Computer Networks and Communication

Lecture 14-15

Network Security

Introduction to Network Security

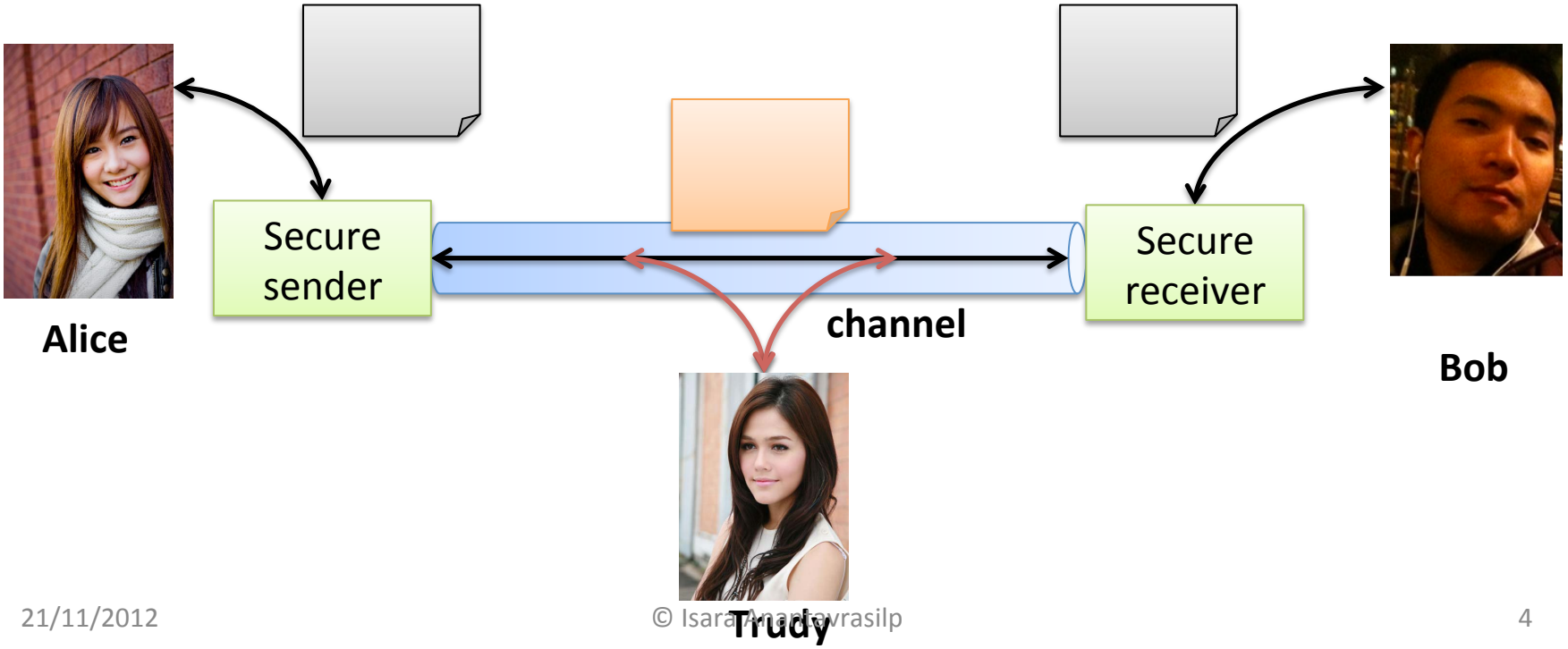
- In the last two classes, we will discuss:
  - Principle of network security:
    - Cryptography
    - Message integrity
    - Authentication
    - Key distribution
  - Security in practice

Network Security Consists of..

- **Confidentiality**: only sender, intended receiver should "understand" message contents
  - sender encrypts message
  - receiver decrypts message
- **Message Integrity**: sender and receiver want to ensure message not altered (**in transit, or afterwards**) without detection
- **Authentication**: sender and receiver want to confirm identity of each other
- **Access and Availability**: services must be accessible and available to users

Network-*In*security

- Introducing Alice, Bob and Trudy
- Bob and Alice want to communicate securely
- Trudy does not like that and wants to intercept, delete or alter the messages



Bob and Alice

- In real life, they could be:
  - WWW server and web browser
    - Webmail
    - Online banking
  - DNS servers
  - Routers exchanging routing tables
  - Remote login client and server
    - Telnet
    - FTP

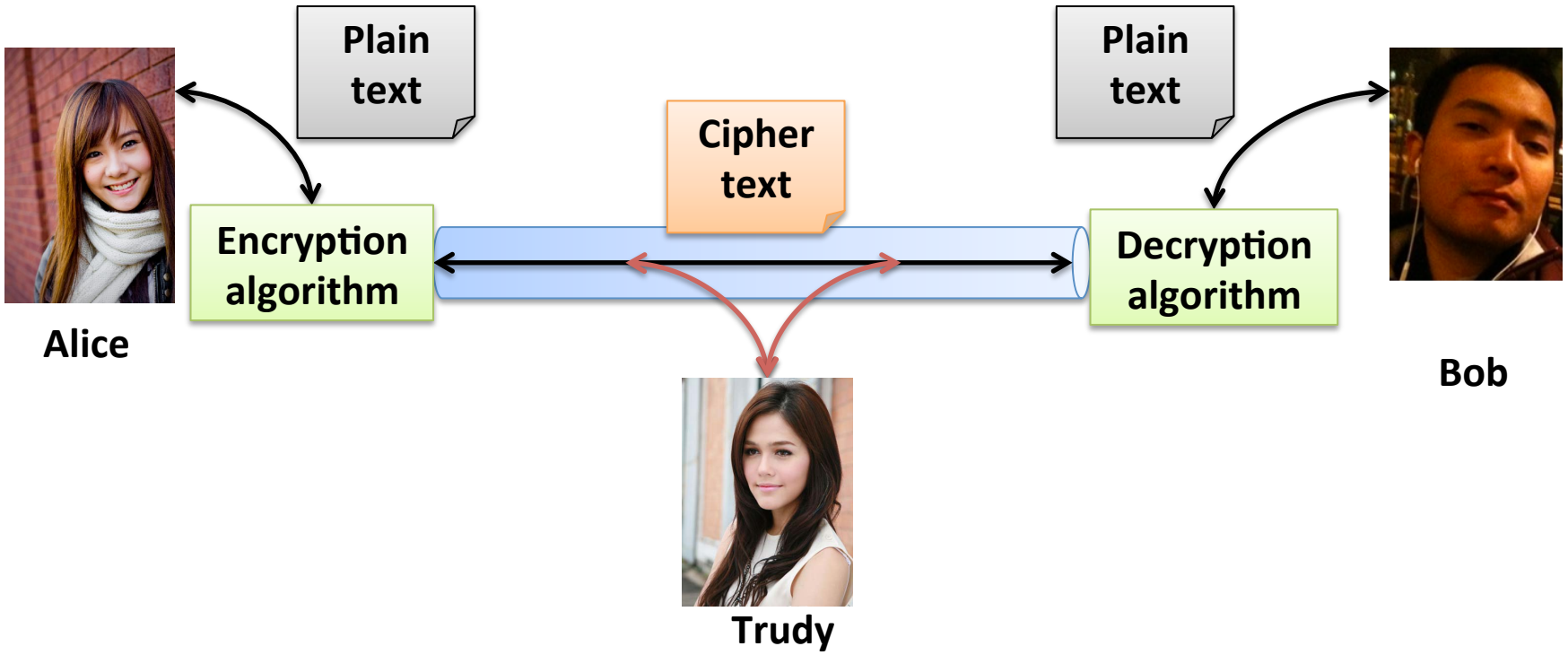
Trudy

- What can Trudy do (or want to do)?
  - **eavesdrop**: intercept messages
  - actively **insert** messages into connection
  - **impersonation**: can fake (spoof) source address in packet (or any field in packet)
  - **hijacking**: "take over" ongoing connection by removing sender or receiver, inserting himself in place
  - **denial of service**: prevent service from being used by others (e.g., by overloading resources)

Principle of Cryptography

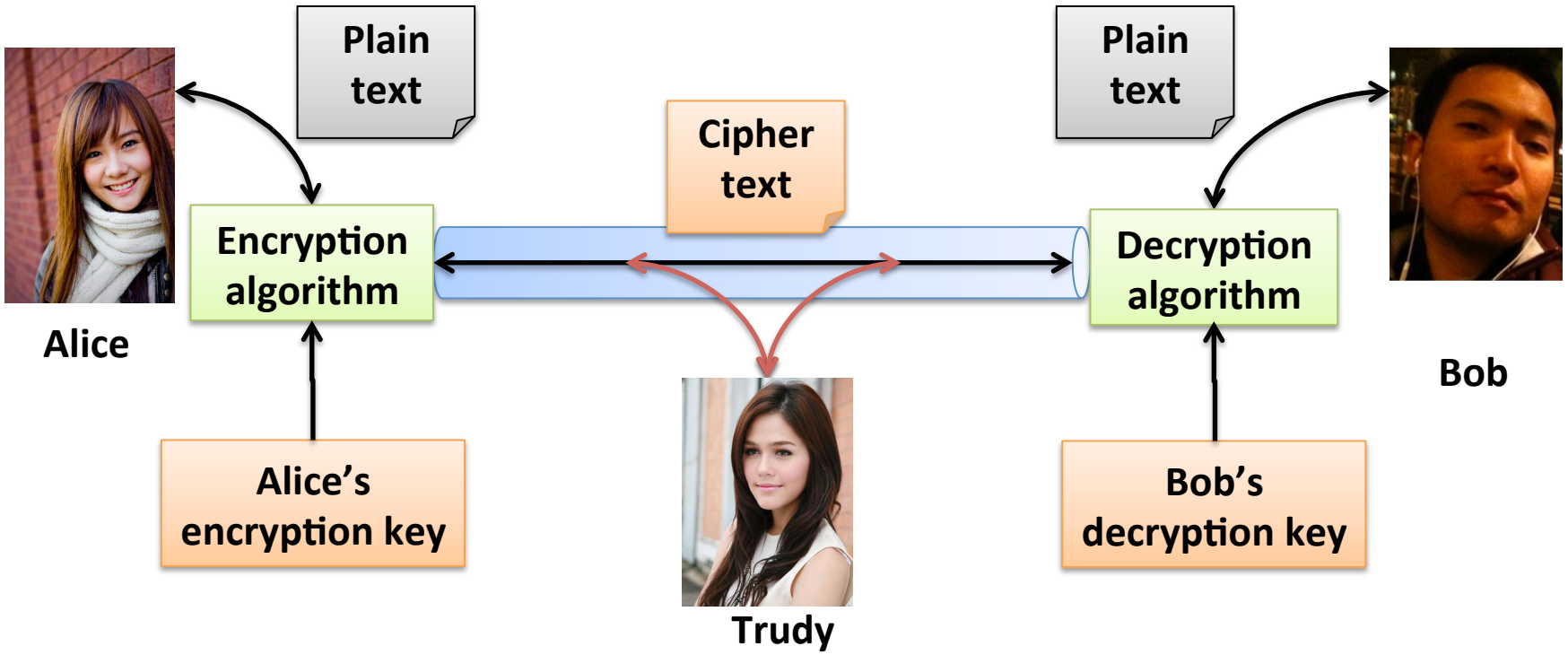
- **Cryptography**:
  - Study of techniques for secure communication
  - From Greek words: secret writing
  - Has been used for a few thousand years
    - Most well know ancient crypto: **Caesar Cipher**
- Types of Cryptography
  - **Cipher**: Character-to-character or bit-to-bit transformation
  - **Code**: Replacing one word with another word or sentences. (For example, Morse Code)

Cryptography Terminology



- How to be sure Trudy will never be able to decrypt cipher text, ever?
  - Keep changing the algorithm?

En-/Decryption Keys



- The en-/decryption algorithms should be the same
- We should be able to change a small parameter that influences the resulting cipher text
  - **Keys**: Making new key is easier than making new kind of lock

Symmetric and Public Keys

- **Symmetric key** crypto:
  - Sender and receiver keys are the same
  - That is, encryption and decryption employ the same key
  - The sender and receiver must keep this key secret
- **Public key** crypto:
  - **Encryption** key is **public**. Everyone can encrypt using this key.
  - **Decryption** key is **private**. The receiver must keep this key secret.
- In both scheme, **the key to the strength of the crypto is the length (size) of the key**

Symmetric Key Cryptography

- **Caesar Cipher**: Replace a character with the character of the next *k* position
  - *k* = 3
  - **A** -> **D**, **B** -> **E**, **Z** -> **C**
- **Substitution Cipher**: substituting one thing for another
  - Example**: monoalphabetic cipher**: substitute one letter for another letter

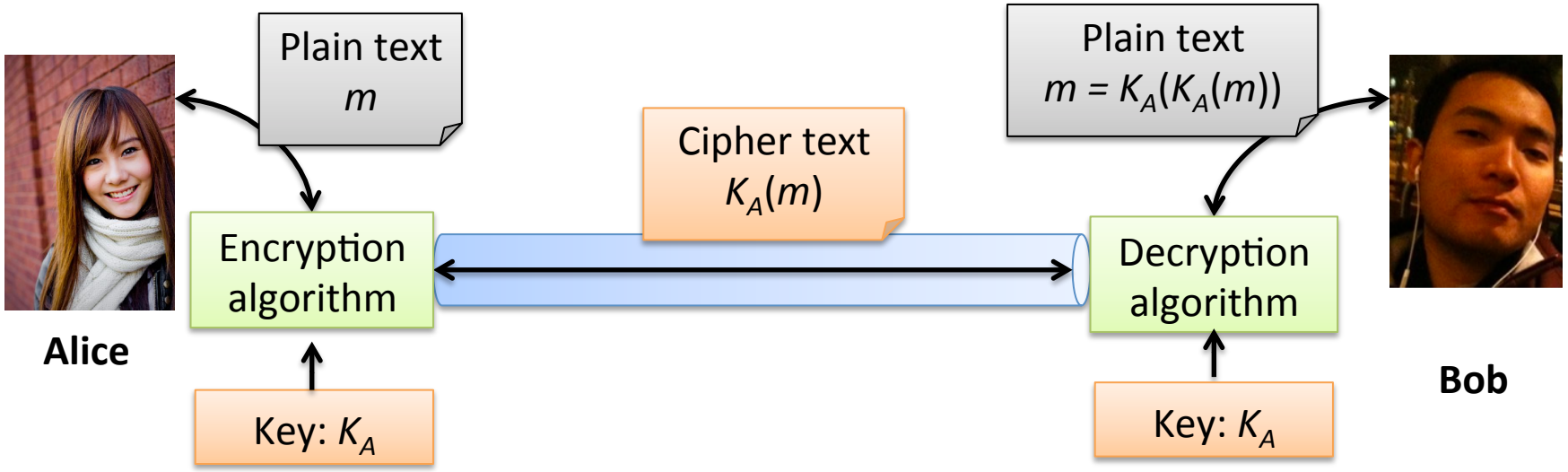
```
plaintext:   abcdefghijklmnopqrstuvwxyz
                ↓                    ↓
ciphertext:  mnbvcxzasdfghjklpoiuytrewq

Plaintext:  bob. i love you. alice

ciphertext:  nkn. s gktc wky. mgsbc
```

How hard it is to crack this code?

Symmetric Key Cryptography (2)



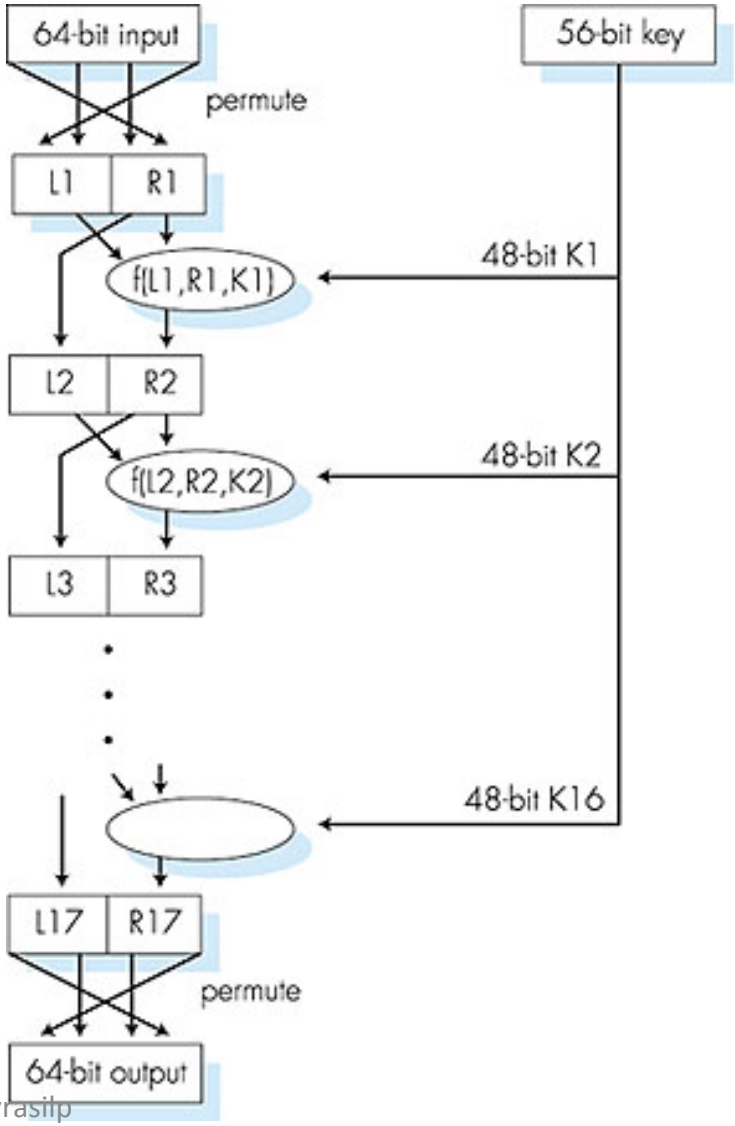
- Symmetric key crypto:
  - Bob and Alice share know same (symmetric) key: K
  - e.g., key is knowing substitution pattern in mono alphabetic substitution cipher
- How do Bob and Alice agree on key value?

Symmetric Key Crypto: DES

- **DES: Data Encryption Standard**
- US encryption standard [NIST 1993]
- **56-bit** symmetric key
- **64-bit** plaintext input
- Co-developed by IBM and NSA
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase
  - Decrypted (brute force) in 4 months

DES (2)

- DES operation
- Initial permutation
  - Apply the same scrambling function 16 times
  - Each using different 48 bits of key
- Final permutation



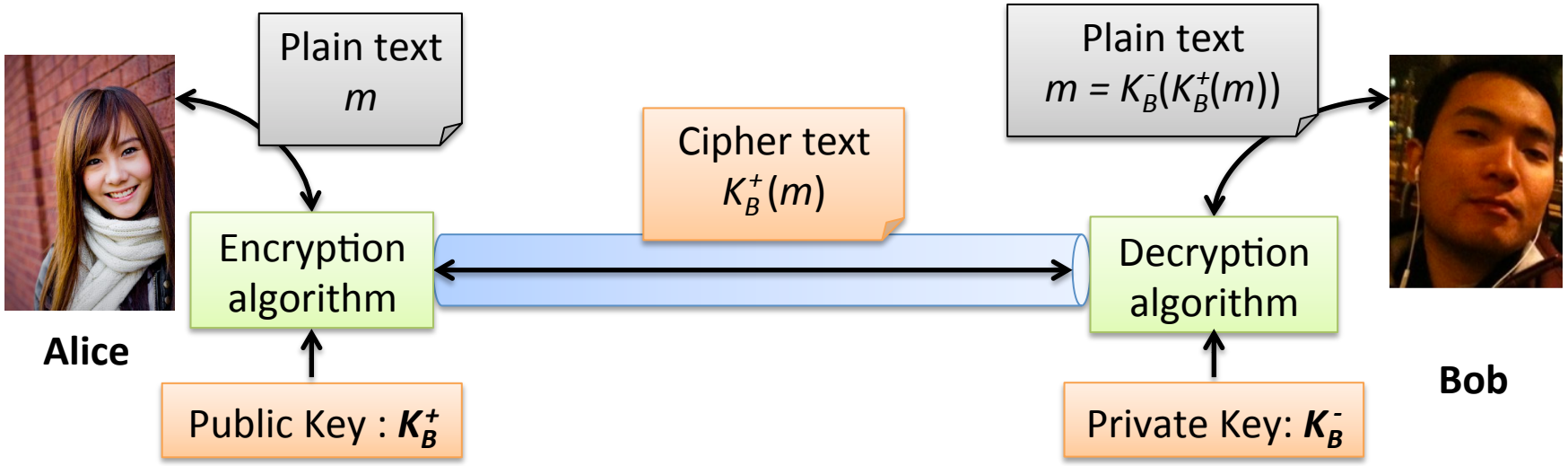
AES

- **AES: Advanced Encryption System**
  - New (Nov. 2001) symmetric-key standard, replacing DES
  - Specified by National Institute of Standards and Technology (NIST)
- Processes data in **128 bit** blocks
- **128,** 192, or **256 bit keys**
- Brute force decryption (try each key)
  - Taking 1 sec on DES
  - Takes 149 trillion years for AES

Problem with Symmetric Key

- Key distribution: How can the two parties agree on a key?
- This is one of the most challenging problem in the entire (military) history until 1970s.

Public Key Cryptography



- Public Key Cryptography:
  - Sender and receiver do not share secret key
  - **Public encryption key** known to all
  - **Private decryption key** known only to receiver
  - Easy to distribute (public) keys

Public Key Encryption Algorithm

- Algorithm-design requirements
  - Need $K_B^+()$ and $K_B^-()$ such that

$$K_B^-(K_B^+(m)) = m$$

  - Given public key K_B^+ ,it should be impossible to compute private key K_B^-
  - K_B^+ is strong and cannot be broken

RSA

- **RSA (Rivest, Shamir, Adelson)** algorithm
  - The first and still very popular algorithm that satisfies requirements in the previous slide
  - Very strong but slow to compute:
    - Long key is required (1024 bits or more)
  - Exploit a very difficult mathematical problem: Determine if a number is prime
  - RSA uses two large prime numbers and compute the key from them
  - It also has the following property:

$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

Agenda

- Principle of network security:
  - ✔ – Cryptography
  - – Message integrity
  - – Authentication
  - – Key distribution
- Security in practice

Message Integrity

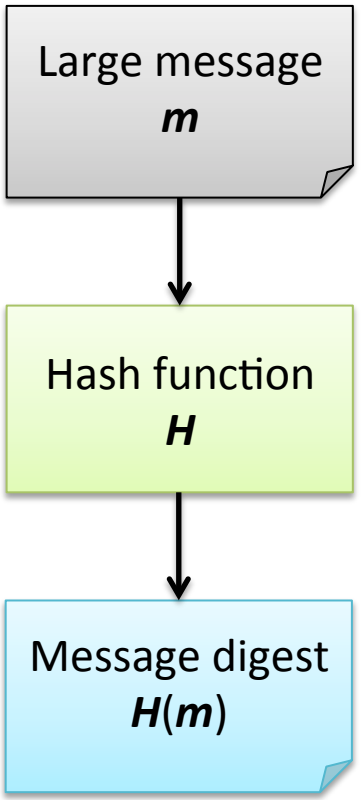
- Example scenario:
  - Bob sends a message to Alice
  - Alice wants to verify if the message is from Bob and unchanged
  - This is call **message integrity** problem
- To achieve this, Alice and Bob need an encryption algorithm ***H*** which has the following property:
  - Given a message ***m***, and
  - the encrypted message ***H***(***m***),
  - It is *very* hard to find ***m'*** such that ***H***(***m'***) = ***H***(***m***)
  - That is, you cannot change ***m*** to ***m'*** and still get ***H***(***m***)

Message Integrity (2)

- Assume that we have ***H*** that has the properties mentioned in the previous slide
- Alice can verify Bob's message as follows:
  - Bob uses a **secret** encryption key ***H*** to encrypt a message ***m*** and get encrypted messaged ***H***(***m***)
  - ***H*** is known to both Bob and Alice and no one else
  - Bob sends both m and ***H***(***m***) to Alice: (***m***, ***H***(***m***))
  - After Alice receives ***m***, she applies ***H*** to ***m***
  - Alice checks if ***H***(***m***) = ***m***
  - If true, then ***m*** is the original message
- But do such functions exist?

Message Digest

- ***H*** does indeed exist!
  - Many of them, actually
  - Hash functions have those properties
- Hash function properties:
  - Many-to-1
  - Produces **fixed-size** **message digest** (also called **Fingerprint**)
  - Given message digest ***H(m)***, computationally infeasible to find ***m'*** such that ***H(m')*** = ***H***(***m***)
    - You cannot compute original message m from ***H***(***m***)
    - You have to know ***m*** to reproduce ***H***(***m***)



Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - Computes **128-bit** message digest in 4-step process.
  - Arbitrary **128-bit** string ***x***, appears difficult to construct message ***m*** whose MD5 hash is equal to ***x***
  - Invented by Ronald Rivest who co-invented RSA
- SHA-1 is also used.
  - US standard [NIST, FIPS PUB 180-1]
  - **160-bit** message digest

Where are we now?

- Network security provides

✔ – **Confidentiality**:
  - Keep the message secret
  - Key cryptography solves this

✔ – **Message Integrity**:
  - Ensure message not altered (**in transit, or afterwards**)
  - Message digest solves this

– **Authentication**:
  - Sender and receiver want to confirm identity of each other

Review on Public Key Cryptography

- Public key:
  - Public, open for everyone
  - Used to **encrypt** messages to a recipient
  - Public key of Bob is denoted: K_B^+
- Private key
  - Kept secret by the owner alone
  - Used to **decrypt** messages sent to the key owner
  - Private key of Bob is denoted: K_B^-
- Public key algorithm: **RSA**
- RSA encryption and decryption properties:

$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

  - You can also use private key to encrypt and use the public key to decrypt

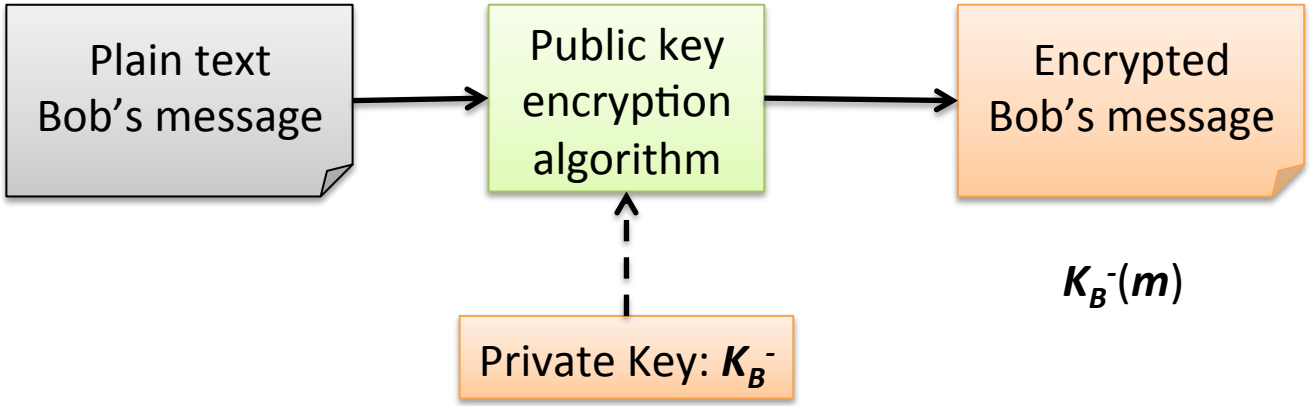
Digital Signature

- **Digital Signature**: Cryptographic technique analogous to hand-written signatures.
- Sender (Bob) digitally signs document, establishing he is the document owner/creator.
- **Verifiable, non-forgeable**: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document
- You want to know if you are talking to a bank
- The bank also wants to know the client is really you

Digital Signature with Public Key

- Simple digital signature for message m:
  - Bob signs ***m*** by encrypting with his private key K_B^-, creating **signed** message, $K_B^-(m)$
  - Only Bob has his private key K_B^-

Bob



Digital Signature with Public Key (2)

- Suppose Alice receives
  - The message ***m***
  - Digital signature $K_B^-(m)$
- Alice verifies ***m*** signed by Bob by
  - Applying Bob's public key K_B^+ to $K_B^-(m)$
  - Then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.
- Thus, Alice can be sure that
  - Bob signed ***m***
  - No one else signed ***m***
  - Bob signed m and not ***m'***

Digital Signature with Public Key (3)

- Public key cryptography can be used in digital signing
- However, RSA algorithm is computational expensive
  - We cannot practically use public and private keys to decrypt and encrypt the entire message
- We can use RSA to encrypt only the fixed-size **fingerprint**
  - Bob has to sign only the fingerprint

How Large is RSA Message?

- From message: “attack at dawn”
- Translated into ASCII:

  19766202164023008896244827187751 50
- And encrypted with RSA:

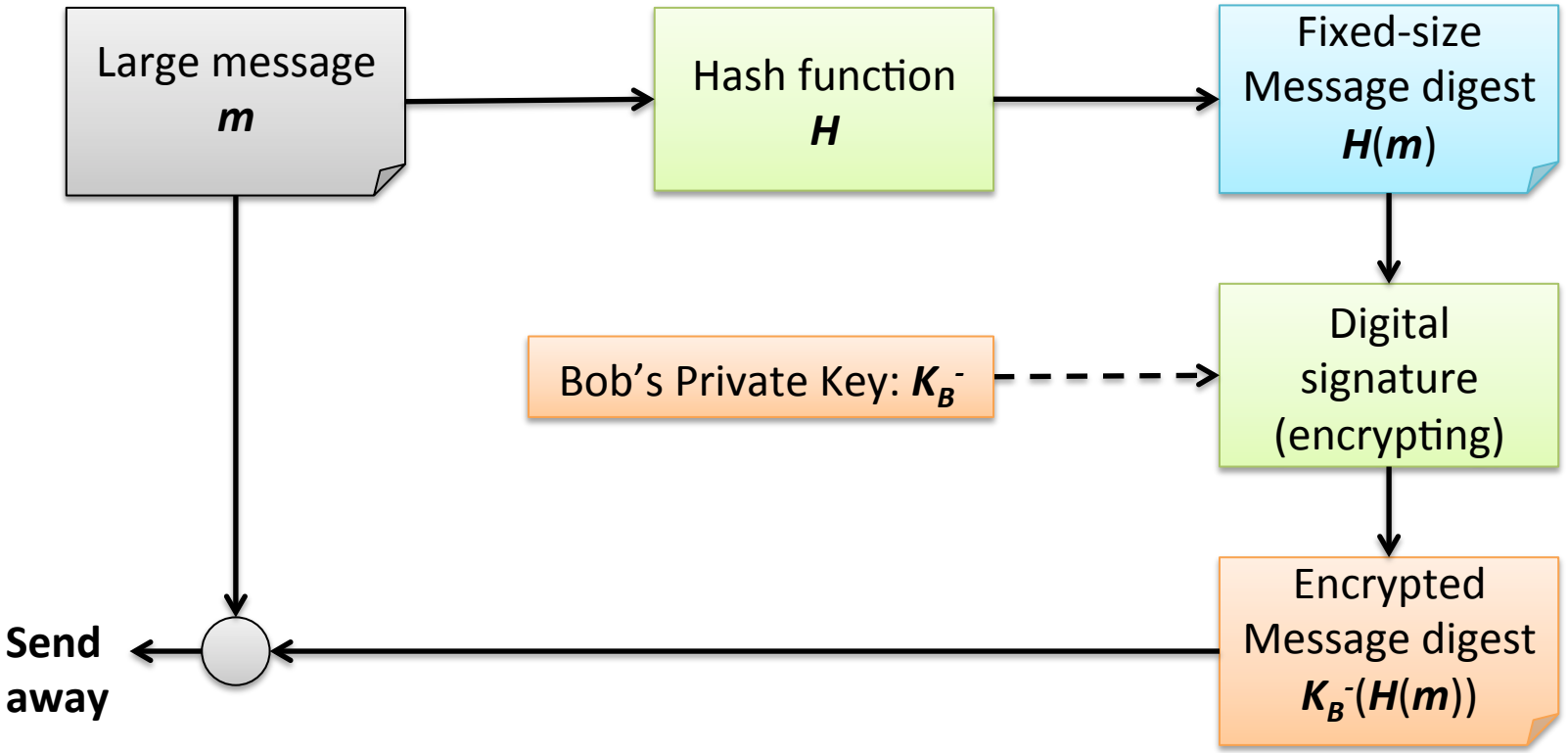
```
350521113386730266902124239370533285118807608115799816206428023466858106231098502359430490809733862411137840407947041939782153784997654130836464387847409523069325349451950801838615742252262188798272324539128205968864403775360824656817500744174591514854074458625110234722355608230534977915189288202722577877 86
```

- And we have not considered the encryption/decryption time yet!

Example source: http://percepi.blogspot.com/2012/05/how-rsa-works-with-examples.html

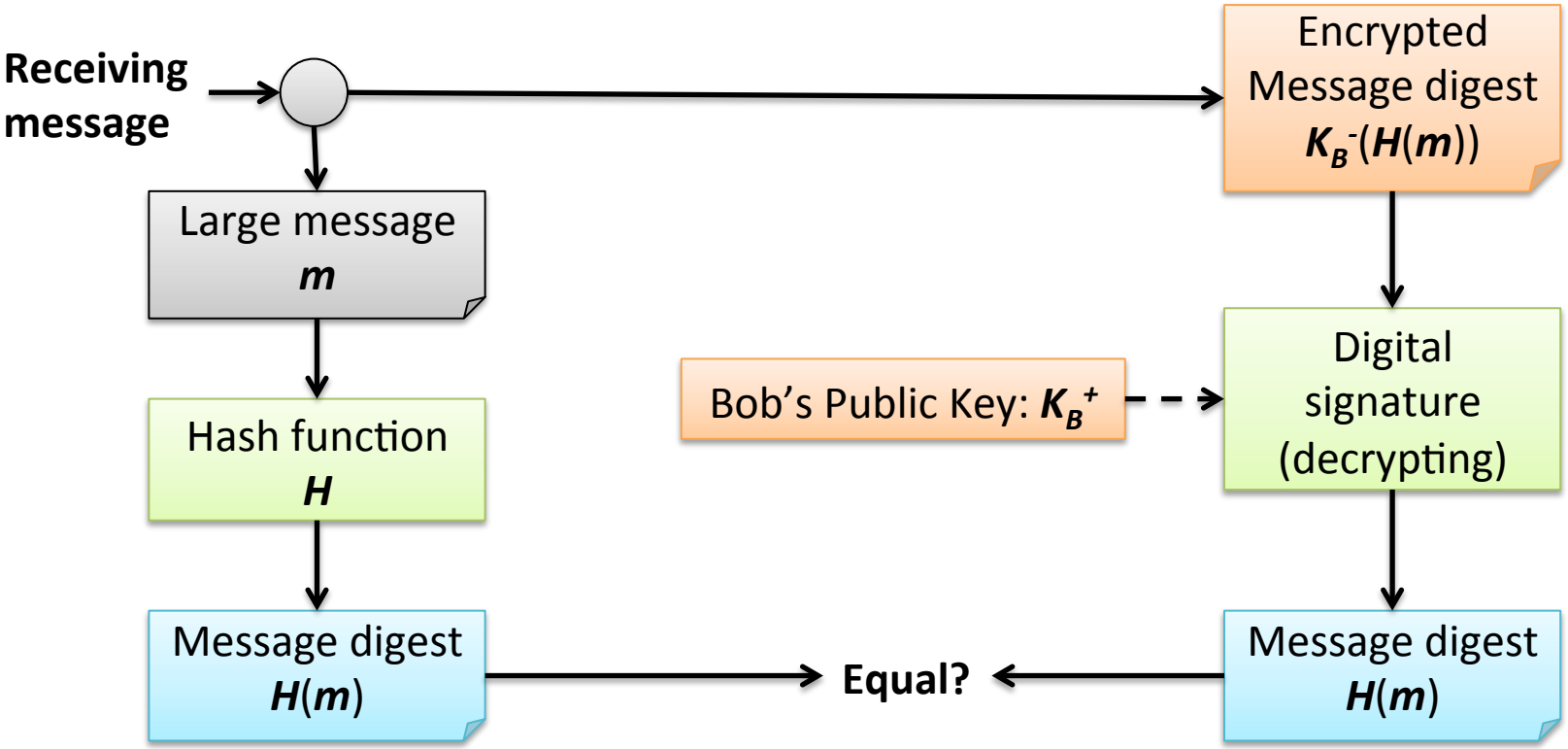
Signed Message Digest

- Bob sends digitally signed message



Verifying Signed Message Digest

- Alice verifies signature and integrity of digitally signed message



Signed Message Digest (2)

- With signed message digest, Alice can be sure that Bob's message has not changed
  - Because Bob signed message fingerprint with his private key
  - We have **message integrity**
- But what about confidentiality and authenticity?
  - There is no confidentiality! Why?
  - Also still no authenticity! Why?

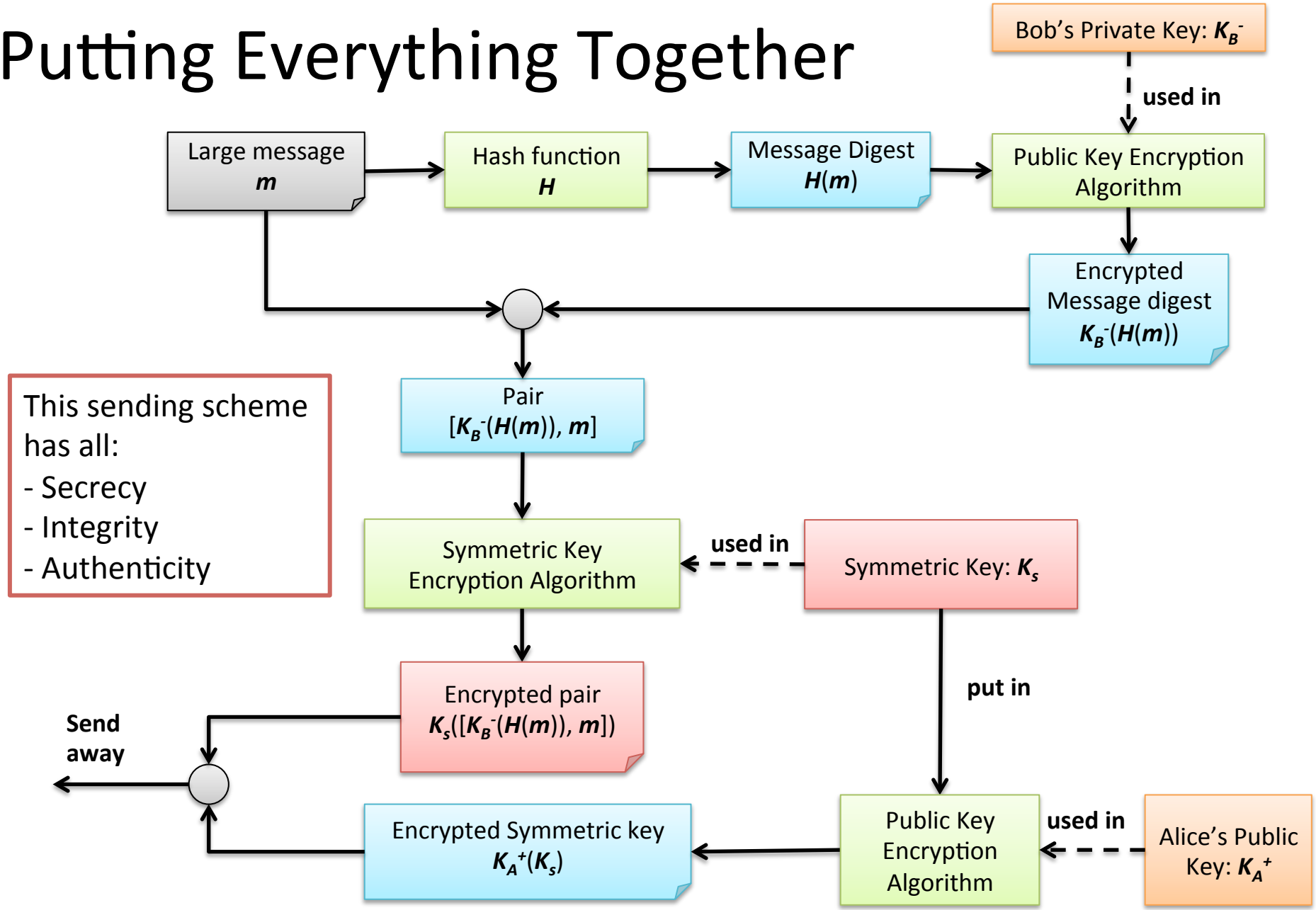
Problems with Secure Message

- Using a hash function to compute message digest (fingerprint) yields only message integrity, (incomplete) authentication but no secrecy
  - Because we always need to send plaintext message along with the fingerprint
  - We cannot (by design) revert fingerprint into the plaintext
- Using only RSA to compute encrypted message yields integrity and (incomplete) authentication and secrecy but too expensive

Toward Completely Secure Message

- We want secrecy, integrity and authenticity
- How do we do that?
  - With a symmetric key!
- Symmetric key scheme is super fast compared to public key scheme (e.g. RSA)
- We can use symmetric key to encrypt any message easily and quickly
- **Problem**: Key distribution
- **Solution**: Use public key to encrypt only the "**key**"

Putting Everything Together



Who is the Sender?

- Key cryptography and digital signature:
  - Alice can verify messages from Bob by decrypting Bob's signature using Bob's public key K_B^+
  - Here, Alice assumes that only Bob has K_B^-
- Question: How can Alice be sure that K_B^- and K_B^+ really belong to Bob?
- Example scenario:
  - Trudy gives Alice her public key, K_T^+,
  - Then Trudy says she is Bob and K_T^+ is Bob's private key
  - How can Alice knows who really is Bob?
  - How can Alice knows which key K_T^+ or K_B^+ belongs to real Bob?

Authentication Issues

- Goal: Bob wants to make sure that he is talking to Alice. (Prove her identity)

Alice

"I am Alice"

Bob

What could go wrong?

Trudy

Spoofing Attack

- Goal: Bob wants to make sure that he is talking to Alice. (Prove her identity)

Alice

Bob

"I am Alice"

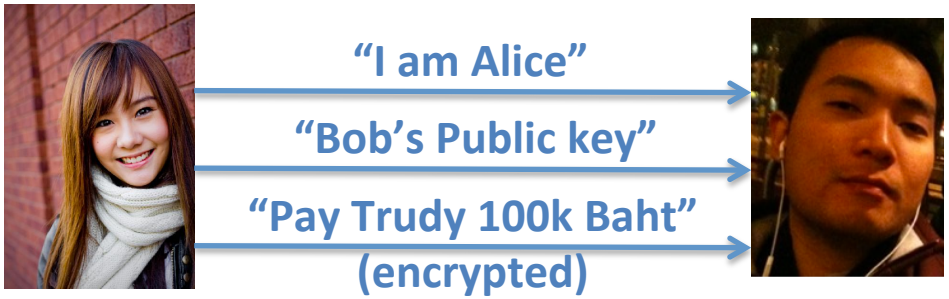
Trudy

Bob thinks he is talking to Alice
However, Trudy impersonates herself as Alice

"Spoofing attack"
- **IP spoofing**
- **ARP spoofing**
- **Web site spoofing/phishing**

Authentication Issues (2)

- **Situation 2**: Alice encrypt the data with public key



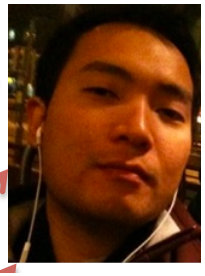

Trudy

What could go wrong?

Playback Attack

- **Situation 2**: Bob and Alice agree on a common password

Alice



"I am Alice"

"Bob's Public key"

Bob

"Pay Trudy 100k Baht"
(encrypted)

Trudy

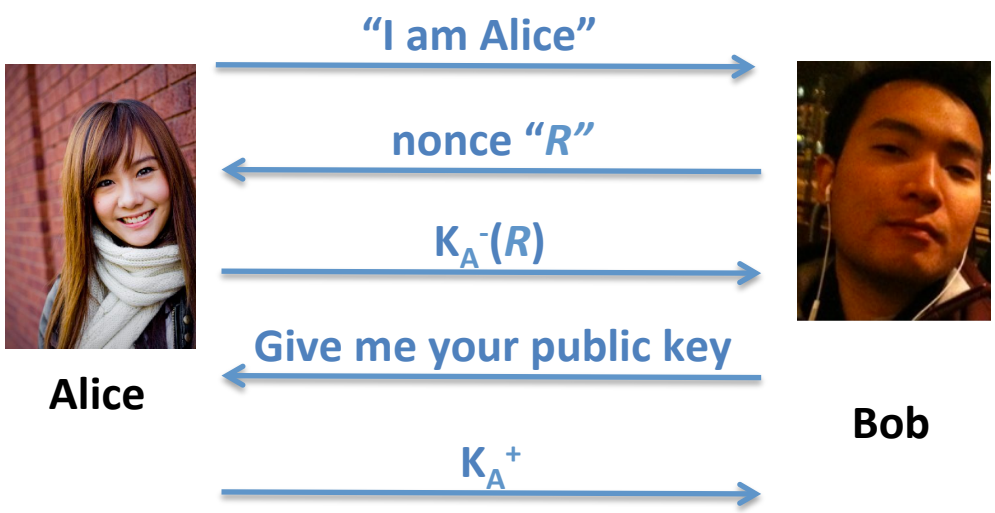
Trudy eavesdrops the conversation and "replays" the messages

This method works even when the password is encrypted

"playback attack"

Preventing Playback

- Before exchanging messages, Bob sends Alice a **nonce**
  - A random message
  - Used only once in a lifetime
- Then Alice encrypt the nonce with her private key and send back the encrypted nonce



Bob computes
$K_A^+(K_A^-(R)) = R$
He knows that only Alice could have the private key K_A^- such that $K_A^+(K_A^-(R)) = R$

Nonce *R* ensures that the message is fresh (not replayed)

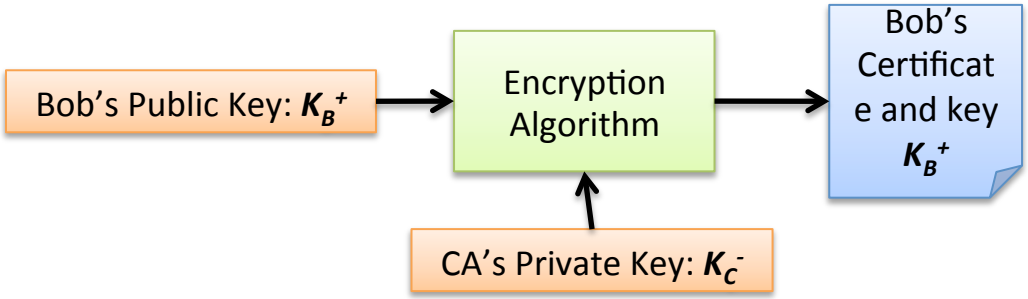
K_A ensures identity of Alice

Certification Authority

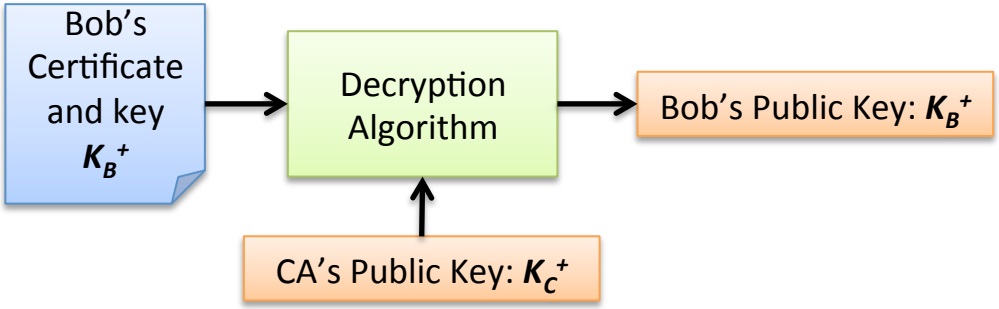
- Public key is useful only when you can verify that you have actual public key of an **entity,** e.g.
  - **Person**: Am I sending an email to the correct guy?
  - **Website**: Are we at the real website of a bank?
- Thus, we need a **trusted** third party who can verify identity information for us
- **Certification Authority (CA)**
  - Validate identities of entities
  - Issue certificates: After an entity is verified, a **certificate** that binds the entity and its public-key is issued

Key Verification with CA

- Bob verifies his identity with a CA and obtain a certificate

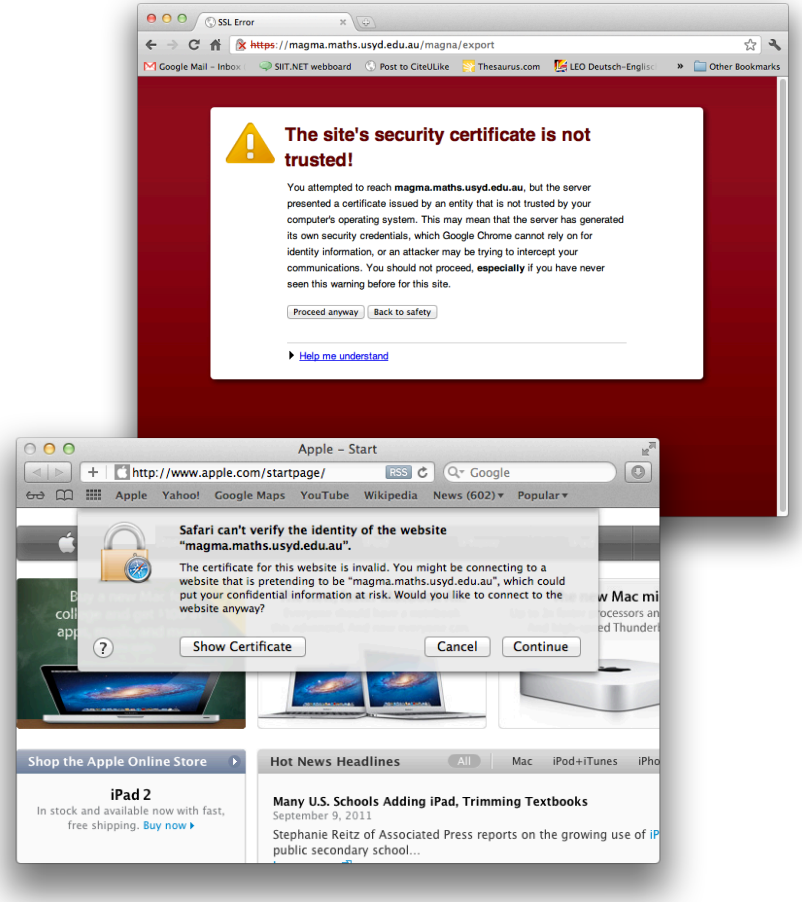


- With certificate, before Alice trusts Bob's key, she has to verify with CA first



Security Certificate in Practice

- We use certificate authentication every day
  - Do you realize it?
- Web browsers verify identities of websites for us automatically
  - URL or web address
- What if CA is fake too?
  - Can we trust any CA?
  - "**Web of trust**"

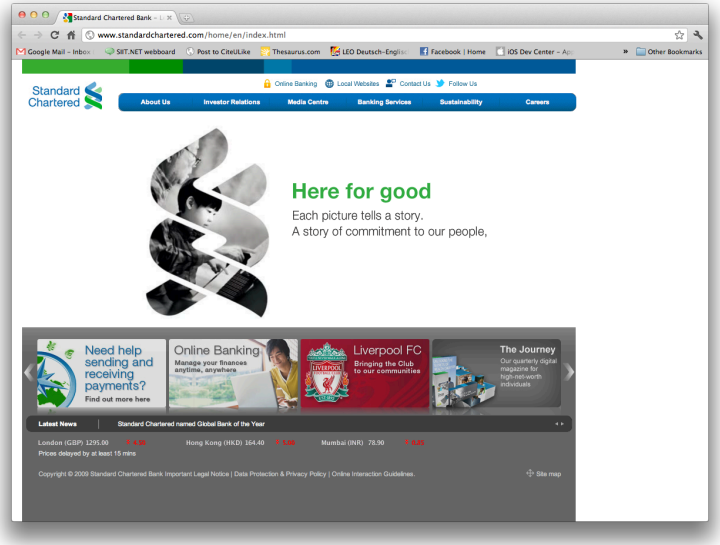


Authentication in Real Life

- When you go to a bank you see this:

- You always know where you are

- When you go to an online bank you see this:



- Do you really know where you are?

Yes, with certificates

Agenda

- Principle of network security:
- ✔ – Cryptography
- ✔ – Message integrity
- ✔ – Authentication
- ✔ – Key distribution
- Security in practice

PGP

- **Pretty Good Privacy (PGP)**
  - Email encryption scheme similar to the one on Slide 39
  - The most popular scheme on the planet
  - Uses symmetric key cryptography, public key cryptography, hash function, and digital signature as described.
  - Provides secrecy, sender authentication, integrity.
  - Inventor, Phil Zimmerman, was target of 3-year federal investigation.

PGP Signed Message

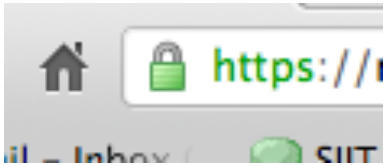
```
---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

Bob:My husband is out of town tonight.
   Passionately yours, Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZP9t6n7G6m5Gw2
---END PGP SIGNATURE---
```

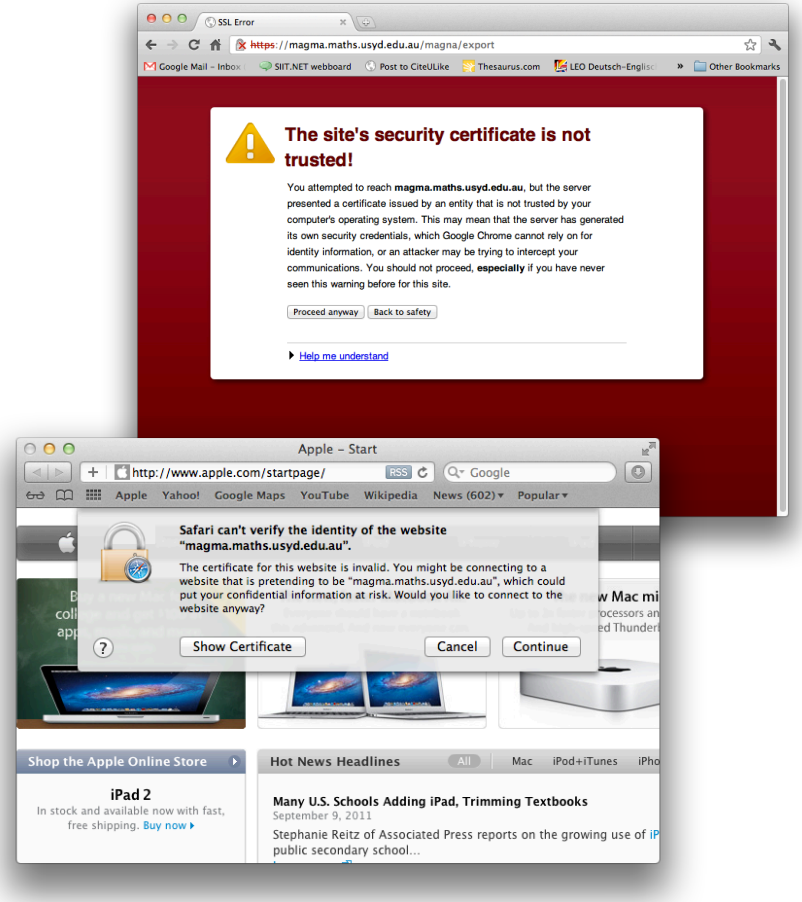
Secure Socket Layer

- **Secure Socket Layer (SSL)**
  - Transport layer security to any TCP-based app using SSL services.
  - Used between Web browsers, servers for e-commerce (**https**).
- Provided secure services:
  - Server authentication
  - Data encryption
  - Client authentication (optional)



Secure Socket Layer (2)

- Server authentication:
  - SSL-enabled browser includes public keys for trusted CAs.
  - Browser requests server certificate, issued by trusted CA.
  - Browser uses CA's public key to extract server's public key from certificate.



Agenda

- Principle of network security:
- ✔ – Cryptography
- ✔ – Message integrity
- ✔ – Authentication
- ✔ – Key distribution
- ✔ Security in practice