



# Romain Fontugne

2019 Fall Semester

## Information Network Systems

*Security, cryptography*

### Last lecture

We've seen how to monitor traffic

- Tcpdump/Wireshark
  - Record/monitor traffic
  - Get a comprehensive view of network traffic
  - Detect anomalies

Romain Fontugne

Information Network Systems

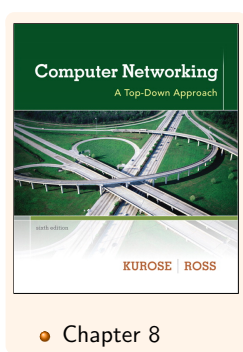
2019 Fall Semester

1 / 42

### Today's Lecture: Security Cryptography

### What is Network Security?

- 1 What is Network Security?
- 2 Symmetric Key Cryptography
- 3 Public Key Cryptography
- 4 Message Integrity
- 5 End-Point Authentication



• Chapter 8

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

Romain Fontugne

Information Network Systems

2019 Fall Semester

2 / 42

Romain Fontugne

Information Network Systems

2019 Fall Semester

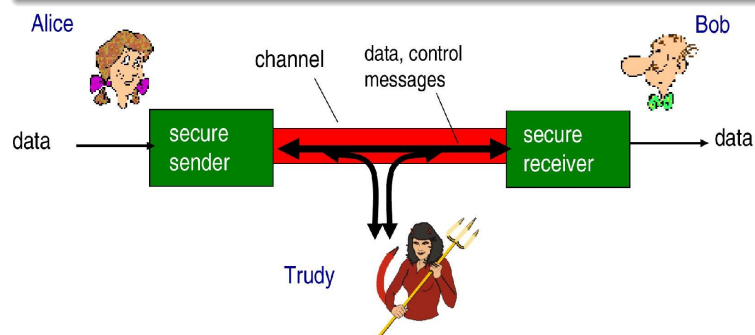
3 / 42

### Friends and enemies: Alice, Bob, Trudy

### Who might Bob and Alice be?

#### Example

- Well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



- ... well, real-life Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- On-line banking client/server
- DNS servers
- Routers exchanging routing table updates
- Other examples?

Romain Fontugne

Information Network Systems

2019 Fall Semester

4 / 42

Romain Fontugne

Information Network Systems

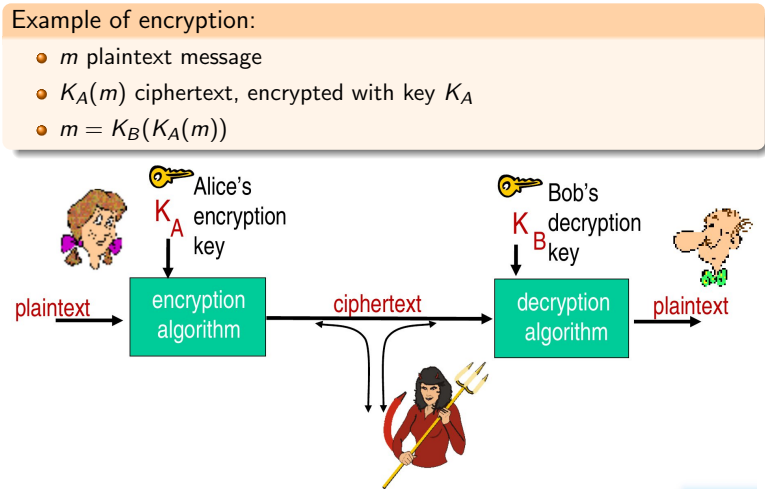
2019 Fall Semester

5 / 42

There are bad guys (and girls) out there!

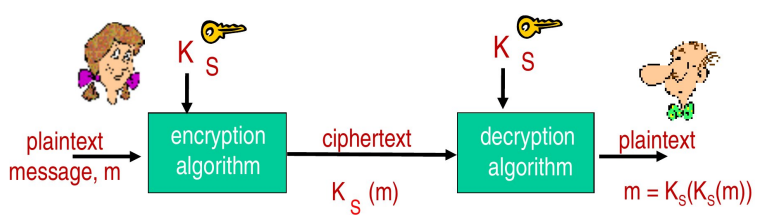
- What can a “bad guy” do?
- **Eavesdrop**: intercept messages
  - Actively **modify/insert/delete messages** into connection
  - **Impersonation**: can fake (spoof) source address in packet (or any field in packet)
  - **Hijacking**: “take over” ongoing connection by removing sender or receiver, inserting himself in place
  - **Denial of service (DoS)**: prevent service from being used by others (e.g., by overloading resources)

Principles of Cryptography



Symmetric key cryptography

- Bob and Alice share 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?



Simple encryption scheme

Substitution Cipher

- Substituting one thing for another
- Monoalphabetic cipher: substitute one letter for another

**plaintext:**    *abcdefghijklmnopqrstuvwxyz*

**ciphertext:**   *mnbvcxzasdfghjklpoiuytrewq*

e.g.:    **Plaintext:** *bob. i love you. alice*  
          **ciphertext:** *nkn. s gktc wky. mgsbc*

Encryption key: mapping from set of 26 letters to set of 26 letters

Breaking an Encryption Scheme

- **Cipher-text only attack**: Trudy has ciphertext she can analyze two approaches:
  - Brute force: search through all keys
  - Statistical analysis
- **known-plaintext attack**: Trudy has plaintext corresponding to ciphertext e.g., in monoalphabetic cipher, Trudy determines pairings for a,i,l,c,e,b,o,
- **chosen-plaintext attack**: Trudy can get ciphertext for chosen plaintext

Symmetric key crypto on Internet

- DES: Data Encryption Standard
- Uses function instead of predetermined tables
  - 56-bit symmetric key, 64-bit plaintext input
  - 3DES: more secure, encrypt 3 times with 3 different keys
- AES: Advanced Encryption Standard
- replaced DES (Nov 2001)
  - 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

Public Key Cryptography

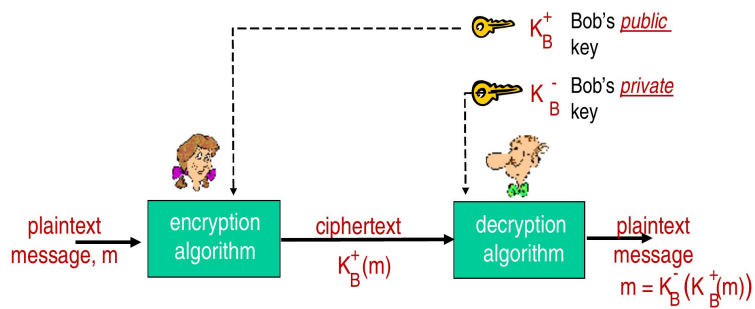
Symmetric key crypto

- requires sender, receiver know **shared secret key**
- Q: how to agree on key in first place (particularly if never "met")?

Public key crypto

- radically different approach
- sender, receiver **do not share secret key**
- **public encryption key** known to all
- **private decryption key** known only to receiver

Public Key cryptography



Public key encryption algorithms

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  $K_B^-$
- A good example: **RSA** (Rivest, Shamir, Adelson) algorithm

RSA: Getting Ready

- **Message: just a bit pattern**
- bit pattern can be uniquely represented by an integer number
- thus, **encrypting a message is equivalent to encrypting a number.**
- example:  
 $m = 10010001$ . This message is uniquely represented by the decimal number 145.
- to encrypt  $m$ , we encrypt the corresponding number, which gives a new number (the ciphertext).

RSA: Creating public/private key pair

- Choose two large prime numbers  $p, q$ . (e.g., 1024 bits each)
- Compute  $n = pq$  and  $z = (p - 1)(q - 1)$
- Choose  $e$  (with  $e < n$ ) that has no common factors with  $z$  ( $e, z$  are "relatively prime").
- Choose  $d$  such that  $ed - 1$  is exactly divisible by  $z$ . (in other words:  $ed \bmod z = 1$ ).
- Public key,  $K_B^+$ , is  $(n, e)$ . Private key,  $K_B^-$ , is  $(n, d)$ .

RSA: encryption, decryption

- Given  $(n, e)$  and  $(n, d)$  as computed above
- To encrypt message  $m (< n)$ , compute
  - $c = m^e \bmod n$
- To decrypt received bit pattern,  $c$ , compute
  - $m = c^d \bmod n$

Magic happens!

$$m = (m^e \bmod n)^d \bmod n$$

RSA example

Encrypting 8-bit messages

- Bob chooses  $p = 5, q = 7$ . Then  $n = 35, z = 24$
- $e = 5$  (so  $e, z$  relatively prime)
- $d = 29$  (so  $ed - 1$  exactly divisible by  $z$ )

encrypt:

bit pattern	$m$	$m^e$	$c = m^e \bmod n$
00001100	12	24832	17

decrypt:

$c$	$c^d$	$m = c^d \bmod n$
17	481968572106750915091411825223071697	12

Why does RSA work? Prerequisite

$x \bmod n =$  **remainder of  $x$  when divide by  $n$**

- Facts:
  - $((a \bmod n) + (b \bmod n)) \bmod n = (a + b) \bmod n$
  - $((a \bmod n) - (b \bmod n)) \bmod n = (a - b) \bmod n$
  - $((a \bmod n) * (b \bmod n)) \bmod n = (a * b) \bmod n$
- thus  $(a \bmod n)^d \bmod n = a^d \bmod n$
- example,  $a = 14, n = 10, d = 2$ :
  - $(a \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  - $a^d = 14^2 = 196$  thus  $a^d \bmod 10 = 6$

Why does RSA work?

Must show that  $c^d \bmod n = m$  where  $c = m^e \bmod n$

- Fact: for any  $x$  and  $y$ :  $x^y \bmod n = x^{(y \bmod z)} \bmod n$  where  $n = pq$  and  $z = (p - 1)(q - 1)$
- Thus,

$$\begin{aligned} c^d \bmod n &= (m^e \bmod n)^d \bmod n \\ &= m^{ed} \bmod n \\ &= m^{(ed \bmod z)} \bmod n \\ &= m^1 \bmod n \\ &= m \end{aligned}$$

RSA: another important property

The following property will be **very** useful later:

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

- Use public key first, followed by private key
- or, user private key first, followed by public key
- give the same result!**

RSA: another important property

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

Follows directly from modular arithmetic:

$$\begin{aligned} (m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{de} \bmod n \\ &= (m^d \bmod n)^e \bmod n \end{aligned}$$

RSA in practice: session keys

Exponentiation in RSA is **computationally intensive**

- DES is at least 100 times faster than RSA
- Use public key crypto to establish secure connection, then establish second key = symmetric session key - for encrypting data
- Example, session key,  $K_S$ 
  - Bob and Alice use RSA to exchange a symmetric key  $K_S$
  - Once both have  $K_S$ , they use symmetric key cryptography

Message Integrity

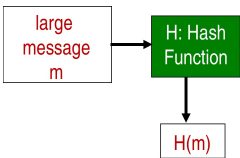
- Allows communicating parties to verify that received messages are authentic.
- Content of message has not been altered
  - Source of message is who/what you think it is
  - Message has not been artificially delayed (playback attack)
  - Sequence of messages is maintained

→ Let's first talk about message digests

Message Digests

- Hash function  $H()$
- Function  $H()$  that takes as input an arbitrary length message and outputs a fixed-length string: "message signature"
  - Note that  $H()$  is a many-to-1 function

- Desirable properties:
- Easy to calculate
  - Irreversibility: Can't determine  $m$  from  $H(m)$
  - Collision resistance: Computationally difficult to produce  $m$  and  $m'$  such that  $H(m) = H(m')$
  - Seemingly random output



Internet checksum: poor message digest

- Internet checksum has some properties of hash function:
- produces fixed length digest (16-bit sum) of input
  - is many-to-one

- But...
- given message with given hash value, it is easy to find another message with same hash value.
  - Example: Simplified checksum: add 4-byte chunks at a time:

message	ASCII format	message	ASCII format
I O U 1	49 4F 55 31	I O U 9	49 4F 55 39
0 0 . 9	30 30 2E 39	0 0 . 1	30 30 2E 31
9 B O B	39 42 D2 42	9 B O B	39 42 D2 42
B2 C1 D2 AC		B2 C1 D2 AC	

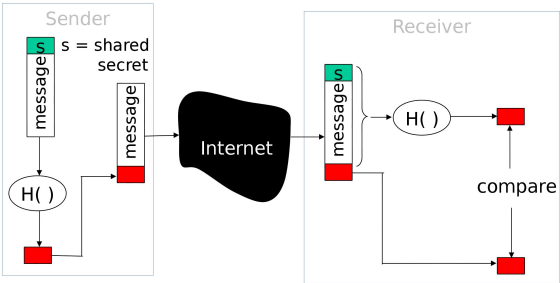
different messages but identical checksums!

Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
- Computes 128-bit message digest in 4-step process.

- SHA-1 is also used
- US standard
  - 160-bit message digest

Message Authentication Code (MAC)



- Authenticates sender
- Verifies message integrity
- No encryption !
- Malicious guys cannot guess shared secret
- Also called "keyed hash"

HMAC: keyed-Hash Message Authentication Code

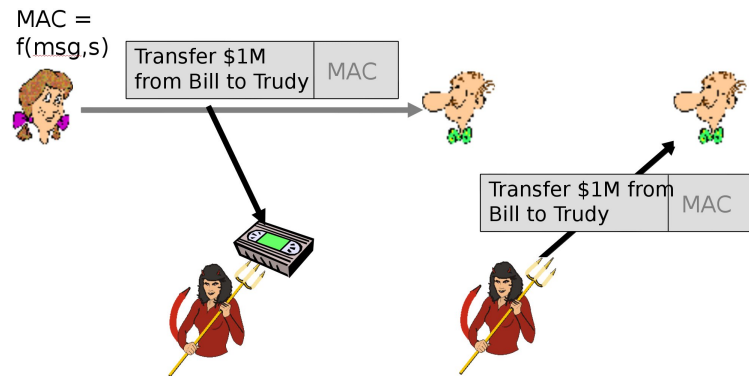
- HMAC: Popular MAC standard
- Addresses some subtle security flaws
    - 1 Concatenates secret to front of message
    - 2 Hashes concatenated message
    - 3 Concatenates the secret to front of digest
    - 4 Hashes the combination again

End-point authentication

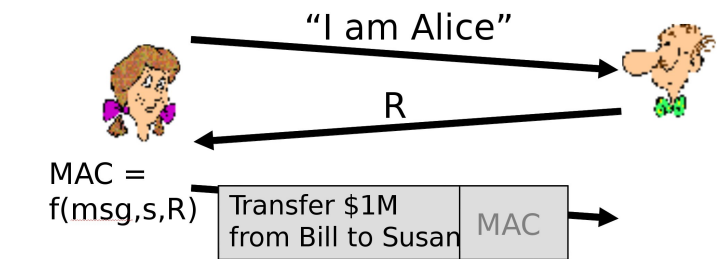
Is message authentication enough?

- Want to be sure of the **originator** of the message
- Assuming Alice and Bob have a shared secret, will MAC provide message authentication.
  - We do know that Alice created the message.
  - But did she send it?

Playback attack



Defending against playback attack: nonce



Nonce: R

- Number used only once in a lifetime
- Ensure that Alice is alive

Digital Signatures

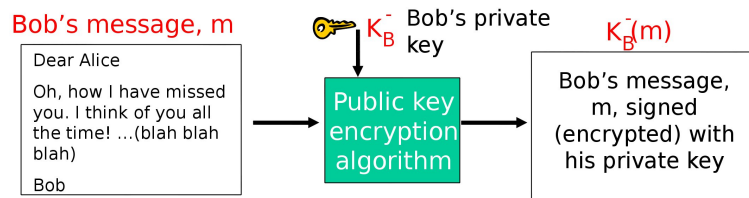
Cryptographic technique analogous to hand-written signatures.

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- Goal is similar to that of a MAC, except now use public-key cryptography
- No need for the 2 peers to have a shared secret
- verifiable, nonforgeable**: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Digital Signatures

Simple digital signature for message m:

- Bob signs m by encrypting with his private key  $K_B^-$ , creating "signed" message,  $K_B^-(m)$

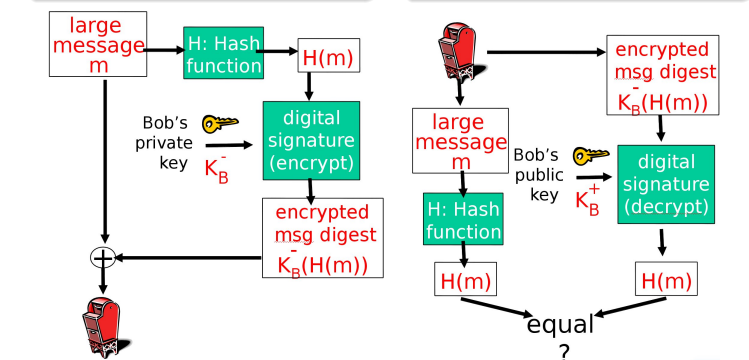


But..

- this is computationally expensive for long messages
- more efficient approach with hash functions

Digital signature = signed message digest

- Bob sends message and the corresponding digital signature
- Alice verifies signature and integrity of digitally signed message



Digital signature

Suppose Alice receives msg  $m$ , digital signature  $K_B^-(H(m))$

- Alice verifies  $m$  signed by Bob by applying Bob's public key  $K_B^+$  to  $K_B^-(H(m))$  then checks  $K_B^+(K_B^-(H(m))) = H(m)$
- If  $K_B^+(K_B^-(H(m))) = H(m)$ , whoever signed  $m$  must have used Bob's private key

Alice thus verifies that:

- Bob signed  $m$
- No one else signed  $m$
- Bob signed  $m$  and not  $m'$

Notes

- Bob is the only one to keep a secret (no secret shared key)
- Note: This time the private key is used to encrypt

Public-key certification

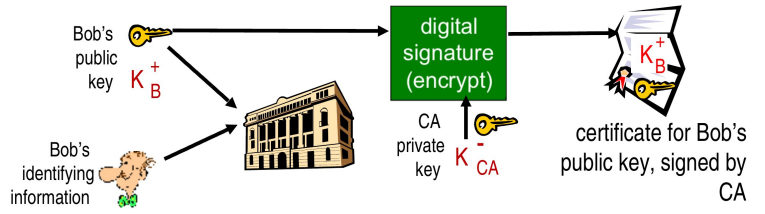
Motivation: Trudy plays pizza prank on Bob

- Trudy creates e-mail order:  
*Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob*
- Trudy signs order with her private key
- Trudy sends order to Pizza Store
- Trudy sends to Pizza Store her public key, but says it's Bob's public key
- Pizza Store verifies signature; then delivers four pizzas to Bob
- Bob doesn't even like Pepperoni

Certification Authorities

Certification authority (CA): binds public key to particular entity, E.

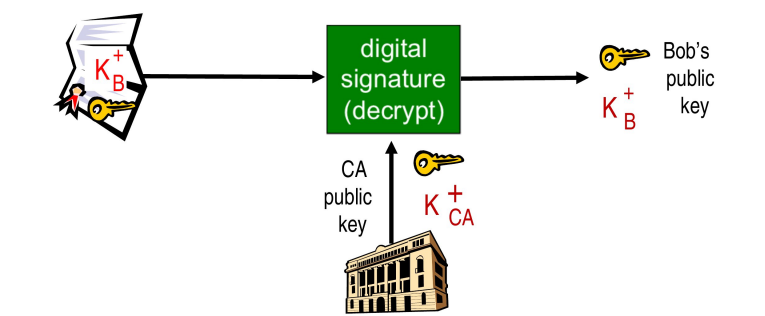
- E (person, router) registers its public key with CA
  - E provides "proof of identity" to CA
  - CA creates certificate binding E to its public key
  - Certificate containing E's public key digitally signed by CA - CA says "this is E's public key"



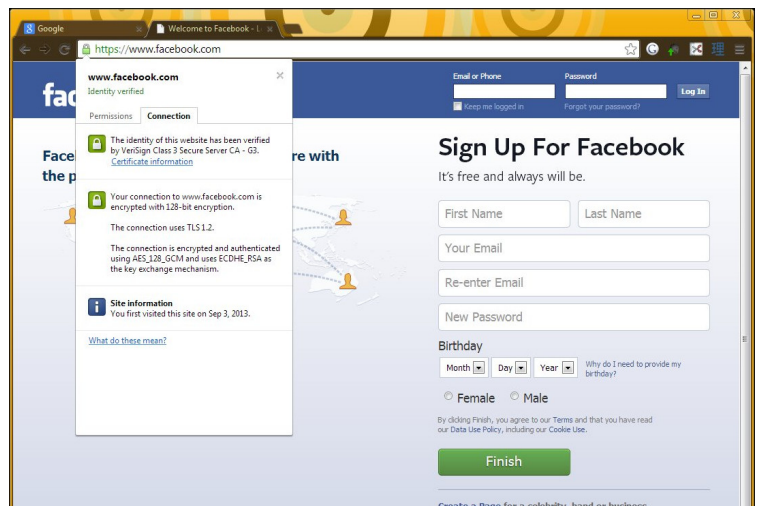
Certification Authorities

When Alice wants Bob's public key:

- gets Bob's certificate (from Bob or elsewhere).
- apply CA's public key to Bob's certificate, get Bob's public key



Example from everyday life



Cryptography: Summary

- Today's lecture covered:
- Basic of cryptography
    - Symmetric key cryptography
    - Public key cryptography
    - Message integrity
    - End-Point Authentication
- Next week: Secured Applications
- Securing email
  - Securing TCP
  - ...

### Today's important points

- Public key
- Public key vs. Symmetric key
- Certificate