Lecture 24 Security
Authentication, TLS/SSL

COMP 332, Fall 2018

Victoria Manfredi

WESLEYAN
UNIVERSITY

Acknowledgements: materials adapted from Computer Networking: A Top Down Approach 7th edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved as well as from slides by Abraham Matta at Boston University, and some material from Computer Networks by Tannenbaum and Wetherall.

Today

1. Announcements
   - hw9 due next Wed. at 11:59p
     - no programming part ☺
   - all homework must be turned in by last day of classes!

2. Network security
   - authentication
   - message integrity

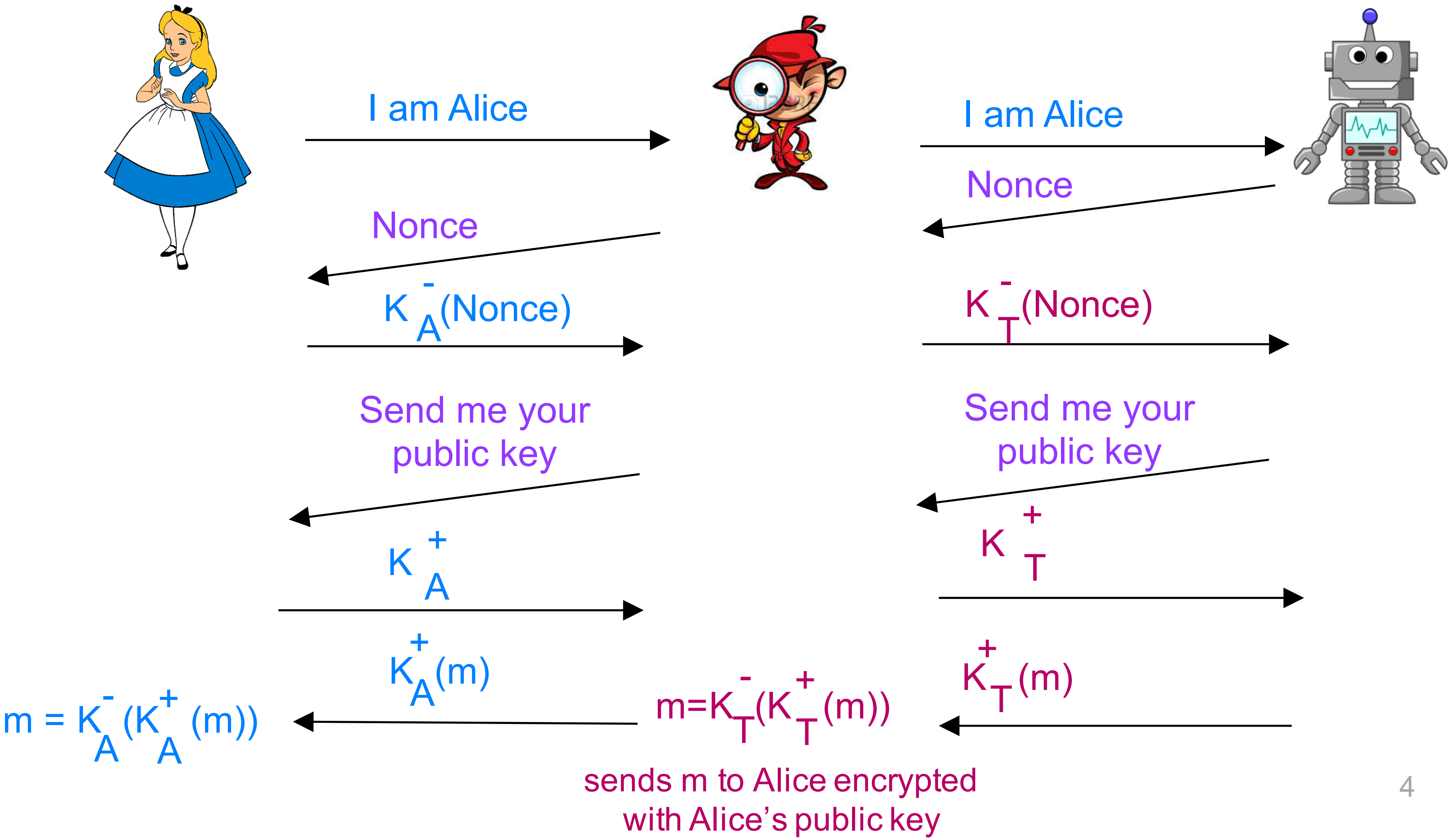
3. Transport layer security
   - overview
   - toy TLS
   - real TLS

Network Security

AUTHENTICATION

Recall: ap5.0 man-in-the-middle attack

Trudy poses as Alice (to Bob) and as Bob (to Alice)



Distinguishing Alice's vs. Trudy's public key

Use certification authority (CA)

- binds public key to particular entity
  - e.g., Alice, Bob, website, …
- 100s of certification authorities

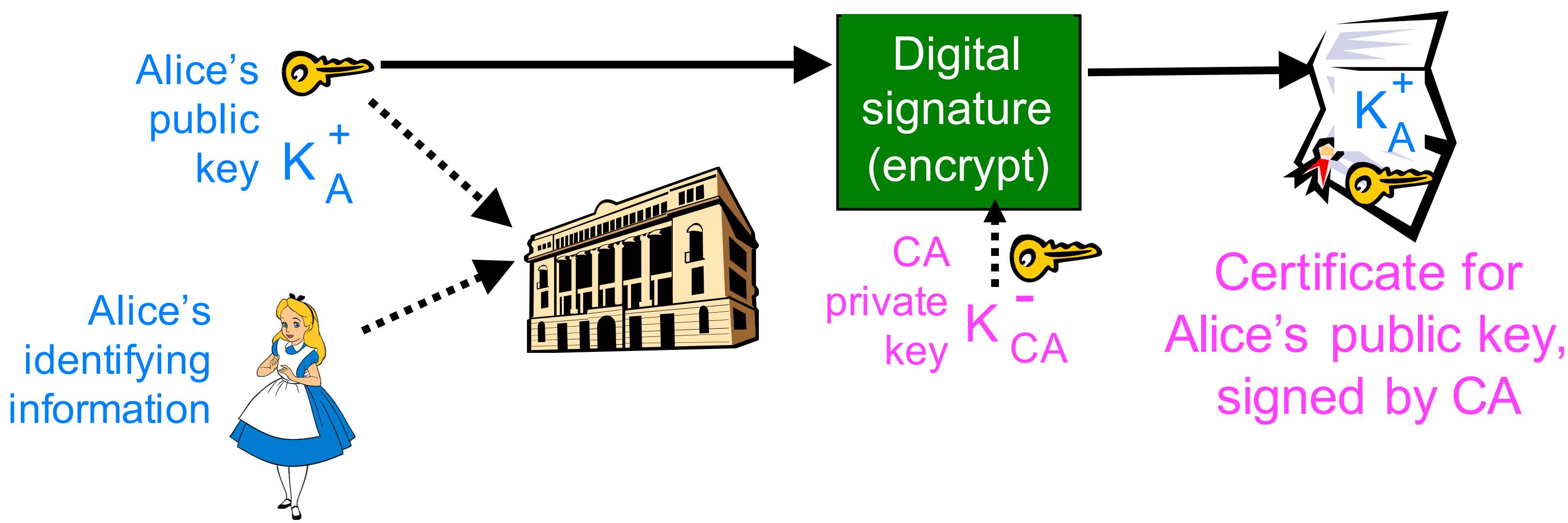
Aside

- CAs are critical but potentially weak link …

How certification authorities work

Alice registers her public key with CA

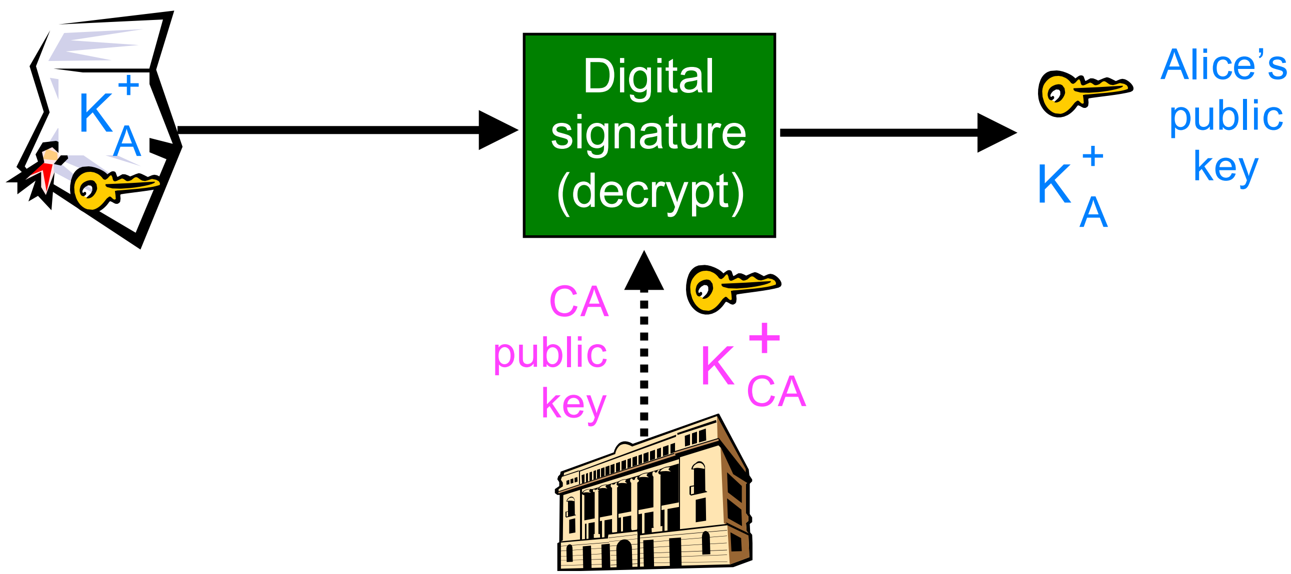
- Alice provides proof of identity to CA
- CA creates certificate binding Alice to its public key
  - certificate digitally signed by CA



Certification authorities

When Bob wants Alice's public key

1. gets Alice's certificate from Alice or elsewhere
2. applies CA's public key to Alice's certificate
3. gets Alice's public key



Example

VeriSign Class 3 Public Primary Certification Authority - G5
↳ Symantec Class 3 EV SSL CA - G3
↳ www.bankofamerica.com

www.bankofamerica.com
Issued by: Symantec Class 3 EV SSL CA - G3
Expires: Thursday, July 26, 2018 at 7:59:59 PM Eastern Daylight Time
This certificate is valid

Details

Subject Name	
Inc. Country	US
Inc. State/Province	Delaware
Business Category	Private Organization
Serial Number	2927442
Country	US
Postal Code	60603
State/Province	Illinois
Locality	Chicago
Street Address	135 S La Salle St
Organization	Bank of America Corporation
Organizational Unit	eComm Network Infrastructure
Common Name	www.bankofamerica.com
Issuer Name	
Country	US
Organization	Symantec Corporation
Organizational Unit	Symantec Trust Network
Common Name	Symantec Class 3 EV SSL CA - G3
Serial Number	4E 49 91 F1 B7 6A 9D 8D 16 23 5F 38 81 DD F5 E1
Version	3
Signature Algorithm	SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
Parameters	none
Not Valid Before	Monday, July 24, 2017 at 8:00:00 PM Eastern Daylight Time
Not Valid After	Thursday, July 26, 2018 at 7:59:59 PM Eastern Daylight Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)

Network Security

MESSAGE INTEGRITY

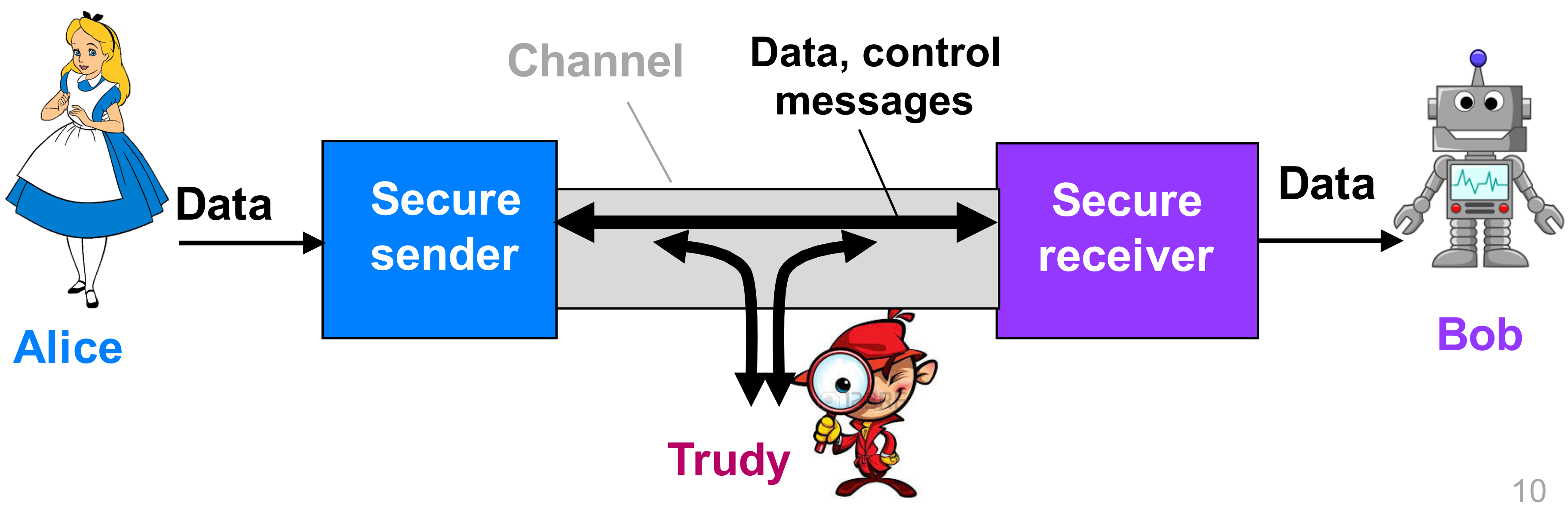
Message integrity

Alice and Bob must be able to detect whether msg changed

1. verify msg originated from Alice
2. verify msg not tampered with on way to Bob

Solution

– digital signatures: cryptographic technique like hand-written signature



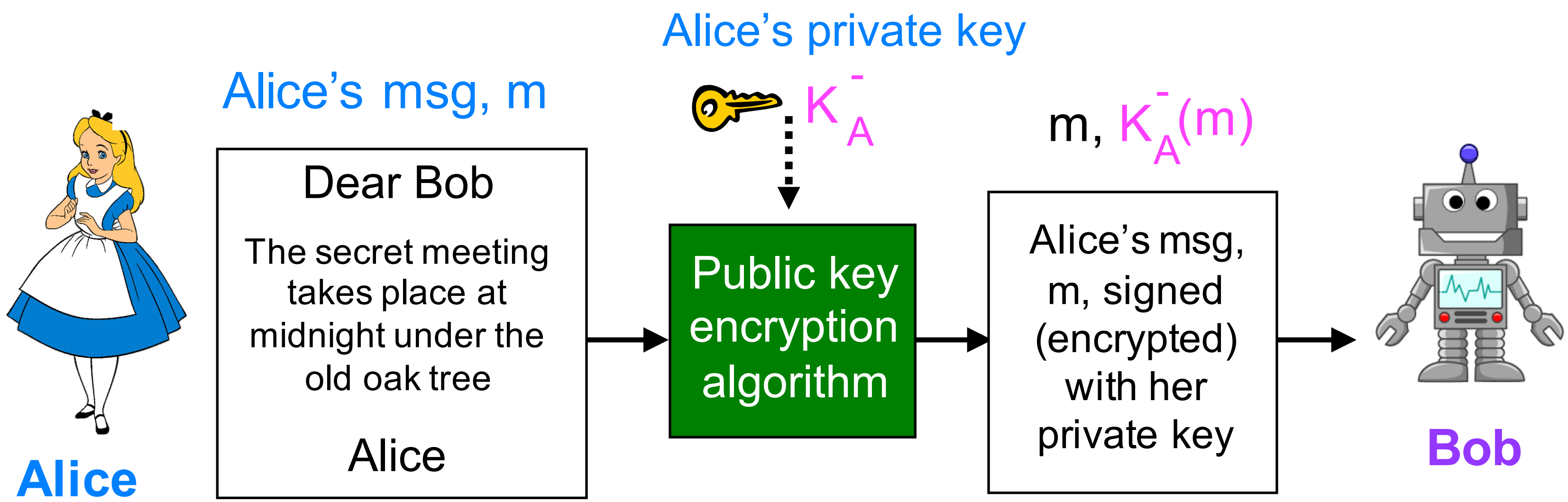
Simple digital signature for message, m

Sender (Alice)

- encrypts msg m with her private key K_A^- to create signed message, $K_A^-(m)$
- proves she is owner/creator

Recipient (Bob)

- applies Alice's public key K_A^+ to $K_A^-(m)$
- if $K_A^+(K_A^-(m)) = m$ whoever signed m was Alice or has Alice's private key



Problem for digital signatures

Public key cryptography is expensive

- more expensive the longer the message is
- Why?

Solution

- sign digital ``fingerprint" of msg rather than msg itself

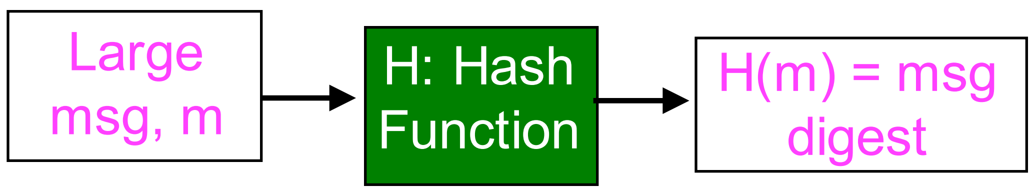
Message digest

Message digest

Desired features are what hash function gives

- fixed-length
- easy-to-compute
- 2 msgs unlikely to have same digest

Apply hash function H to m



Hash function properties

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

Some hash function standards

MD5 hash function (RFC 1321)

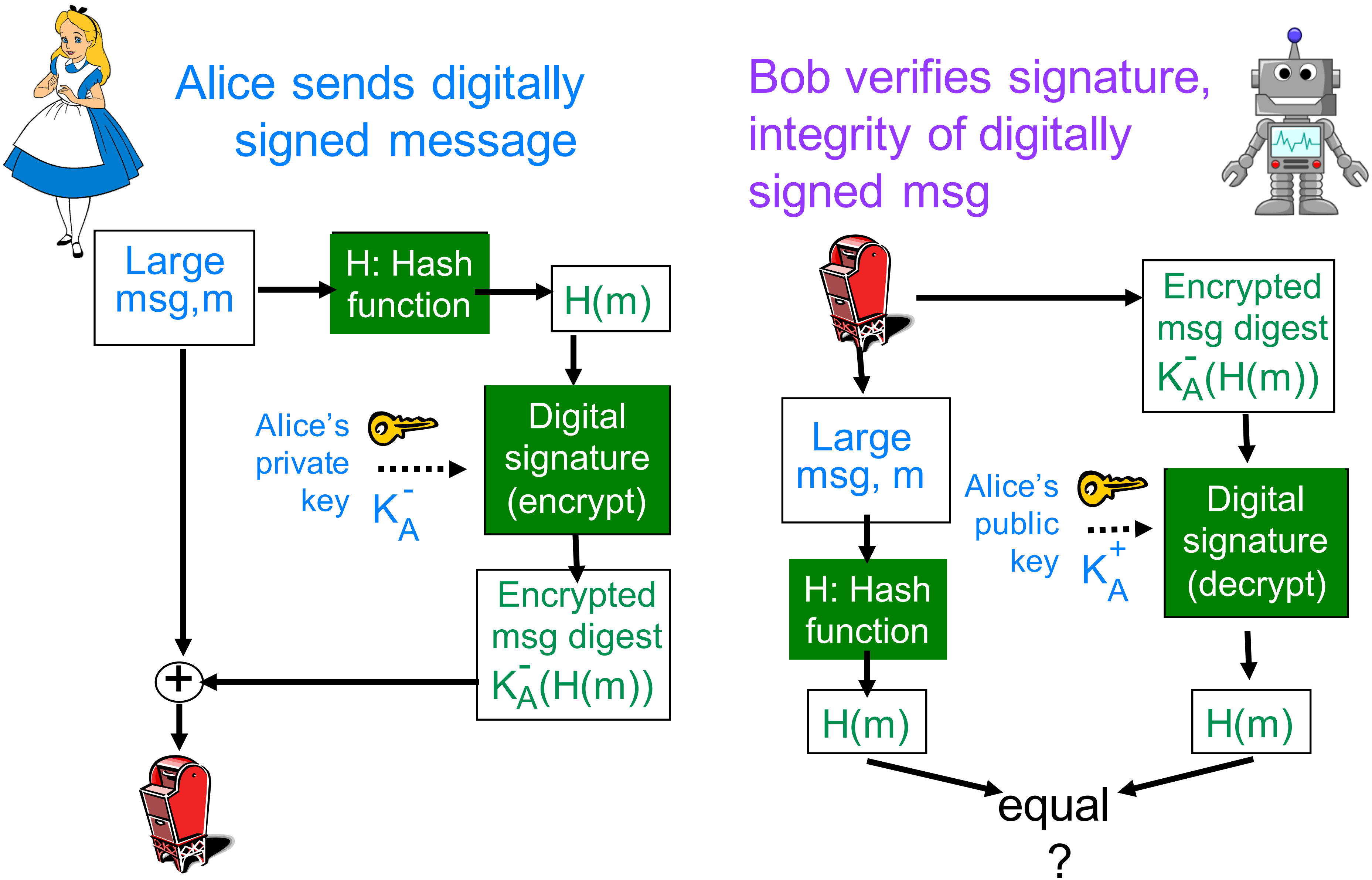
- computes 128-bit message digest in 4-step process.
- "cryptographically broken and unsuitable for further use"
  - CMU Software engineering Institute

SHA-1

- 160-bit message digest
- many vulnerabilities, browsers will no longer use/accept

SHA-2, SHA-3

Use signed message digest as digital signature



Transport Layer Security

OVERVIEW

TLS aka SSL

Secures data at and above transport layer

- provides confidentiality, integrity, authentication
- SSL: Secure Sockets Layer, predecessor to TLS
- TLS: Transport Layer Security

Available to all TCP applications

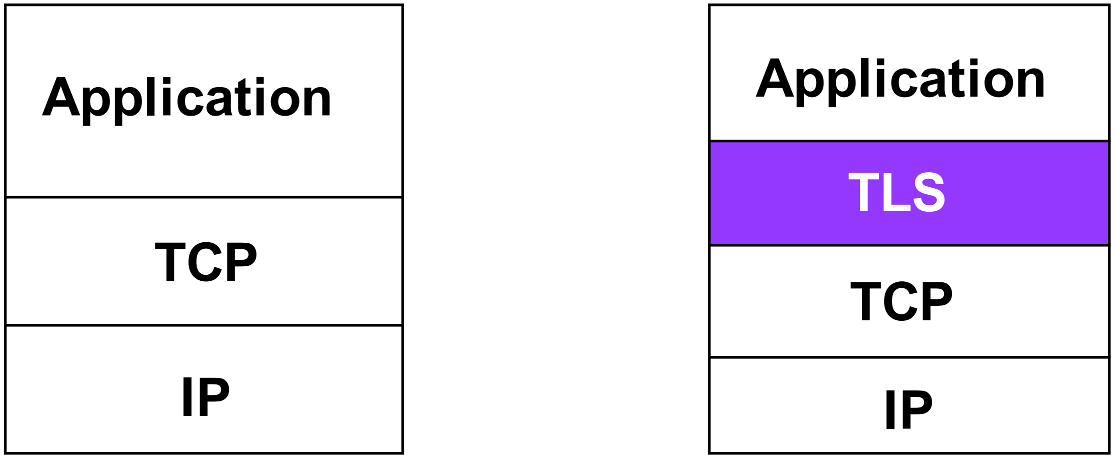
- first setup TCP connection, then run TLS as application

Widely deployed

- supported by almost all browsers, web servers
- billions $/year over SSL
- HTTP + SSL = HTTPS

Where TLS sits in Internet stack

TLS provides application programming interface to apps



Normal application   Application with TLS

Very likely your operating system using open source library

- https://www.openssl.org/
- https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS

TLS goals

Send byte streams & interactive data

- why?

Want set of secret keys for entire connection

- why?

Want certificate exchange as part of protocol handshake phase

- why?

Transport Layer Security

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

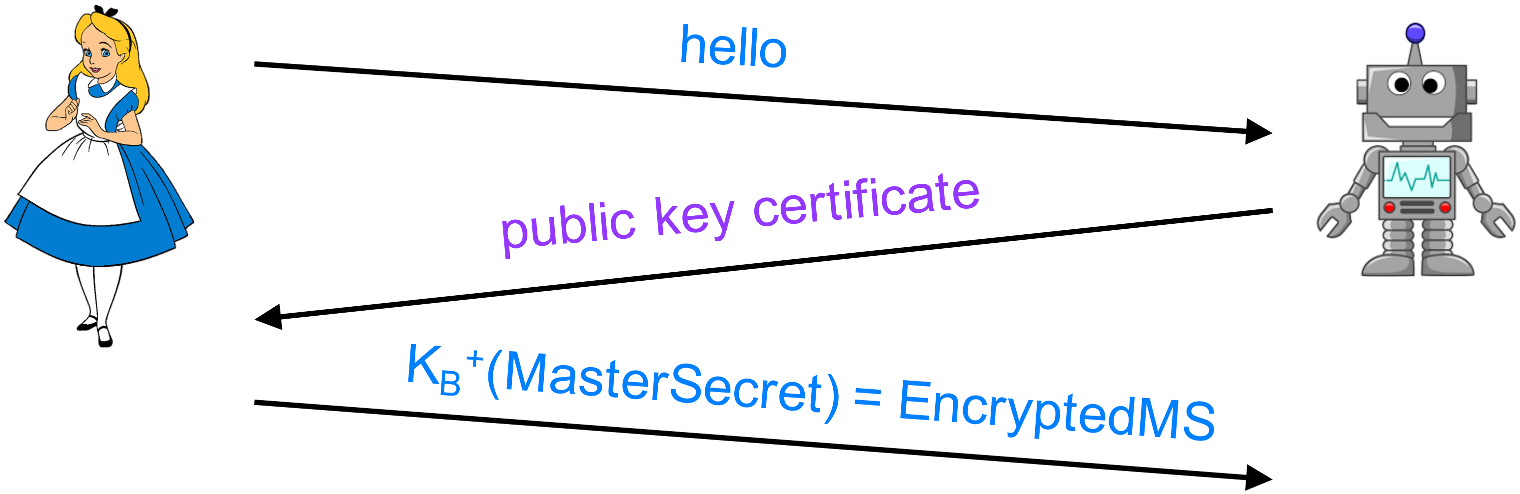
Data transfer

- data to be transferred is broken up into series of records

Connection closure

- special messages to securely close connection

A simple handshake



Derive keys from master secret

- use key derivation function (KDF)
  - takes master secret and additional random data and creates keys

Key derivation

Don't use same key for more than one cryptographic operation

- keys for message authentication code (MAC): like hash
- keys for encryption

Encryption keys

- K_c = encryption key for data sent from client to server
- K_s = encryption key for data sent from server to client

MAC keys

- M_c = MAC key for data sent from client to server
- M_s = MAC key for data sent from server to client

Data records

Why not encrypt data in constant stream as we write it to TCP?

- where to put MAC?
  - if at end, no message integrity until all data processed
- e.g., instant messaging
  - how can we do integrity check over all bytes sent before displaying?

Solution: break stream in series of records

- each record carries MAC
- receiver can act on each record as it arrives

More attacks

What if attacker replays or re-orders records?

- Solution: put sequence # into MAC (no seq # field)
- MAC = MAC(M_x, sequence || data)

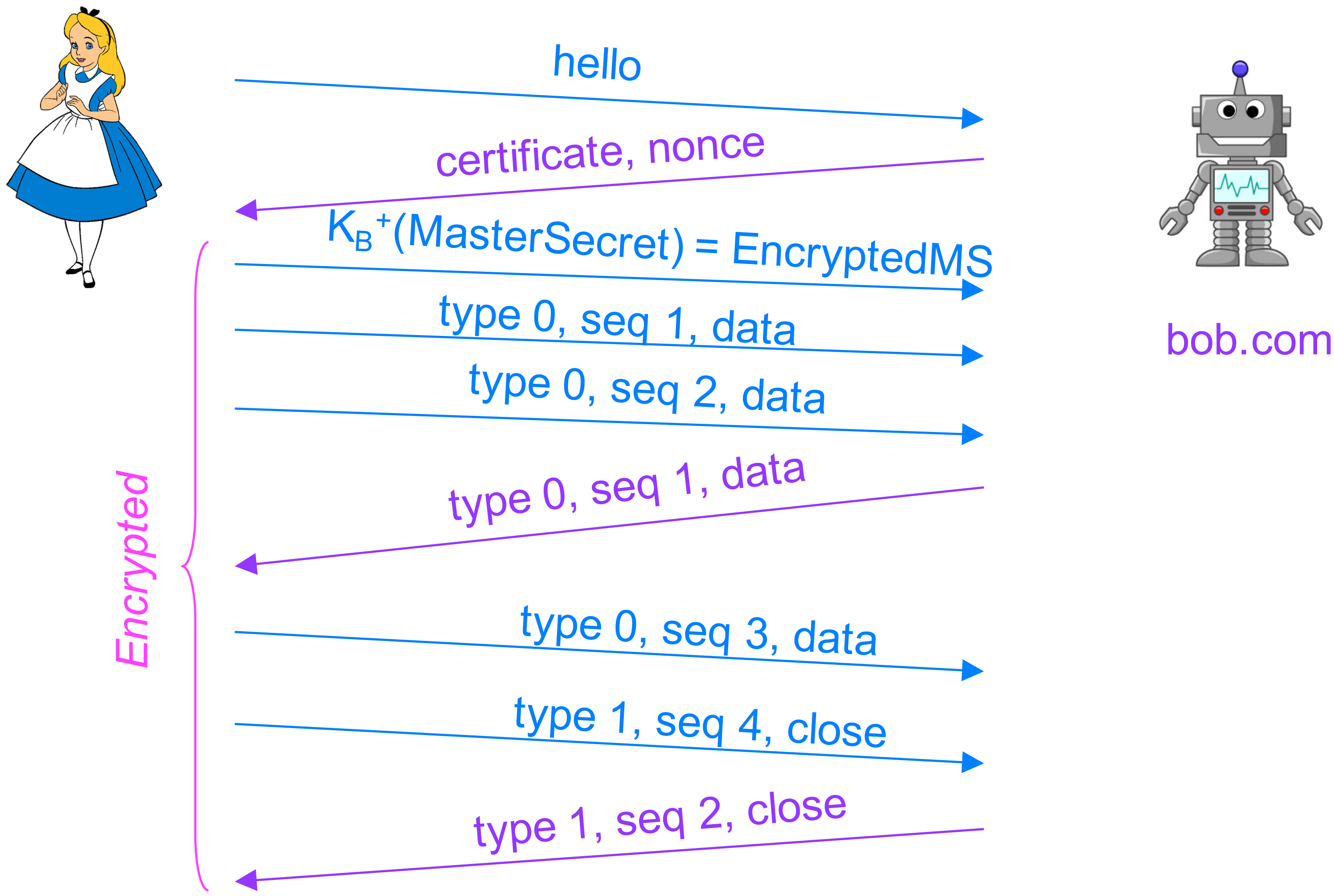
What if attacker replays all records?

- Solution: use nonce

What if attacker forges TCP connection close?

- Solution: have record types, with one type for closure
  - type 0 for data
  - type 1 for closure
- MAC = MAC(M_x, sequence || type || data)

Summary


hello
certificate, nonce
K_B^+(MasterSecret) = EncryptedMS
type 0, seq 1, data
type 0, seq 2, data
type 0, seq 1, data
type 0, seq 3, data
type 1, seq 4, close
type 1, seq 2, close
bob.com
Encrypted