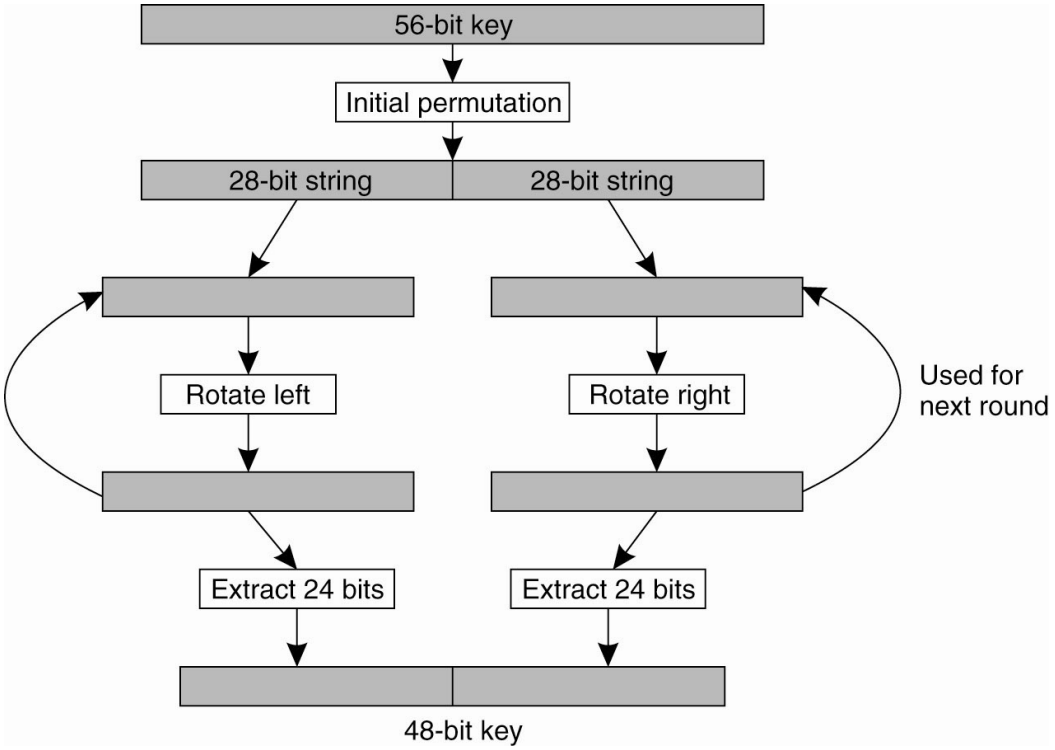
- Round i takes as input the 64-bit block from round i-1
- The block is split in a left part L_{i-1} and a right part R_{i-1}.
- The right part is used for the left part of the next round $L_i = R_{i-1}$
- The function f takes R_{i-1} and key K_i and outputs a 32-bit block
- This output block is XORed with L_{i-1} to output R_i

Secret key system

DES (con't)

Construction of the key K_i for round i

- The master key is permuted and divided into two 28-bit halves
- In each round, each half is first rotated 1 or 2 bits to the left, after which 24 bits are extracted.
- Together with the 24-bits from the other rotated half, a 48 bit key K_i is constructed



Per-round key generation in DES

Secret key system

DES (con't)

- DES
  - Simple enough to be implemented on smart card
  - Resistant to analytical methods
  - Can be broken using brute-force attack by simply searching for a key that breaks it

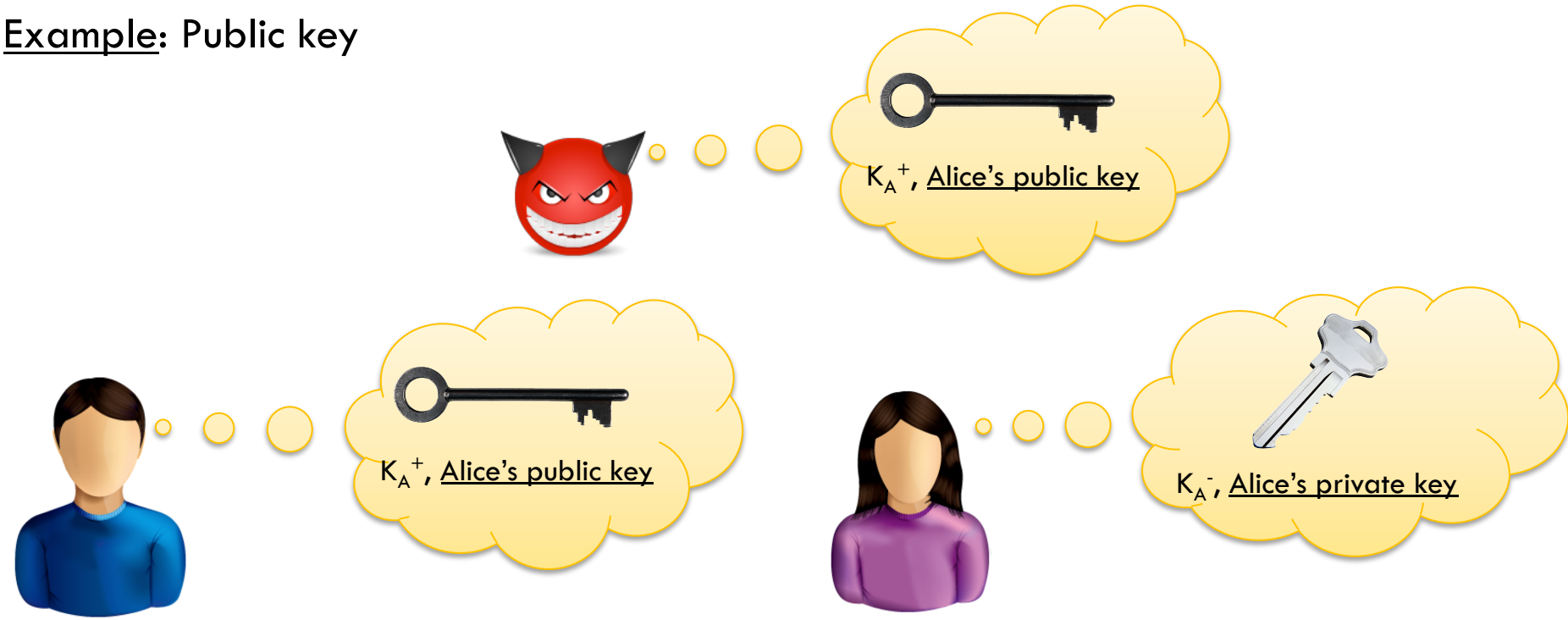
- Triple DES
  - Using DES three times in a special encrypt-decrypt-encrypt mode with different keys
  - Often used

Public Key System

THE UNIVERSITY OF SYDNEY

Public key system

Example: Public key



- Consider that Bob wants to sends a protected message to Alice:
  - Bob encrypts the message m using the public key K_A^+ of Alice into m'
  - Upon reception, Alice decrypts m' using her private key K_A^- to read m
  - Alice is the only one with the private key K_A^-, only Alice can decrypt the message

Public key system

Rivest, Shamir and Adleman (RSA)

- All integers can be written as a multiple of prime number
  e.g., 2100 = 2 * 2 * 3 * 5 * 5 * 7 (2, 3, 5, 7 are the prime factors of 2100)

- No method is known to find the prime factors of large numbers

- In RSA, the private and public keys are constructed from very large prime numbers (w/ 100's of decimal digits)

Public key system

RSA (con't)

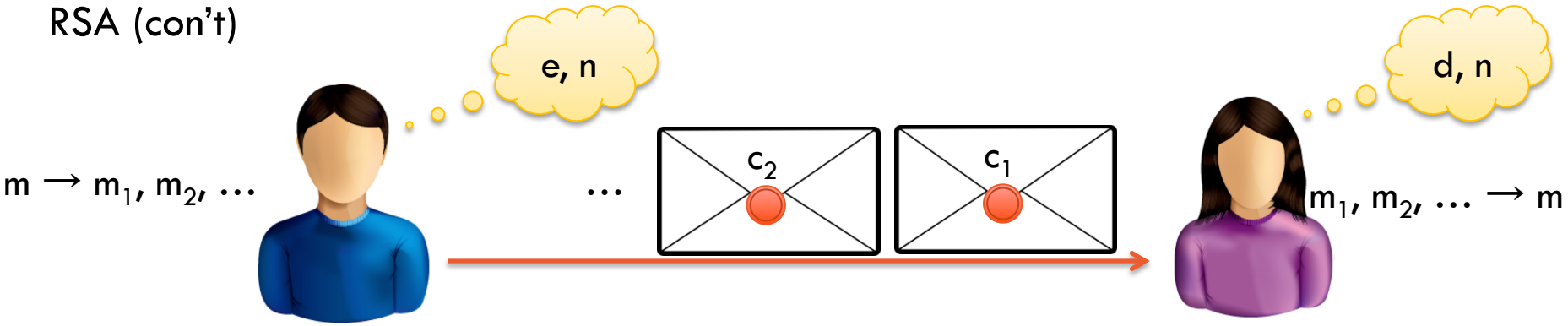
- This is achieved in 4 steps:
  1. Choose large prime numbers p and q
  2. Compute $n = pq$ and $z = (p - 1)(q - 1)$
  3. Choose a number d that is relatively prime to z
  4. Compute the number e such that $ed = 1 \bmod z$

- One number, say d, can be used for decryption while e is used for encryption
  - Message m is divided into fixed length blocks m_i whose binary value is $0 \leq m_i < n$
  - Sender encrypts each block m_i into $c_i = m_i^e \pmod n$ before sending it
  - Receiver decrypts each block $m_i = c_i^d \pmod n$

- Only one among d and e is made public

Public key system

RSA (con't)



- Bob wants to keep the message he sends to Alice confidential
  1. Bob divides the message into fixed length blocks, each block m_i, interpreted as a binary number should lie in $0 \leq m_i < n$
  2. He calculates for each block m_i, $c_i = m_i^e \bmod n$ and sends it to the Bob
  3. Alice computes $m_i = c_i^d \pmod n$ to decrypt the message blocks
- e and n are needed for encryption, whereas d and n are needed for decryption.

- RSA is computationally more complex than DES
- Encrypting message with RSA is much slower (100-1000 multiplying factor were noticed) than DES
- It is thus used essentially to exchange encrypted keys in a secure way
- Less for encrypting "normal" data

Hash Function

THE UNIVERSITY OF SYDNEY

Hash function

Goal: computing a small digest of a message that identifies its content

- Similar to checksum used in TCP/IP
- Deterministic functions: given the same message produces the same digest
- The digest is typically much smaller than the message itself
- Helpful for integrity:
  - Hashing the received message should give the expected digest
  - Otherwise, message integrity is affected

Hash function

Hash function

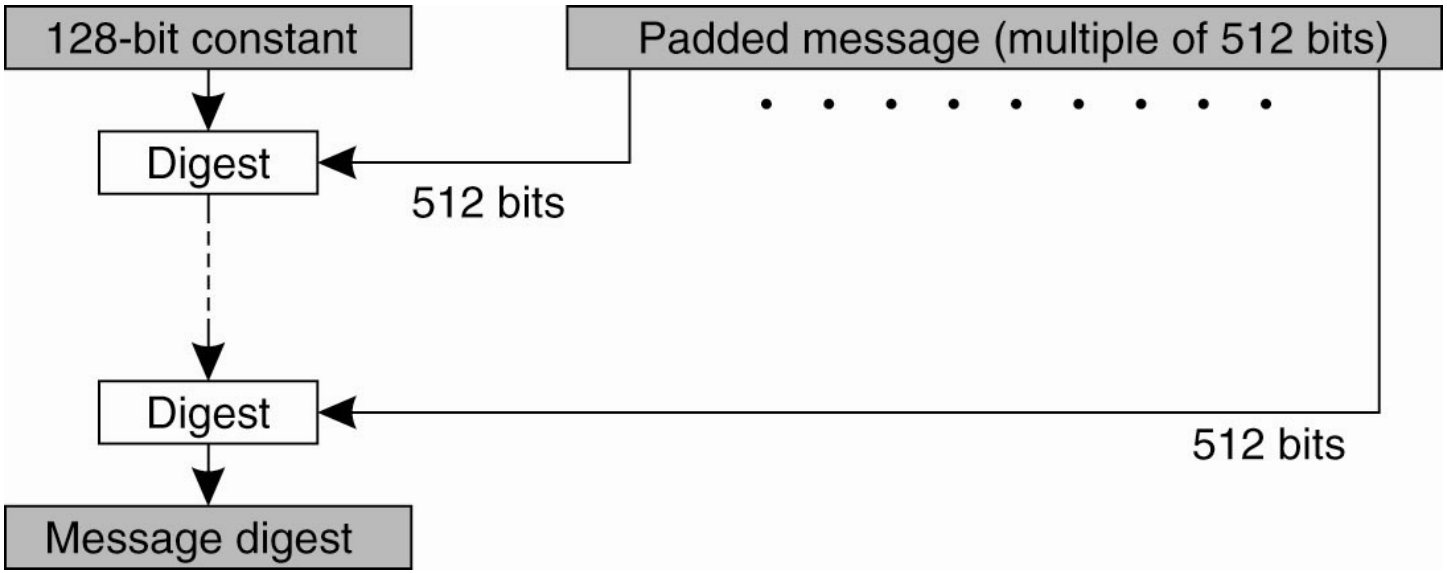
- Hash function H are used to produce a hash h of fixed length given a message m: $h = H(m)$
  - One-way function: computationally infeasible to find an input m that corresponds to an output h, whereas computing h from m is easy
  - Weak collision resistant: given an input m and an output h, it is infeasible to find another different input m' such that $H(m) = H(m')$

- Similarly, same properties apply to encryption function and keys
  - One-way function: computationally infeasible to find a key K when given the plaintext P and associated ciphertext $C = E_K(P)$
  - Weak collision resistant: given a plaintext P and a key K, it is infeasible to find another different key K' such that $E_K(P) = E_{K'}(P)$

Hash function

Message digest 5 (MD5)

- The message is padded to 448 bits (mod 512), appended w/ a 64-bit length
- Starting with some constant 128-bit value, the algorithm has k phases
- k is the number of 512-bit blocks comprising the padded message
- During each phase, a 128-bit digest is computed out of 512-bit block of data coming from the padded message, and the 128-bit digest computing in the preceding phase.



Hash function

MD5 (con't)

- A phase of MD5 consists of 4 rounds of computations, using functions:
  - F(x,y,z) = (x AND y) OR ((NOT x) AND z)
  - G(x,y,z) = (x AND z) OR (y AND (NOT z))
  - H(x,y,z) = x XOR (y XOR z)
  - I(x,y,z) = y XOR (x OR (NOT z))
- Each of these operations operates on 32-bit variables x, y, z
- Consider a 512-bit block b from padded message that is processed in phase k
- Block b is divided into 16 32-bit sub-blocks b0, b1, …, b15
- In each round, the corresponding function executes 16 iterations

MD5 = k phases; 1 phase = 4 rounds; 1 round = 16 iterations

Hash function

MD5 (con't)

- During the 1st round, function F changes variables denoted as p, q, r and s in 16 iterations
- These variables are carried to the next round, and are passed on to the next phase when the current phase is done
- Function G, H, I are used for round 2, 3, and 4 resp.

Iterations 1–8	Iterations 9–16
$p \leftarrow (p + F(q,r,s) + b_0 + C_1) \lll 7$	$p \leftarrow (p + F(q,r,s) + b_8 + C_9) \lll 7$
$s \leftarrow (s + F(p,q,r) + b_1 + C_2) \lll 12$	$s \leftarrow (s + F(p,q,r) + b_9 + C_{10}) \lll 12$
$r \leftarrow (r + F(s,p,q) + b_2 + C_3) \lll 17$	$r \leftarrow (r + F(s,p,q) + b_{10} + C_{11}) \lll 17$
$q \leftarrow (q + F(r,s,p) + b_3 + C_4) \lll 22$	$q \leftarrow (q + F(r,s,p) + b_{11} + C_{12}) \lll 22$
$p \leftarrow (p + F(q,r,s) + b_4 + C_5) \lll 7$	$p \leftarrow (p + F(q,r,s) + b_{12} + C_{13}) \lll 7$
$s \leftarrow (s + F(p,q,r) + b_5 + C_6) \lll 12$	$s \leftarrow (s + F(p,q,r) + b_{13} + C_{14}) \lll 12$
$r \leftarrow (r + F(s,p,q) + b_6 + C_7) \lll 17$	$r \leftarrow (r + F(s,p,q) + b_{14} + C_{15}) \lll 17$
$q \leftarrow (q + F(r,s,p) + b_7 + C_8) \lll 22$	$q \leftarrow (q + F(r,s,p) + b_{15} + C_{16}) \lll 22$

The 16 iterations during the first round in a phase in MD5

MD5 = these 16 operations x 4 rounds (with functions F, G, H, I resp.) x 16 sub-blocks x k phases

Authentication

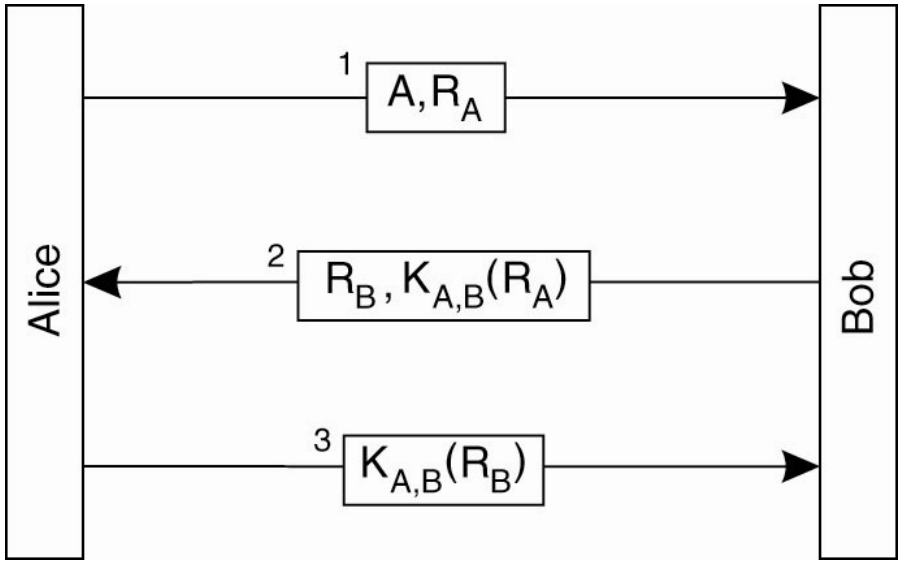
THE UNIVERSITY OF SYDNEY

Authentication

Secret key authentication

- A simpler strategy would be:

1. Alice sends a challenge along with her identity
2. Bob encrypts the challenge with $K_{A,B}$ and sends it with a new challenge
3. Alice encrypts the challenge and sends the result to Bob



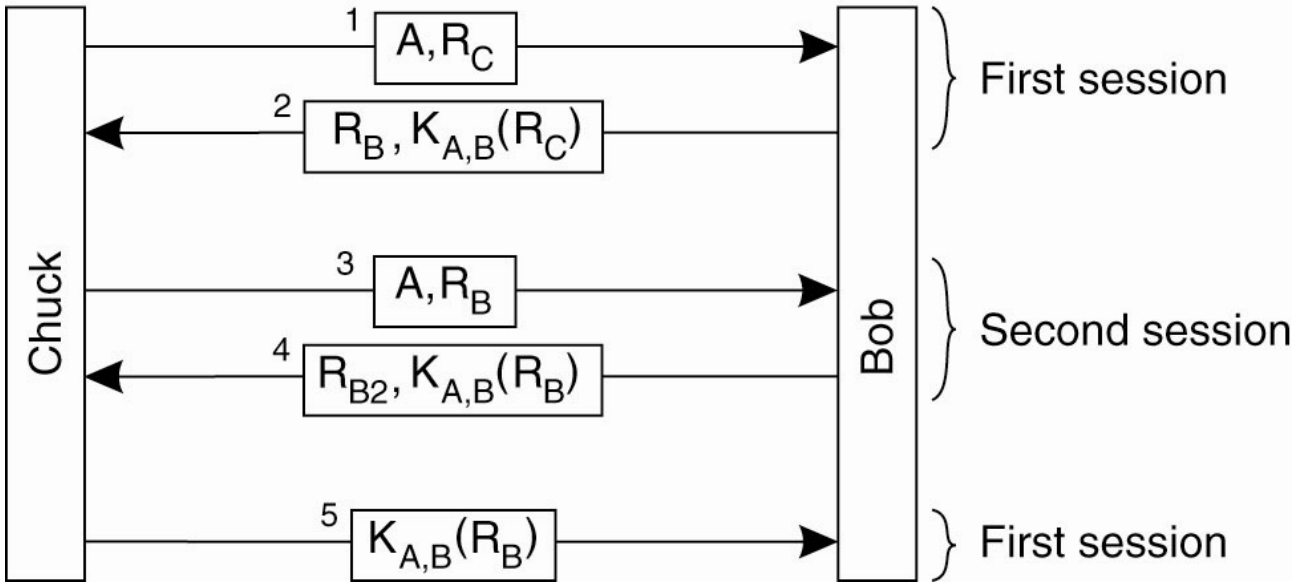
Does it work?

Authentication

Secret key authentication

- A simpler strategy would be:

1. Alice sends a challenge along with her identity
2. Bob encrypts the challenge with $K_{A,B}$ and sends it with a new challenge
3. Alice encrypts the challenge and sends the result to Bob



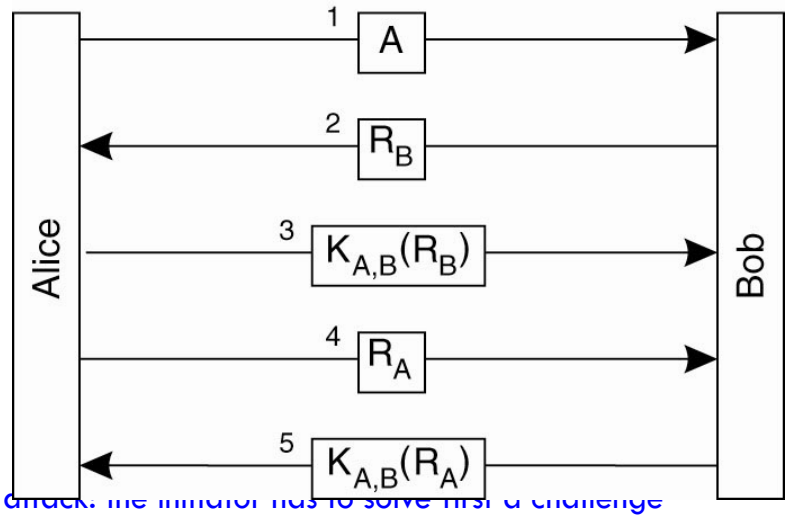
- Reflection attack: Unfortunately, a malicious user can ask Bob to encode many challenges and exploit this encryption to pretend to be Bob

Authentication

Secret key authentication (con't)

- Solution: Alice sends a challenge to Bob that can only be solved if Bob knows the secret key $K_{A,B}$

1. Alice (A) sends her identity to Bob (B) to initiate communication channel
2. Bob sends a challenge(i.e., random number) to Alice
3. Alice encrypts the challenge with the secret key $K_{A,B}$
4. Alice sends a challenge to Bob
5. Bob encrypts the challenge and sends the encrypted challenge $K_{A,B}$



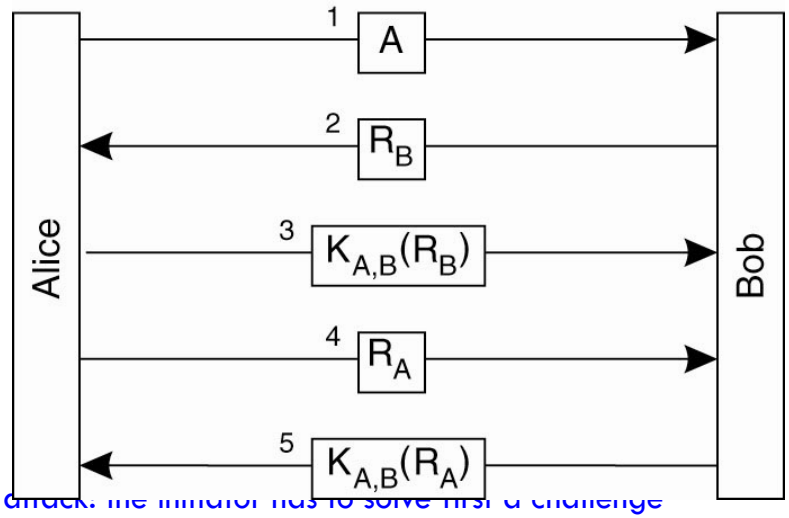
- This remedies the reflection attack: the initiator has to solve first a challenge

Authentication

Secret key authentication (con't)

- Solution: Alice sends a challenge to Bob that can only be solved if Bob knows the secret key $K_{A,B}$

1. Alice (A) sends her identity to Bob (B) to initiate communication channel
2. Bob sends a challenge(i.e., random number) to Alice
3. Alice encrypts the challenge with the secret key $K_{A,B}$
4. Alice sends a challenge to Bob
5. Bob encrypts the challenge and sends the encrypted challenge $K_{A,B}$



- This remedies the reflection attack. The initiator has to solve first a challenge

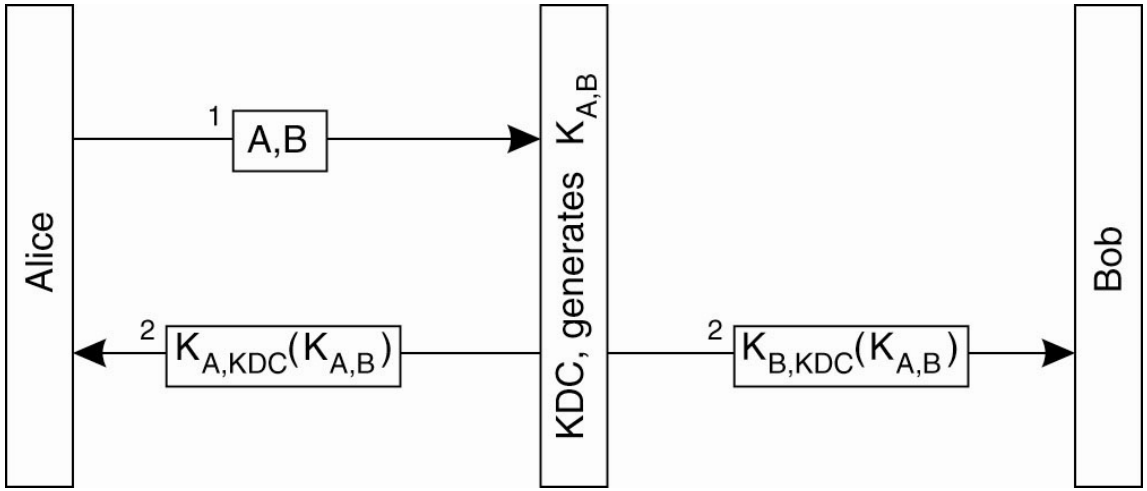
› Many keys would have to be used (n(n-1)/2 keys for communication among n nodes)

Authentication

Key distribution center (KDC) authentication

- A centralized key distribution center (KDC) reduces the number of keys to n.

1. Alice sends a request to communicate with Bob to the KDC
2. The KDC sends the key $K_{A,B}$ encrypted with the key $K_{A,KDC}$ to Alice and the key $K_{A,B}$ encrypted with key $K_{B,KDC}$ to Bob



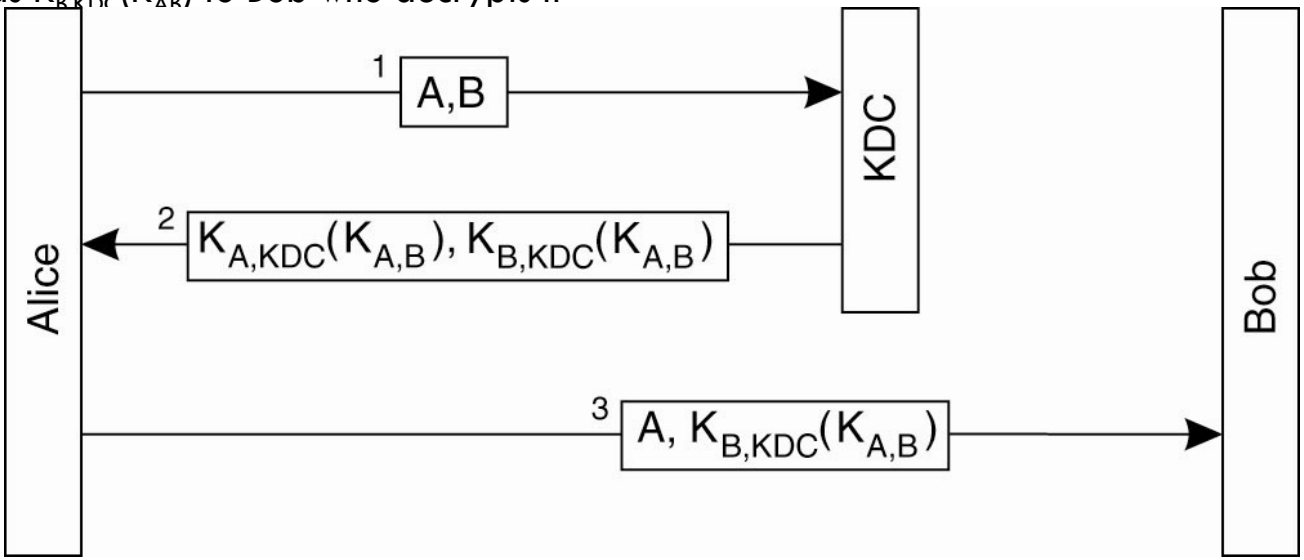
- KDC reduces the number of keys to n
- Alice may use the channel while Bob did not receive the secret key yet

Authentication

KDC authentication (con't)

- It is better to let Alice set up the connection

1. Alice sends a message to KDC with a challenge
2. The KDC responds with a ticket $K_{A,KDC}(K_{A,B})$, $K_{B,KDC}(K_{AB})$
3. Alice sends $K_{B,KDC}(K_{AB})$ to Bob who decrypts it



Alice uses the channel after Bob received the secret key from her

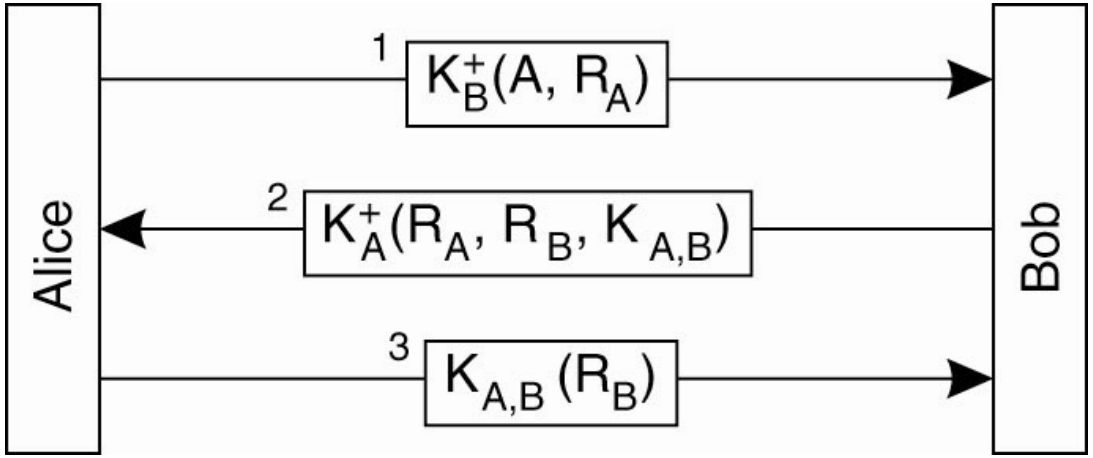
(cf. Needham-Schroeder authentication protocol for further details)

Authentication

Public key authentication

- Assumption: Alice knows Bob's public key

1. Alice sends a challenge R_A to Bob encrypted with Bob's public key K_B^+
2. Bob decrypts the challenge and encrypts with key K_A^+ the result, his own challenge to Alice along with a new generated session key $K_{A,B}$ for further communication
3. Alice decrypts the message using her key K_A^-
4. Alice responds to Bob using the session key $K_{A,B}$ generated by Bob



Does not require a centralized KDC

Integrity

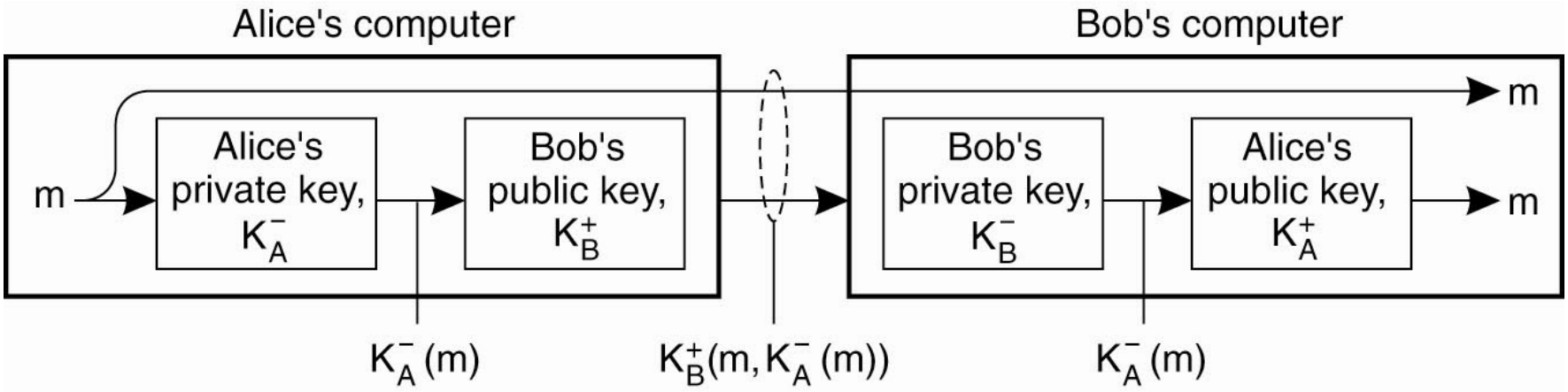
THE UNIVERSITY OF SYDNEY

Message integrity

Public key signature

- Alice sends a message P to Bob

1. Alice encrypts it with her private key K_A^- and sends it off to Bob
2. She can use Bob's public key K_B^+ to keep the message secret and sends $K_B^+(P, K_A^-(P))$, combining P and the version she signed
3. Bob decrypts the signed version of the message with Alice's public key. If the message is the same as the non-signed one, then it has been sent by Alice.



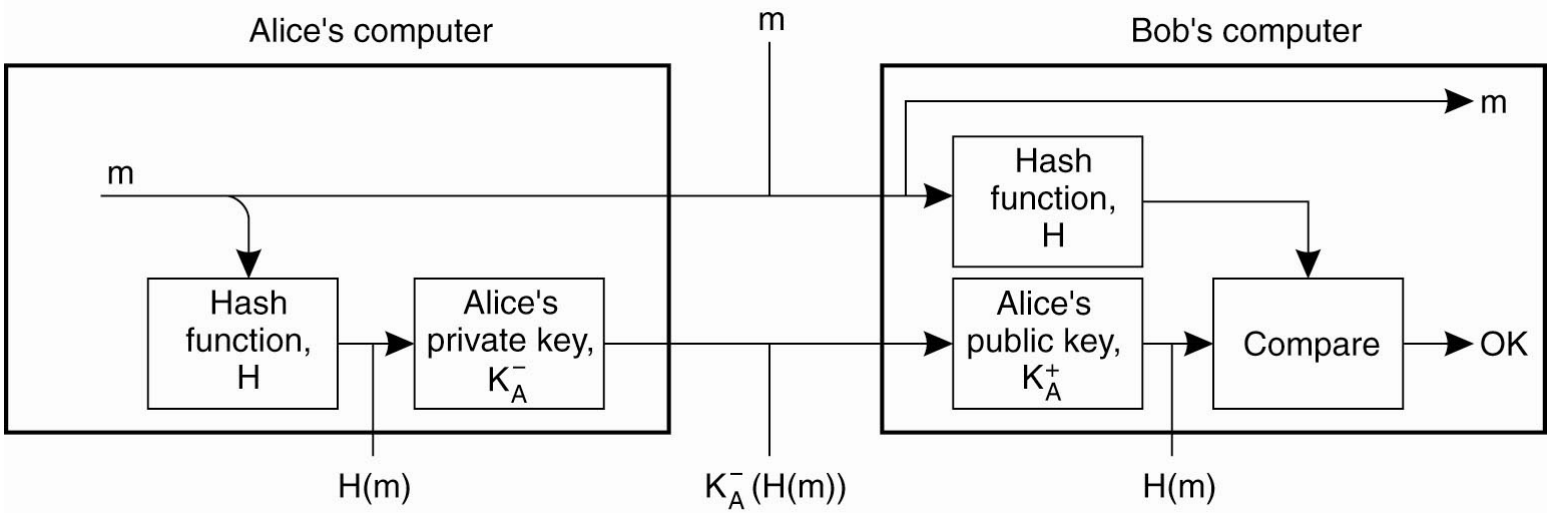
The entire message must be encrypted

Message integrity

Public key and hash function signature

- Alice sends a message P to Bob

1. Alice computes a message digest and encrypts it with her private key K_A^- (The message can be sent encrypted using Bob's public key K_B^+)
2. Bob decrypts the message with Alice's public key and computes the message digest. If the digest computed from the message received and the decrypted digest match, Bob knows the message has been signed by Alice



The digest is typically smaller than the encrypted message

Conclusion

- Confidentiality requires protection when messages traverse the network
- Integrity (avoid fabrication and modification) requires message digest or signatures
- Cryptography addresses these challenges but solutions are often complex