

CS/ECE 438: Communication Networks

Fall 2019

7. Network Security

Chapter 7: Network Security

Chapter goals:

- understand principles of network security:
 - cryptography and its *many* uses beyond “confidentiality”
 - authentication
 - message integrity
- security in practice:
 - firewalls and intrusion detection systems
 - security in application, transport, network, link layers

Chapter 7: Outline

- ❑ What is Network Security?
- ❑ Principles of Cryptography
- ❑ Authentication, Message Integrity
- ❑ Securing TCP: SSL/TLS
- ❑ Other Topics: Secure Email, IP, WiFi, & Firewalls.

What is network security?

confidentiality: only sender, intended receiver should “understand” message contents

- sender encrypts message
- receiver decrypts message

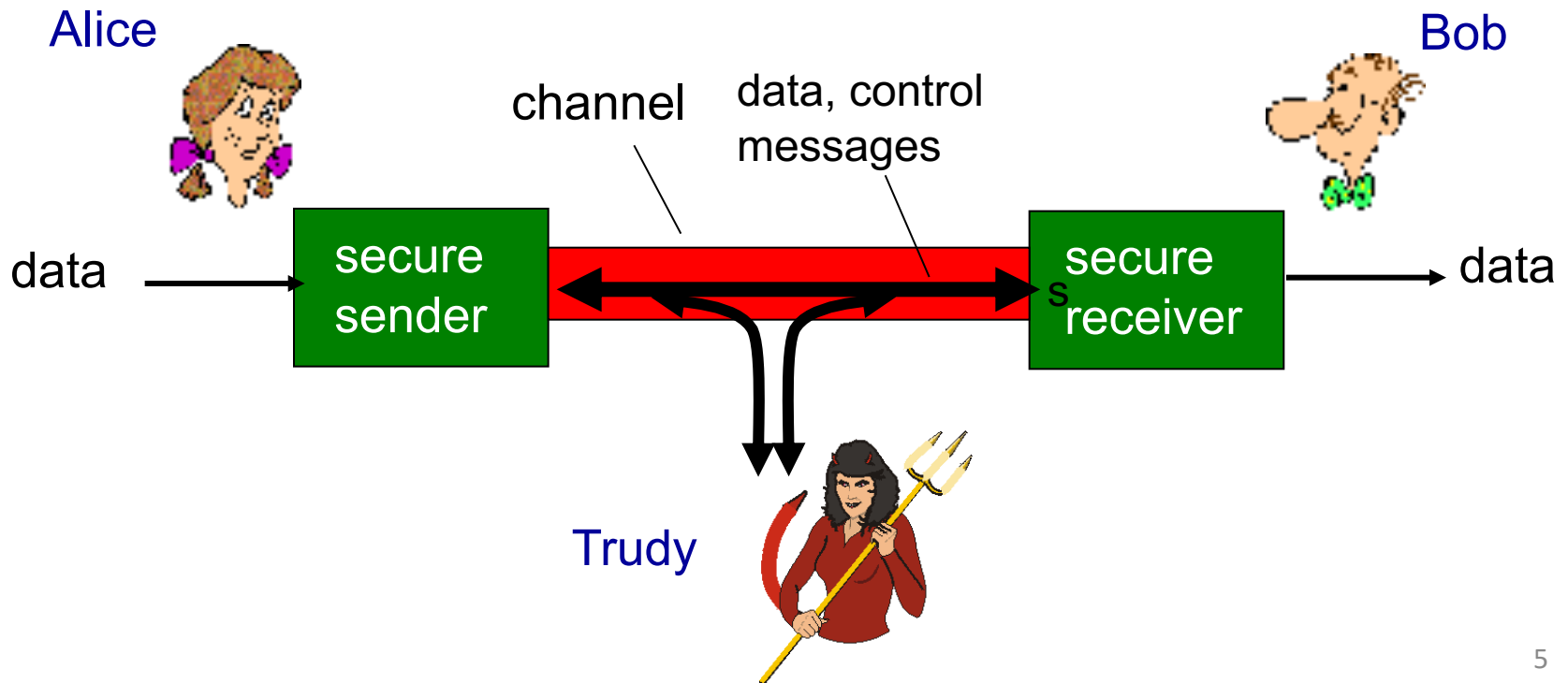
authentication: sender, receiver want to confirm identity of each other

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

Friends and enemies: Alice, Bob, Trudy

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



Who might Bob, Alice be?

- ... 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
- Many other examples!

There are bad guys (and girls) out there!

Q: What can a “bad guy” do?

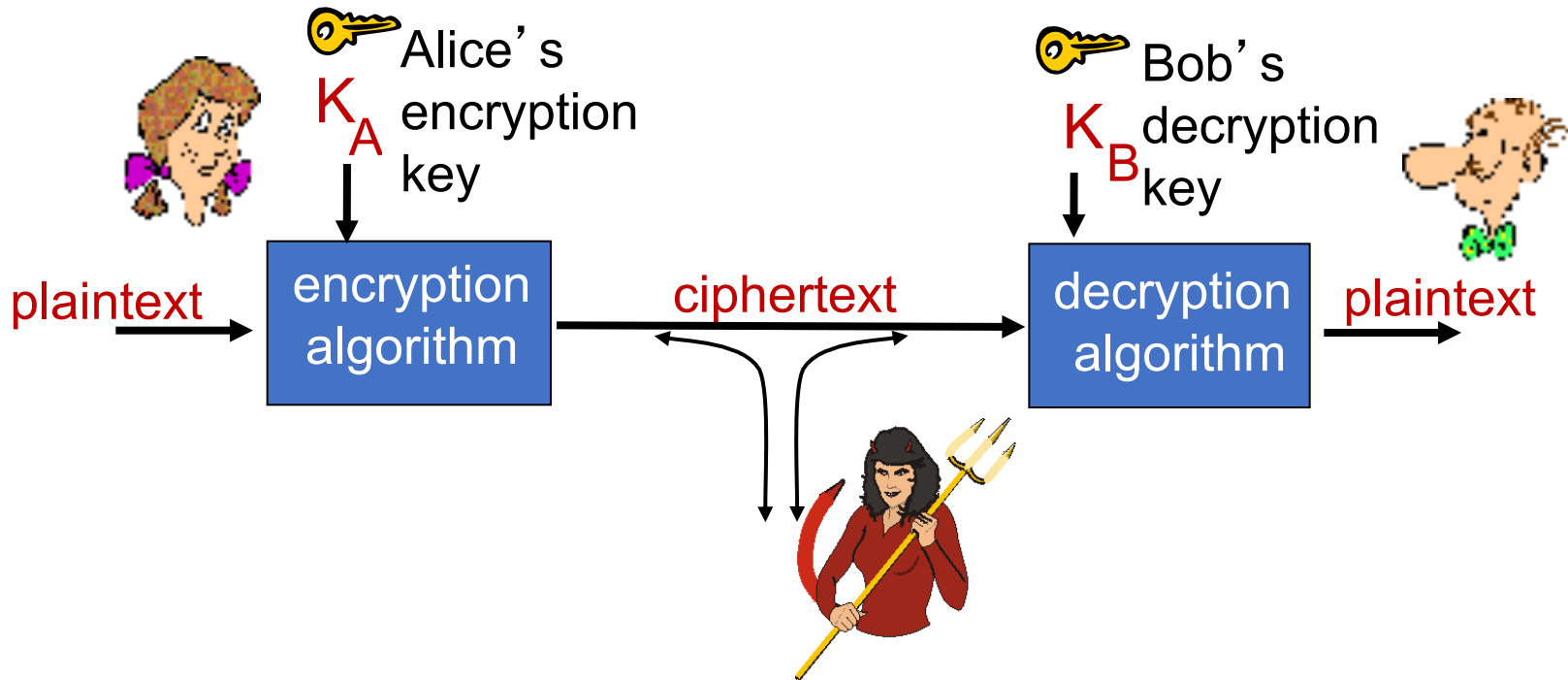
A: A lot!

- *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)

Chapter 7: Outline

- ✓ What is Network Security?
- ❑ Principles of Cryptography
- ❑ Authentication, Message Integrity
- ❑ Securing TCP: SSL/TLS
- ❑ Other Topics: Secure Email, IP, WiFi, & Firewalls.

The language of cryptography

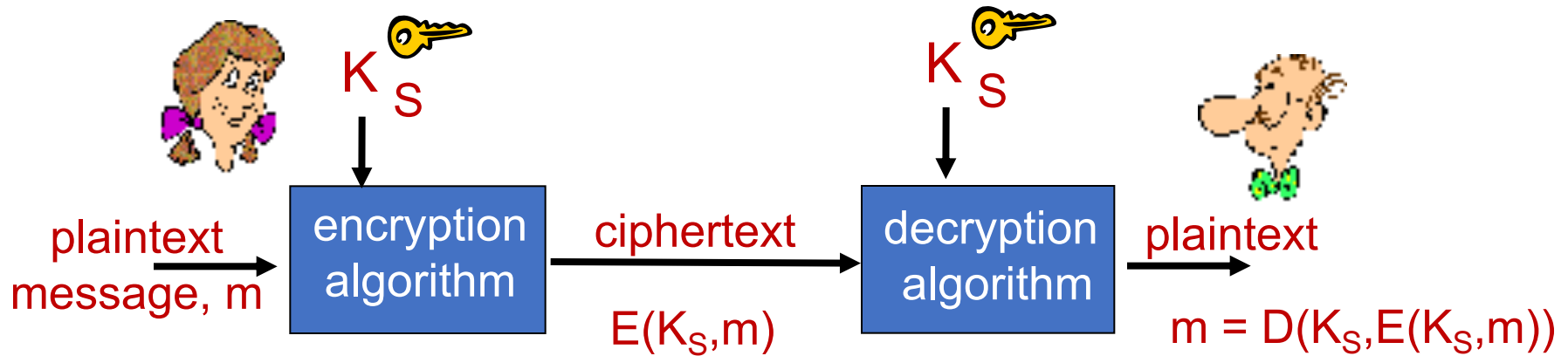


m plaintext message

$E(K_A, m)$ ciphertext, encrypted with key K_A

$m = D(K_B, E(K_A, m))$

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

Symmetric key crypto: DES

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- how secure is DES?
 - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
 - no known good analytic attack
- making DES more secure:
 - 3DES: encrypt 3 times with 3 different keys

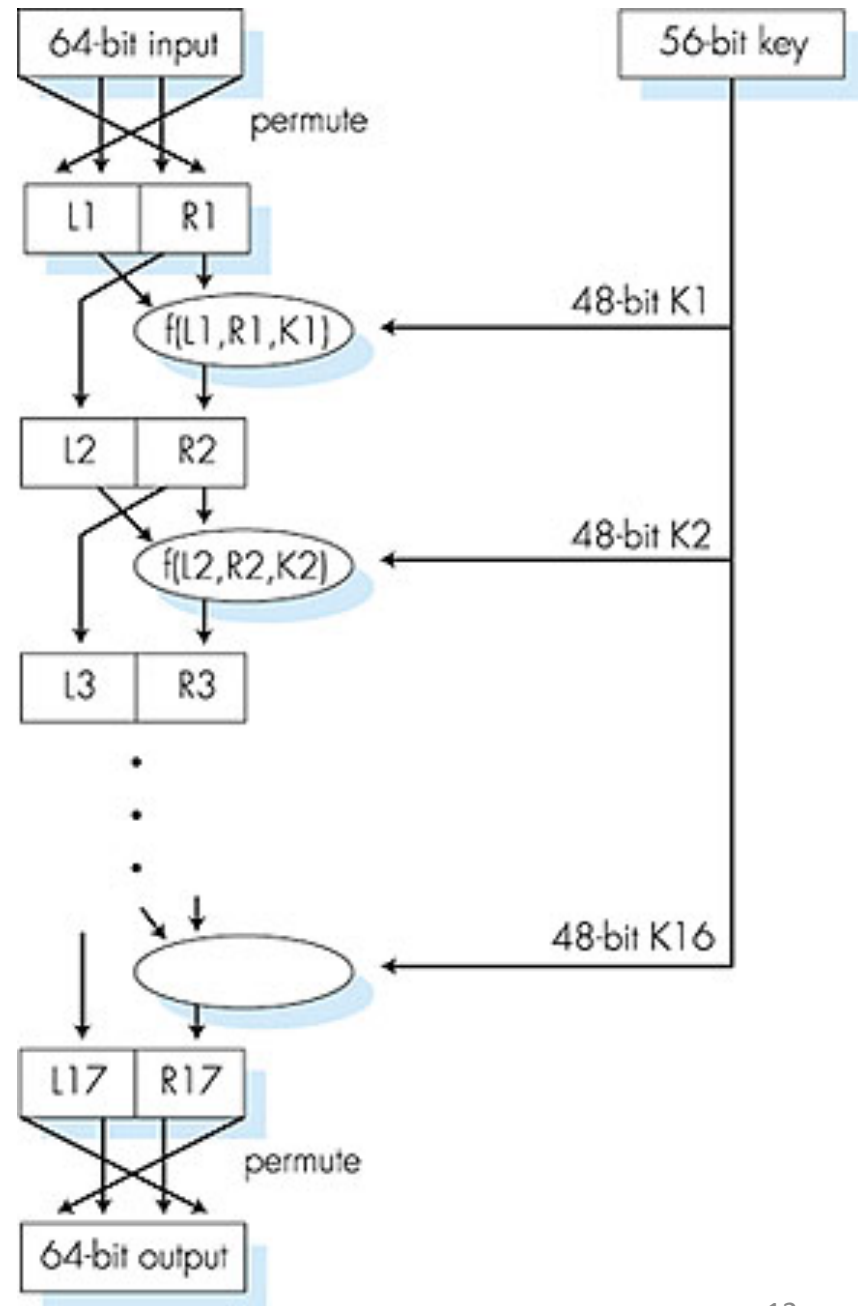
Symmetric key crypto: DES

DES operation

initial permutation

16 identical “rounds” of
function application,
each using different 48
bits of key

final permutation



AES: Advanced Encryption Standard

- symmetric-key NIST 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



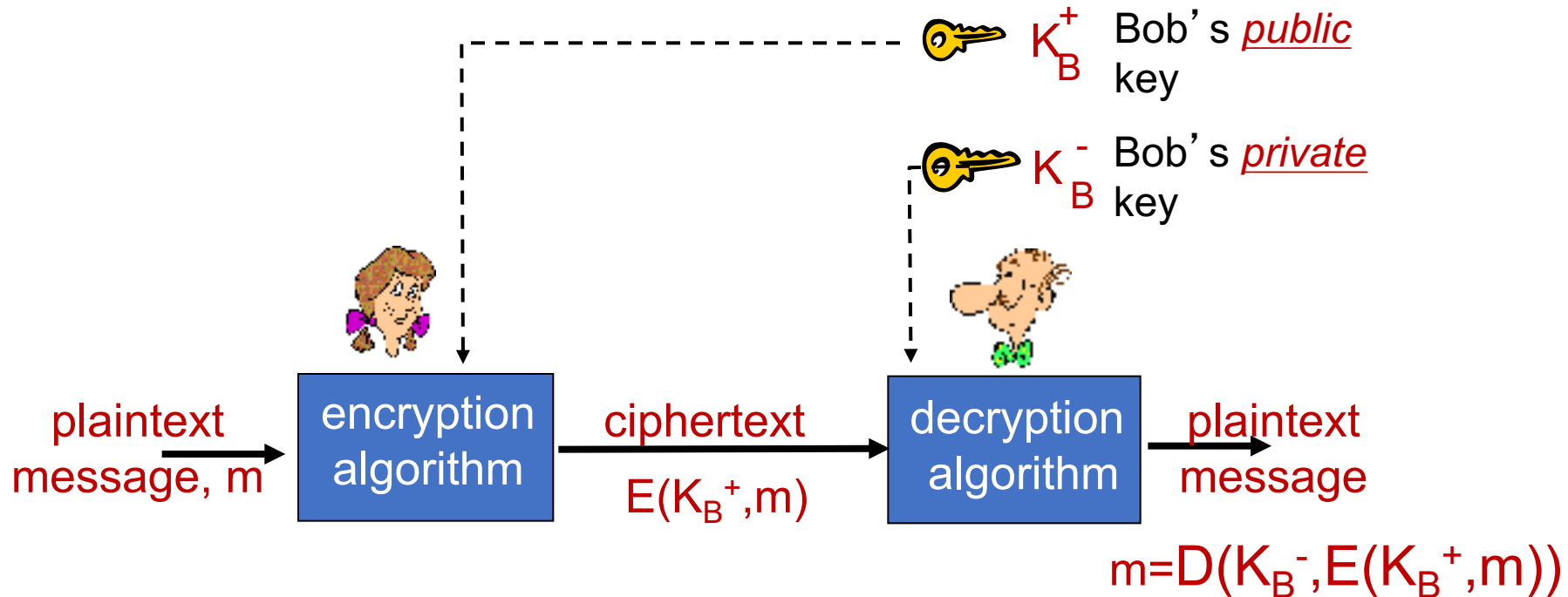
symmetric key crypto

- 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 [Diffie-Hellman76, RSA78]
- 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:

- ① need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

$$D(K_B^-, E(K_B^+, m)) = m$$

- ② given public key K_B^+ , it should be impossible* to compute private key K_B^-

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

1. choose two large prime numbers p, q .
(e.g., 1024 bits each)
2. compute $n = pq$, $z = (p-1)(q-1)$
3. choose e (with $e < n$) that has no common factors with z (e, z are “relatively prime”).
4. choose d such that $ed-1$ is exactly divisible by z .
(in other words: $ed \bmod z = 1$).
5. *public* key is (n, e) . *private* key is (n, d) .

$\underbrace{\hspace{1.5cm}}$
 K_B^+

$\underbrace{\hspace{1.5cm}}$
 K_B^-

RSA: encryption, decryption

0. given (n, e) and (n, d) as computed above

1. to encrypt message $m (< n)$, compute

$$c = m^e \bmod n$$

2. to decrypt received bit pattern, c , compute

$$m = c^d \bmod n$$

*magic
happens!*

$$m = \underbrace{(m^e \bmod n)}_c^d \bmod n$$

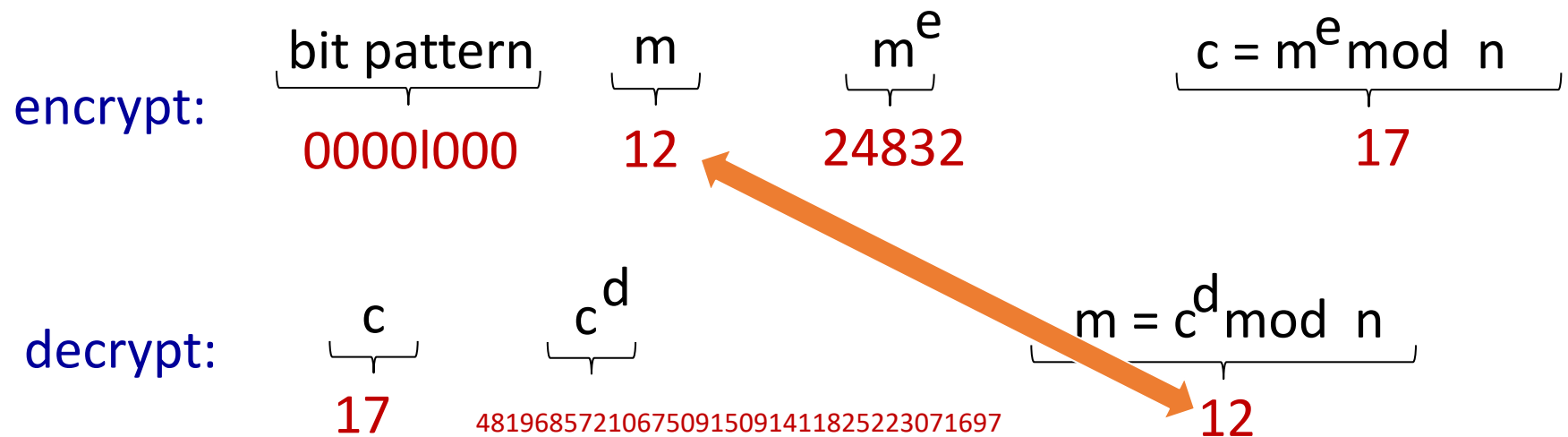
RSA example:

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).

encrypting 8-bit messages.



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

Why is RSA secure?

- suppose you know Bob's public key (n,e) . How hard is it to determine d ?
- essentially need to find factors of n without knowing the two factors p and q
 - fact: factoring a big number is hard

RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^- (K_B^+ (m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+ (K_B^- (m))}_{\text{use private key first, followed by public key}}$$

use public key
first, followed by
private key

use private key
first, followed by
public key

result is the same!

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

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

Chapter 7: Outline

- ✓ What is Network Security?
- ✓ Principles of Cryptography
- Authentication, Message Integrity
- Securing TCP: SSL/TLS
- Other Topics: Secure Email, IP, WiFi, & Firewalls.

Authentication

Goal: Bob wants Alice to “prove” her identity to him

Protocol ap1.0: Alice says “I am Alice”



Failure scenario??



Authentication

Goal: Bob wants Alice to “prove” her identity to him

Protocol ap1.0: Alice says “I am Alice”

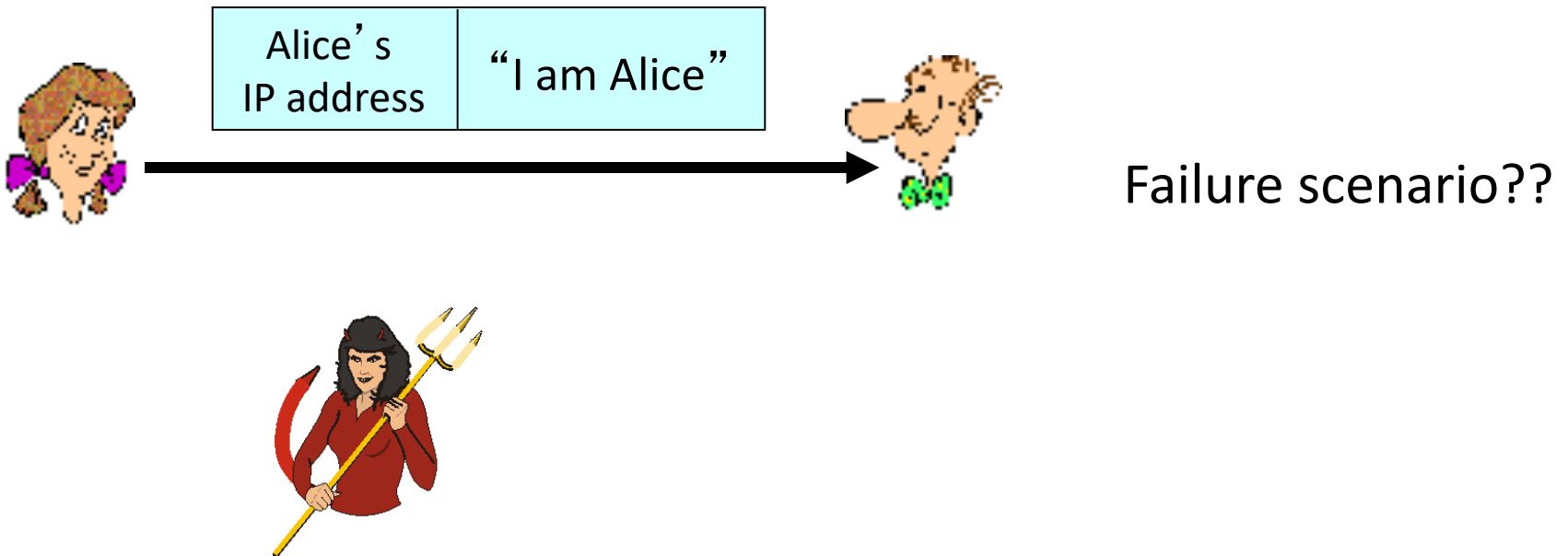


“I am Alice”

in a network,
Bob can not “see” Alice, so
Trudy simply declares
herself to be Alice

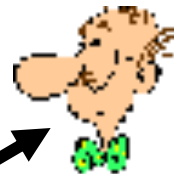
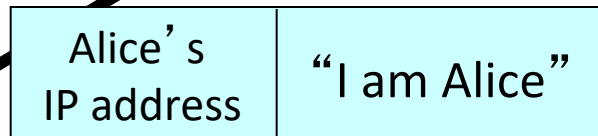
Authentication: another try

Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



Authentication: another try

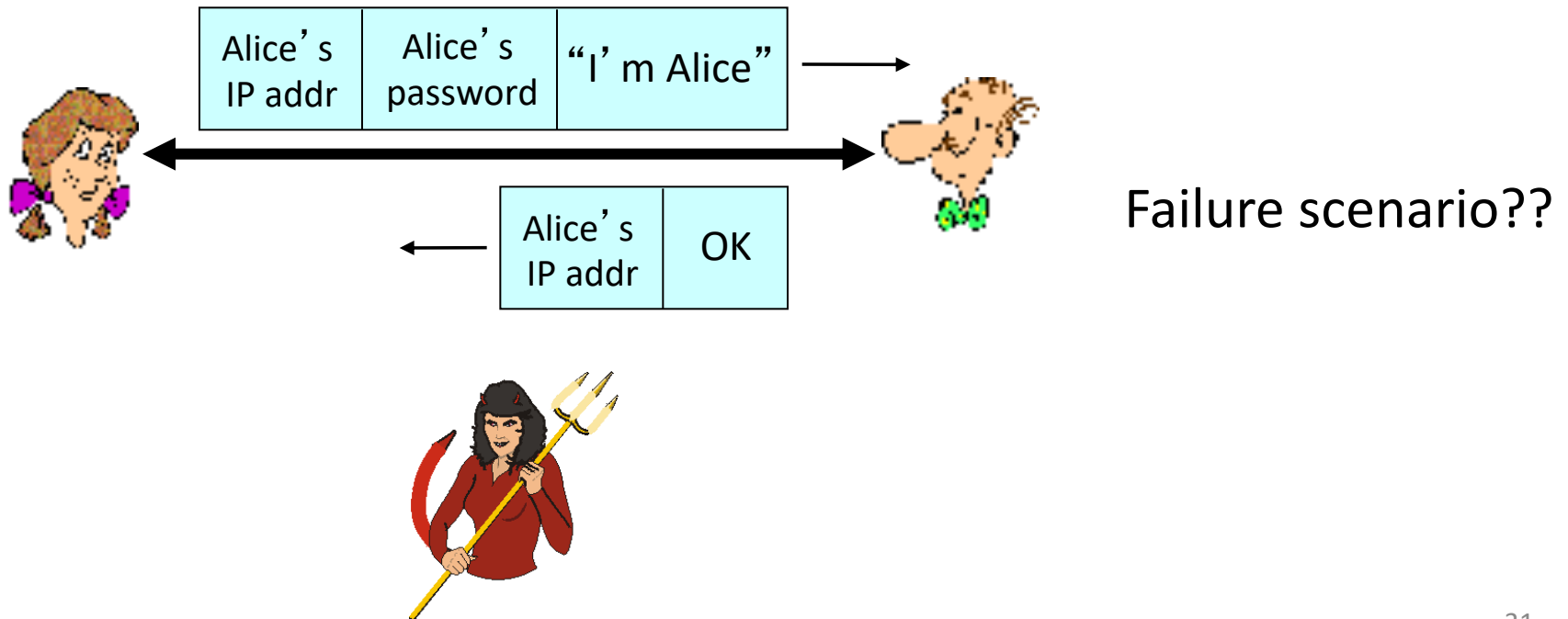
Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



Trudy can create a packet “spoofing” Alice’s address

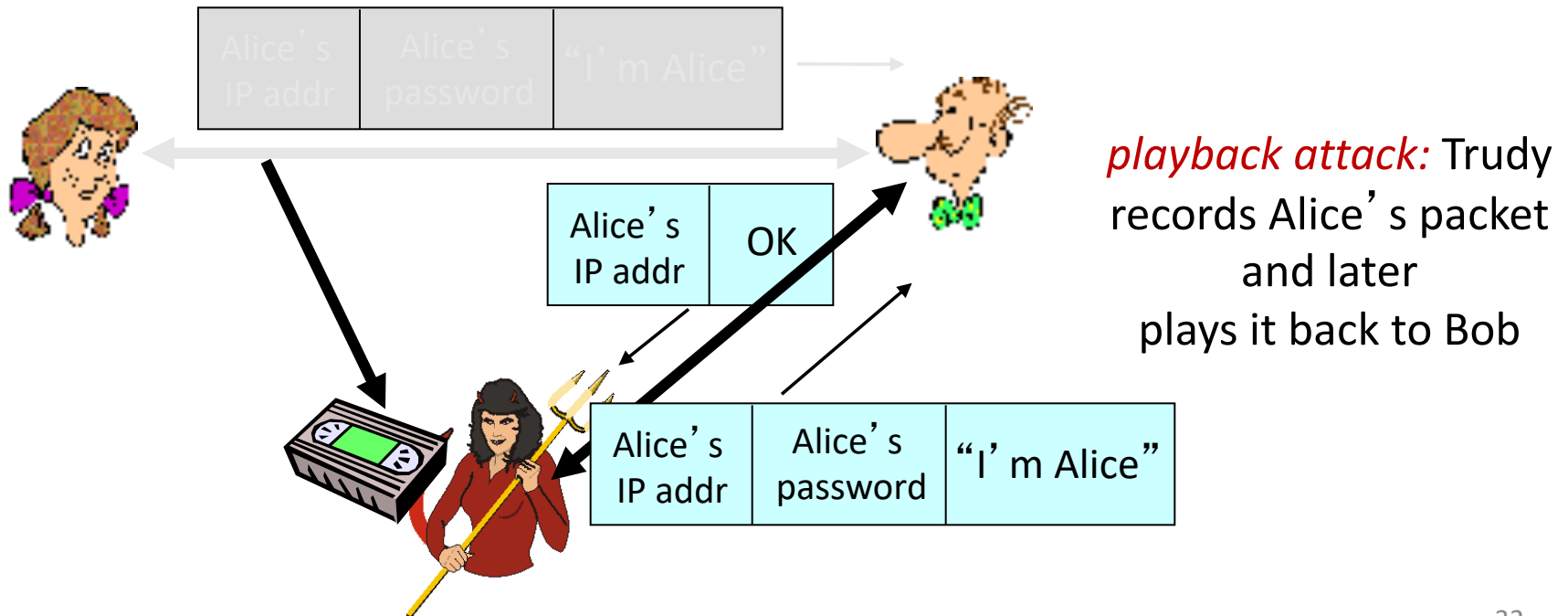
Authentication: another try

Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.



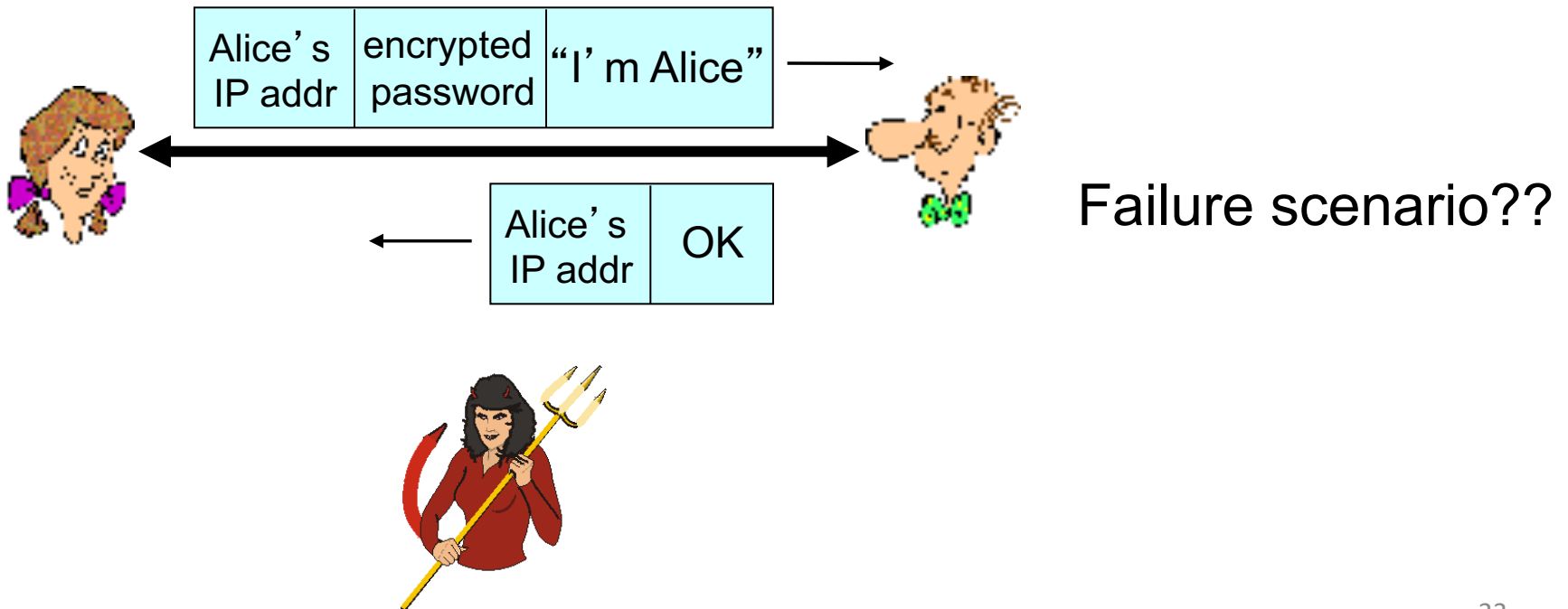
Authentication: another try

Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.



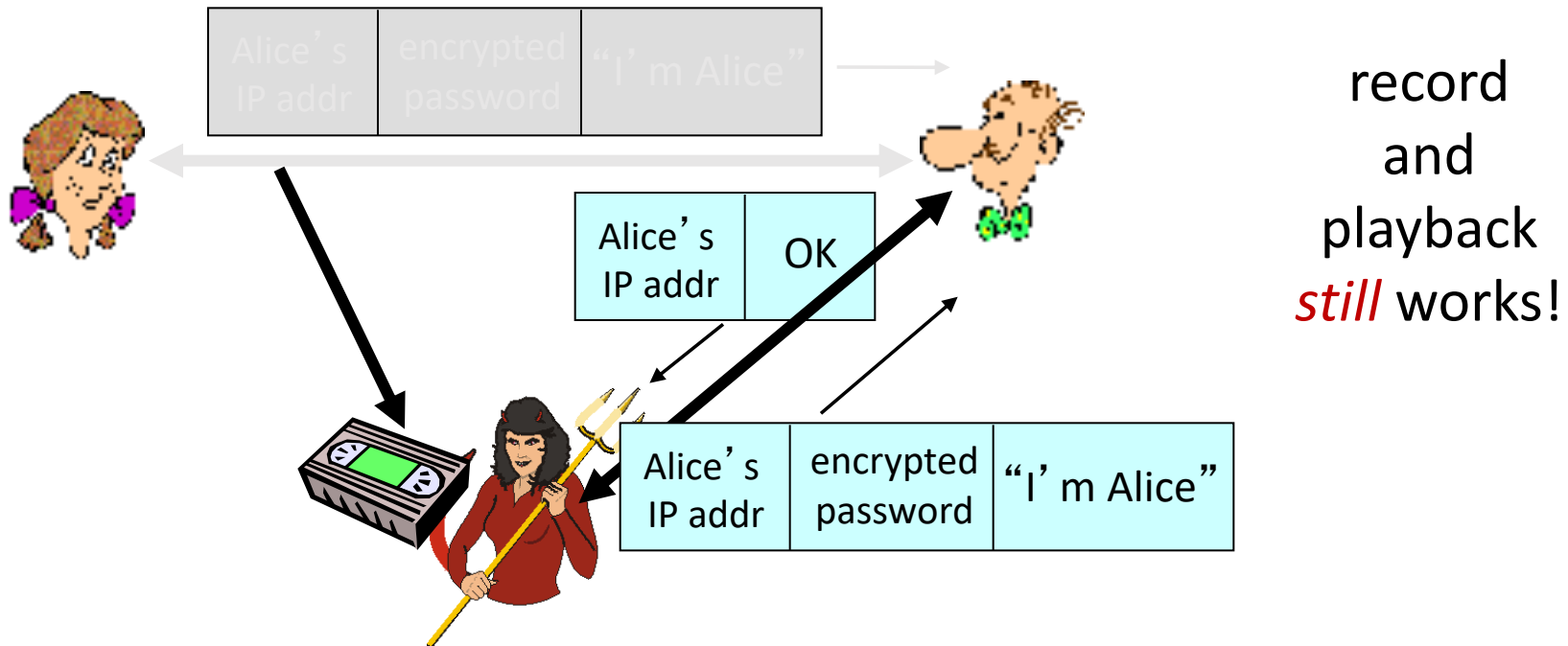
Authentication: yet another try

Protocol ap3.1: Alice says “I am Alice” and sends her *encrypted* secret password to “prove” it.



Authentication: yet another try

Protocol ap3.1: Alice says “I am Alice” and sends her *encrypted* secret password to “prove” it.

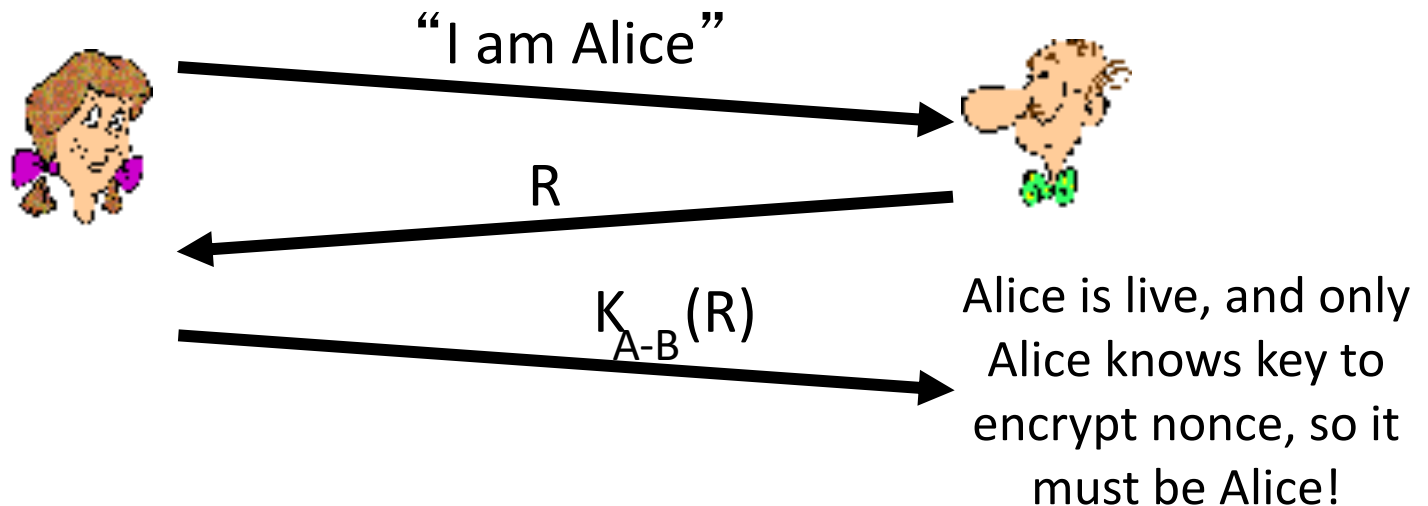


Authentication: yet another try

Goal: avoid playback attack

nonce: number (R) used only *once-in-a-lifetime*

ap4.0: to prove Alice “live”, Bob sends Alice *nonce*, R.
Alice must return R, encrypted with shared secret key



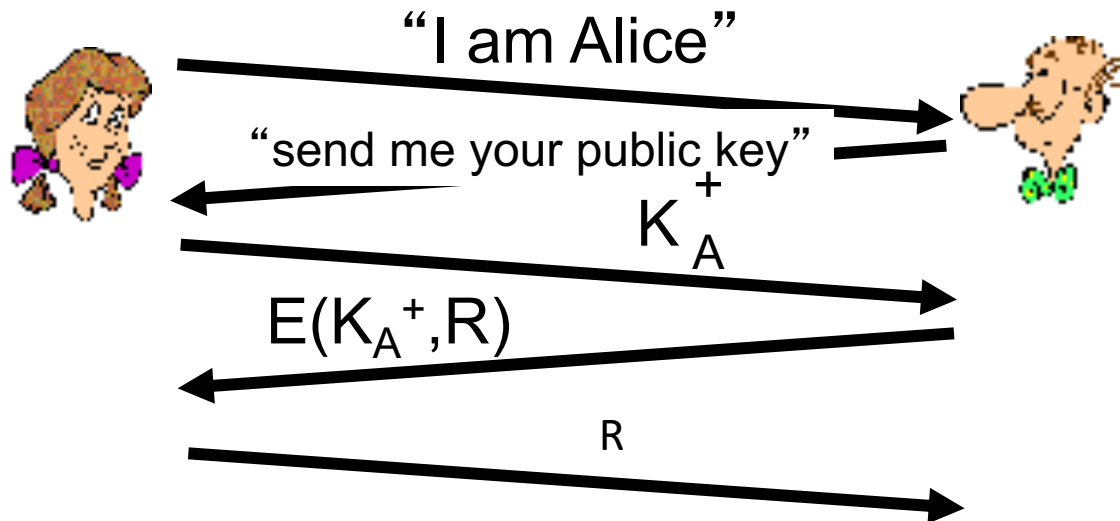
Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

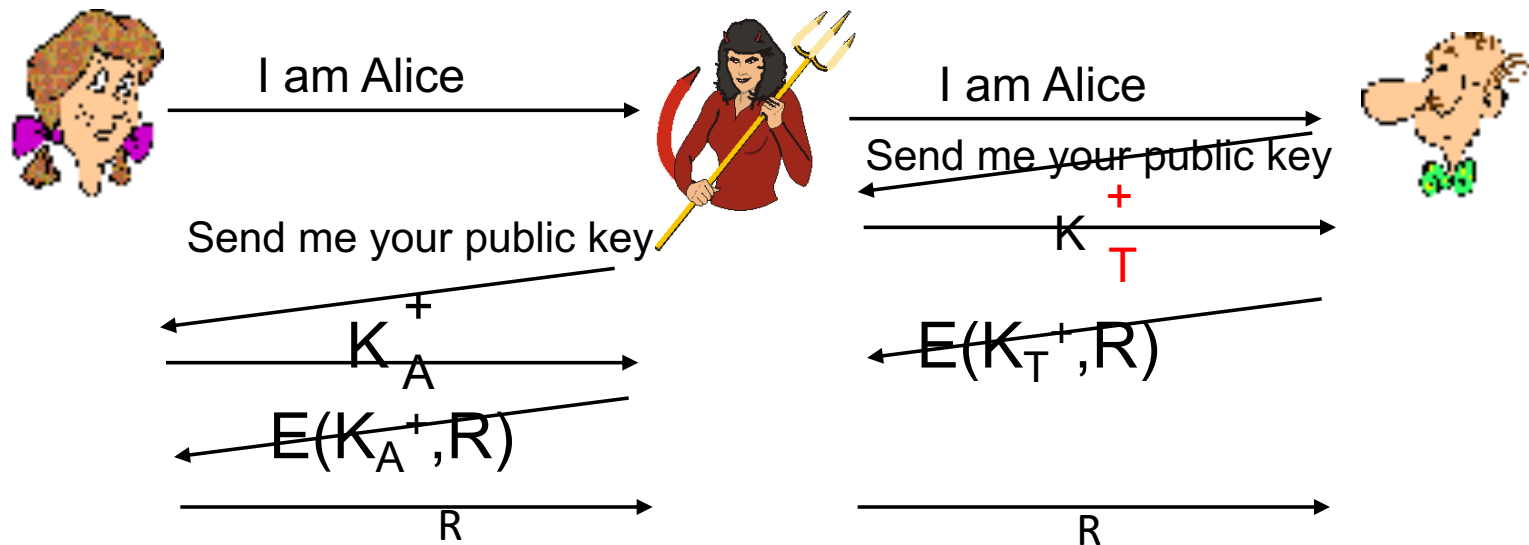
- can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



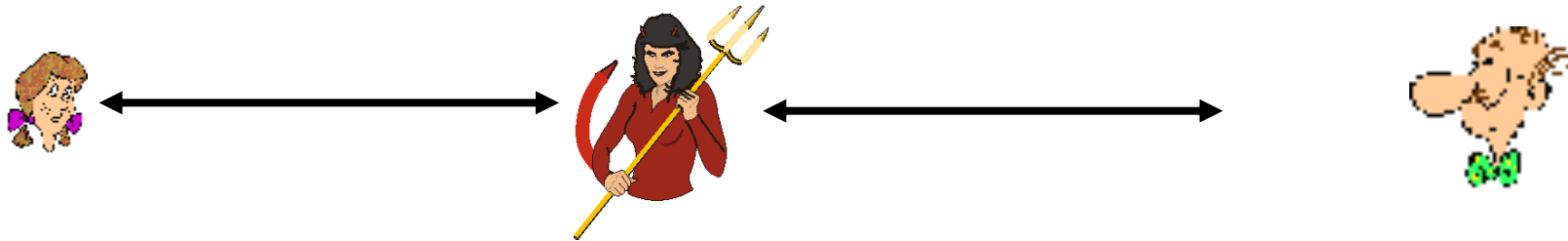
ap5.0: security hole

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



ap5.0: security hole

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



difficult to detect:

- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation!)
- problem is that Trudy receives all messages as well!

Digital signatures

cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- *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 signing with his private key K_B^- , creating signed message, $K_B^-(m)$

Bob's message, m

Dear Alice
Oh, how I have missed
you. I think of you all the
time! ... (blah blah blah)
Bob

 K_B^- Bob's private
key

Public key
signature
algorithm

$m, S(K_B^-, m)$

Bob's message,
 m , signed with his
private key

Digital signatures

- suppose Alice receives msg m , with signature: $m, S(K_B^-, m)$
- Alice verifies m signed by Bob by using Bob's public key K_B^+ to verify $V(K_B^+, S(K_B^-, m), m) = \text{True}$

Alice thus verifies that:

- Bob signed m
- no one else signed m
- Bob signed m and not m'

non-repudiation:

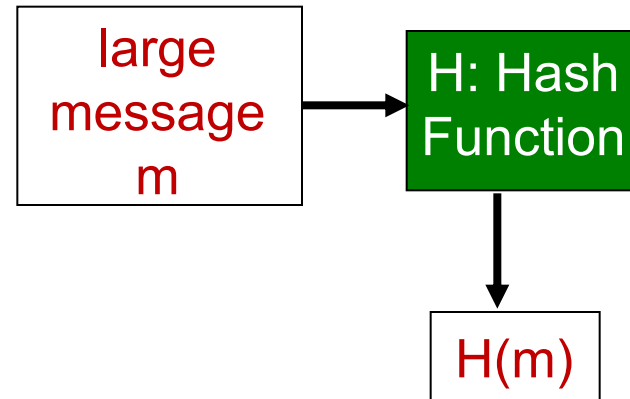
- ✓ Alice can take m , and signature $K_B^-(m)$ to court and prove that Bob signed m

Message digests

computationally expensive
to public-key-encrypt long
messages

goal: fixed-length, easy-
to-compute digital
“fingerprint”

- apply hash function H to m ,
get fixed size message digest,
 $H(m)$.



Hash function properties:

- many-to-1
- produces fixed-size msg
digest (fingerprint)
- given message digest x ,
computationally infeasible
to find m such that $x = H(m)$

Internet checksum: poor crypto hash function

Internet checksum has some properties of hash function:

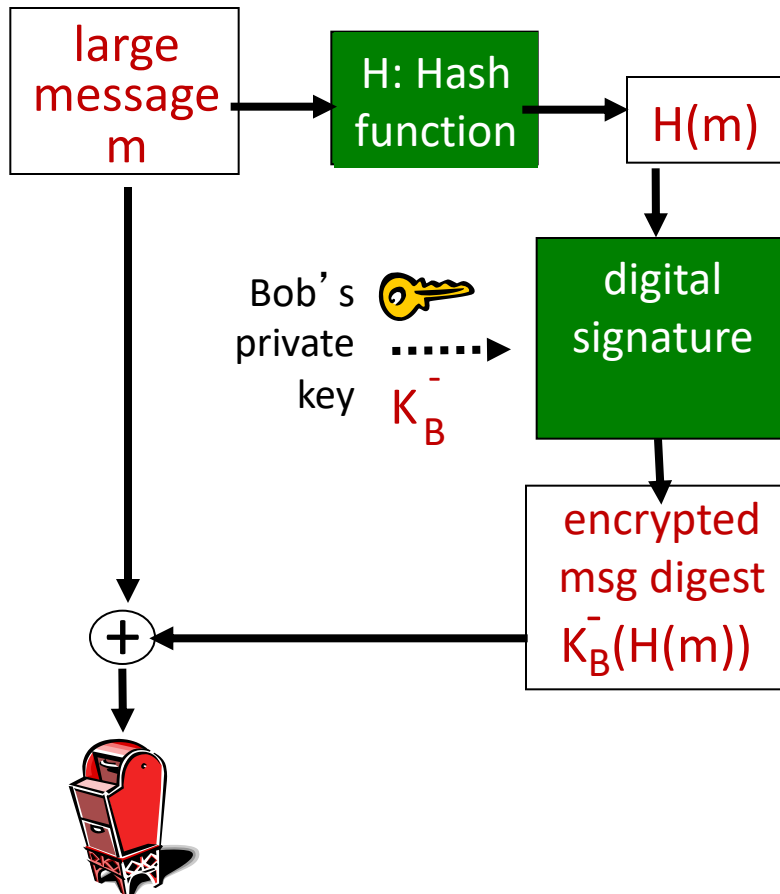
- produces fixed length digest (16-bit sum) of message
- is many-to-one

But given message with given hash value, it is easy to find another message with same hash value:

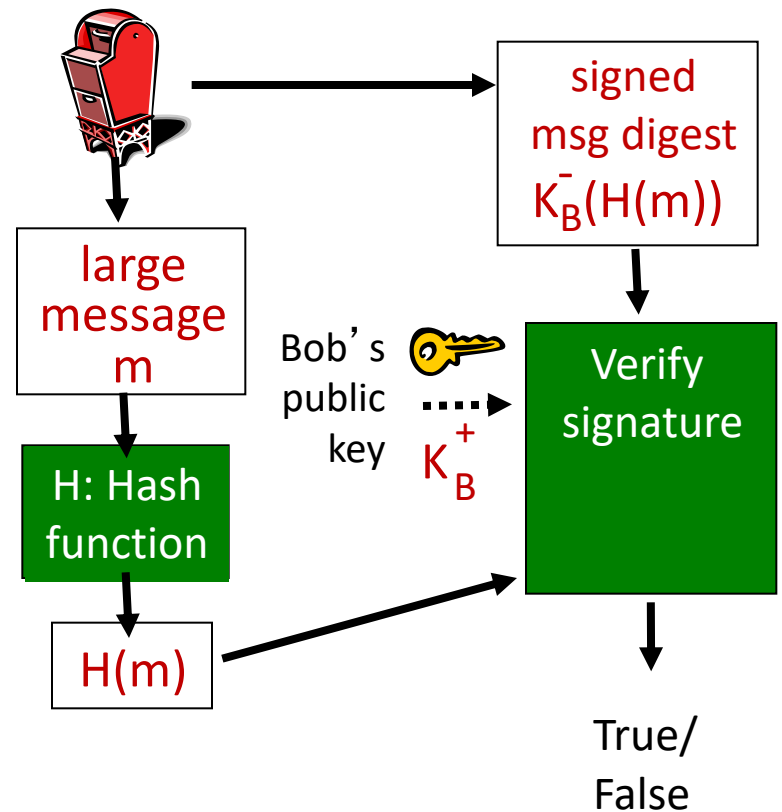
<u>message</u>	<u>ASCII format</u>		<u>message</u>	<u>ASCII format</u>
I O U 1	49 4F 55 31		I O U <u>9</u>	49 4F 55 <u>39</u>
0 0 . 9	30 30 2E 39		0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	39 42 D2 42		9 B O B	39 42 D2 42
	<u>B2 C1 D2 AC</u>	different messages but identical checksums!		<u>B2 C1 D2 AC</u>

Digital signature = signed message digest

Bob sends digitally signed message:

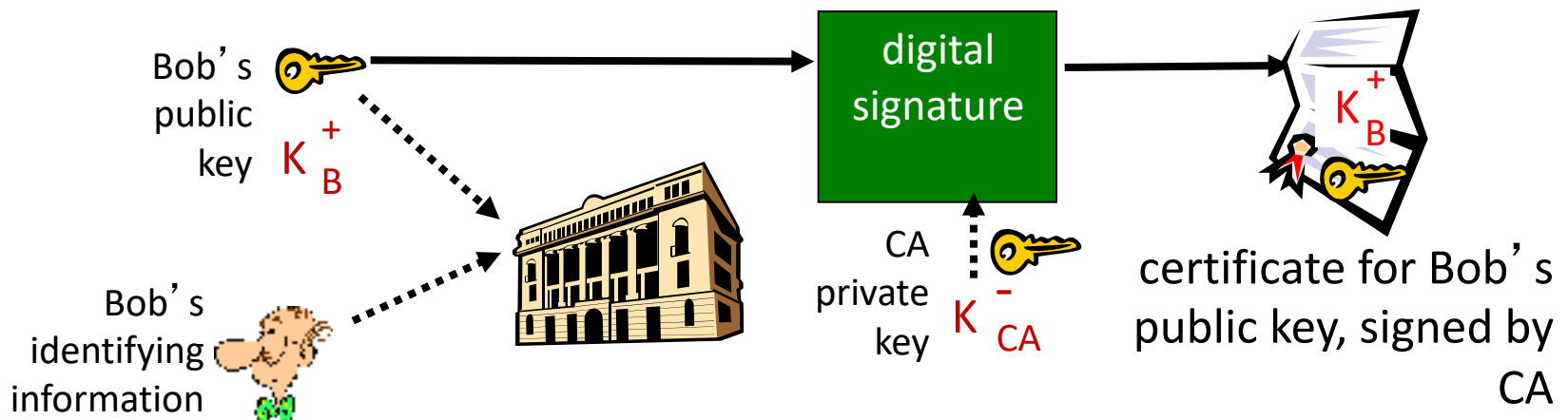


Alice verifies signature, integrity of digitally signed message:



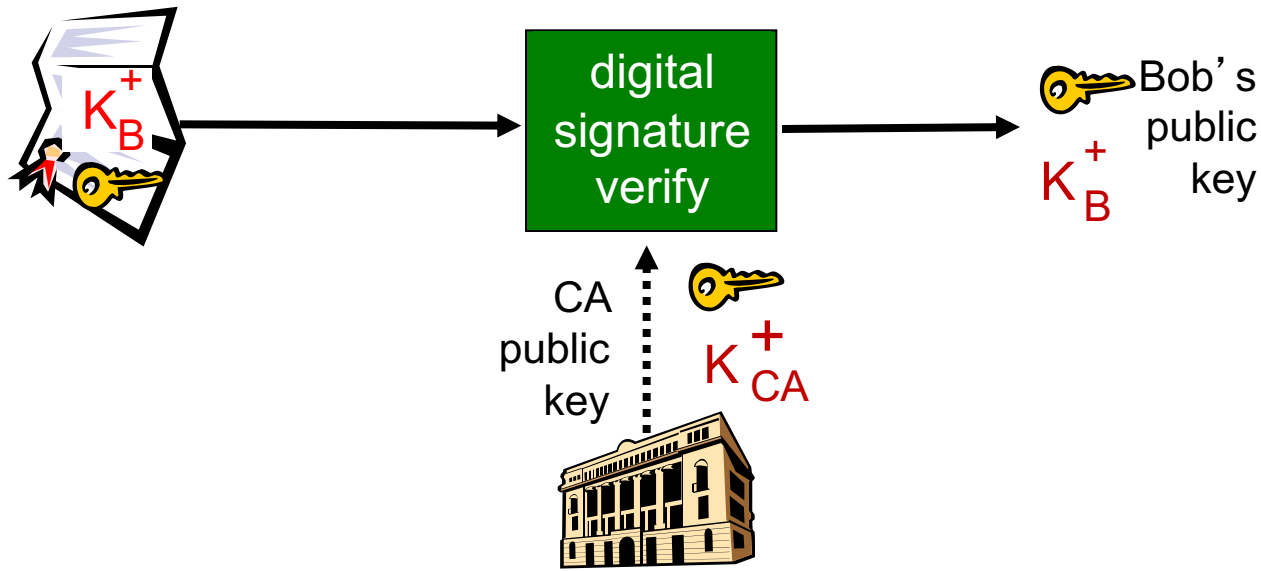
Certification authorities

- *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 (Bob or elsewhere).
 - apply CA's public key to verify Bob's certificate
 - get Bob's public key



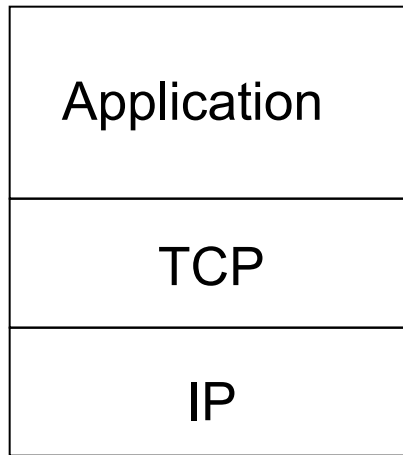
Chapter 7: Outline

- ✓ What is Network Security?
- ✓ Principles of Cryptography
- ✓ Authentication, Message Integrity
- ❑ Securing TCP: SSL/TLS
- ❑ Other Topics: Secure Email, IP, WiFi, & Firewalls.

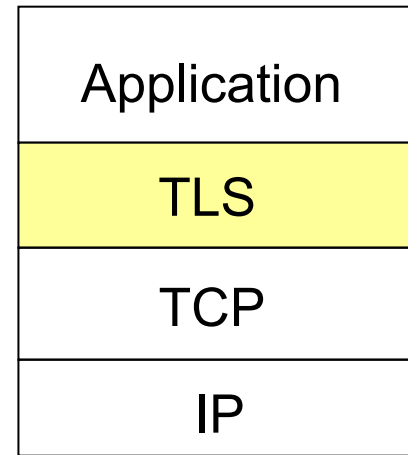
SSL/TLS: Transport Layer Security

- widely deployed security protocol
 - supported by almost all browsers, web servers
 - https
 - billions \$/year over TLS
- mechanisms: [Woo 1994], implementation: Netscape
- Current version: TLS1.2
 - TLS1.3 (aka TLS2 aka TLS4 aka TLS7 on the horizon)
- provides
 - *confidentiality*
 - *integrity*
 - *authentication*
- original goals:
 - Web e-commerce transactions
 - encryption (especially credit-card numbers)
 - Web-server authentication
 - optional client authentication
 - minimum hassle in doing business with new merchant
- available to all TCP applications
 - secure socket interface

TLS and TCP/IP



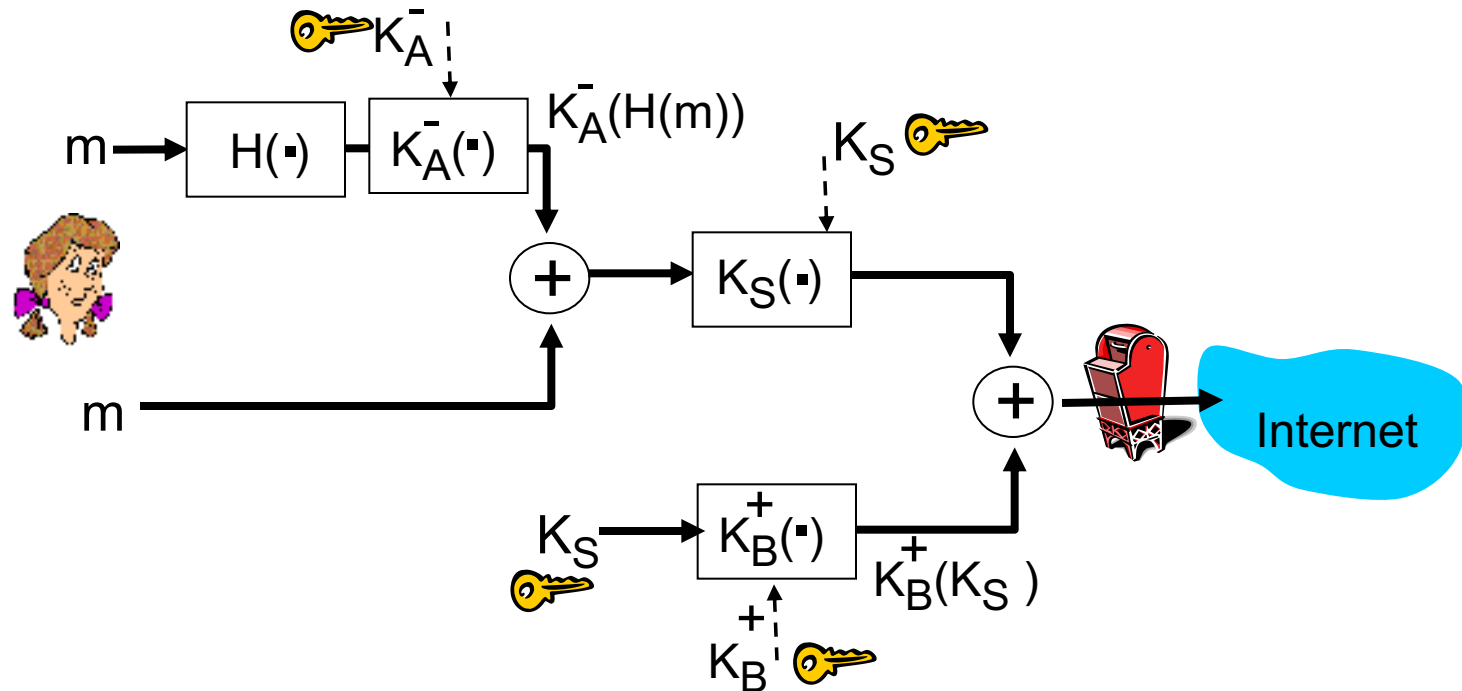
normal application



application with TLS

- TLS provides application programming interface (API) to applications
- C and Java TLS libraries/classes readily available

Could do something like PGP:



- but want to send byte streams & interactive data
- want set of secret keys for entire connection
- want certificate exchange as part of protocol: handshake phase

Toy TLS: a simple secure channel

- *handshake*: Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- *key derivation*: Alice and Bob use shared secret to derive set of keys
- *data transfer*: data to be transferred is broken up into series of records
- *connection closure*: special messages to securely close connection

Chapter 7: Outline

- ✓ What is Network Security?
- ✓ Principles of Cryptography
- ✓ Authentication, Message Integrity
- ✓ Securing TCP: SSL/TLS
- ❑ Other Topics: Email, IP, VPNs, WiFi, & Firewalls.

Chapter 7: Outline

- ✓ What is Network Security?
- ✓ Principles of Cryptography
- ✓ Authentication, Message Integrity
- ✓ Securing TCP: SSL/TLS
- ✓ Other Topics: Secure Email, IP, WiFi, & Firewalls.

Network Security (summary)

basic techniques.....

- cryptography (symmetric and public)
- message integrity
- end-point authentication

.... used in many different security scenarios

- secure email
- secure transport (SSL)

operational security: firewalls and IDS