SOFTENG 364: Computer Networks

Network Security

Kurose and Ross, chapter 8

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

ENGINEERING

Learning Outcomes

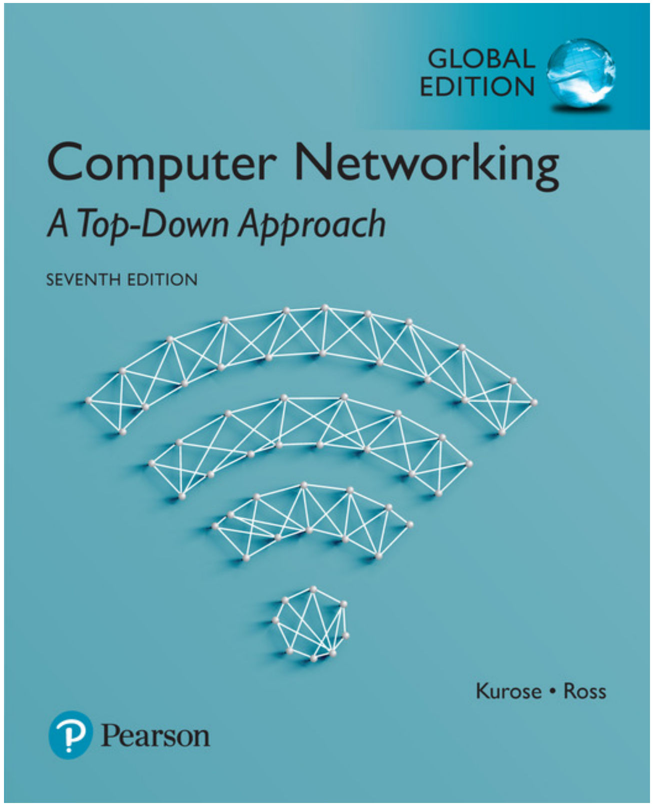
By the end of this module you should be able to:

- Describe the underlying principles of network security
- Explain how cryptography works
- Describe the underlying principals for symmetric and asymmetric cryptography
- Explain what digital signatures and hashes are and how they work
- Describe the challenges in authenticating end-points
- Explain how email can be secured
- Explain why Secure Sockets Layer (SSL) is needed and what it does
- Explain how SSL works
- Describe how SSL prevents malicious actions
- Describe what firewalls are and their purpose
- Explain the advantages and disadvantages of firewalls
- Explain what IDS is and why it is needed

References

Computer Networking: A Top Down Approach, 7th edition (2016). *By J. Kurose & K. Ross.*

- Chapter 8



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

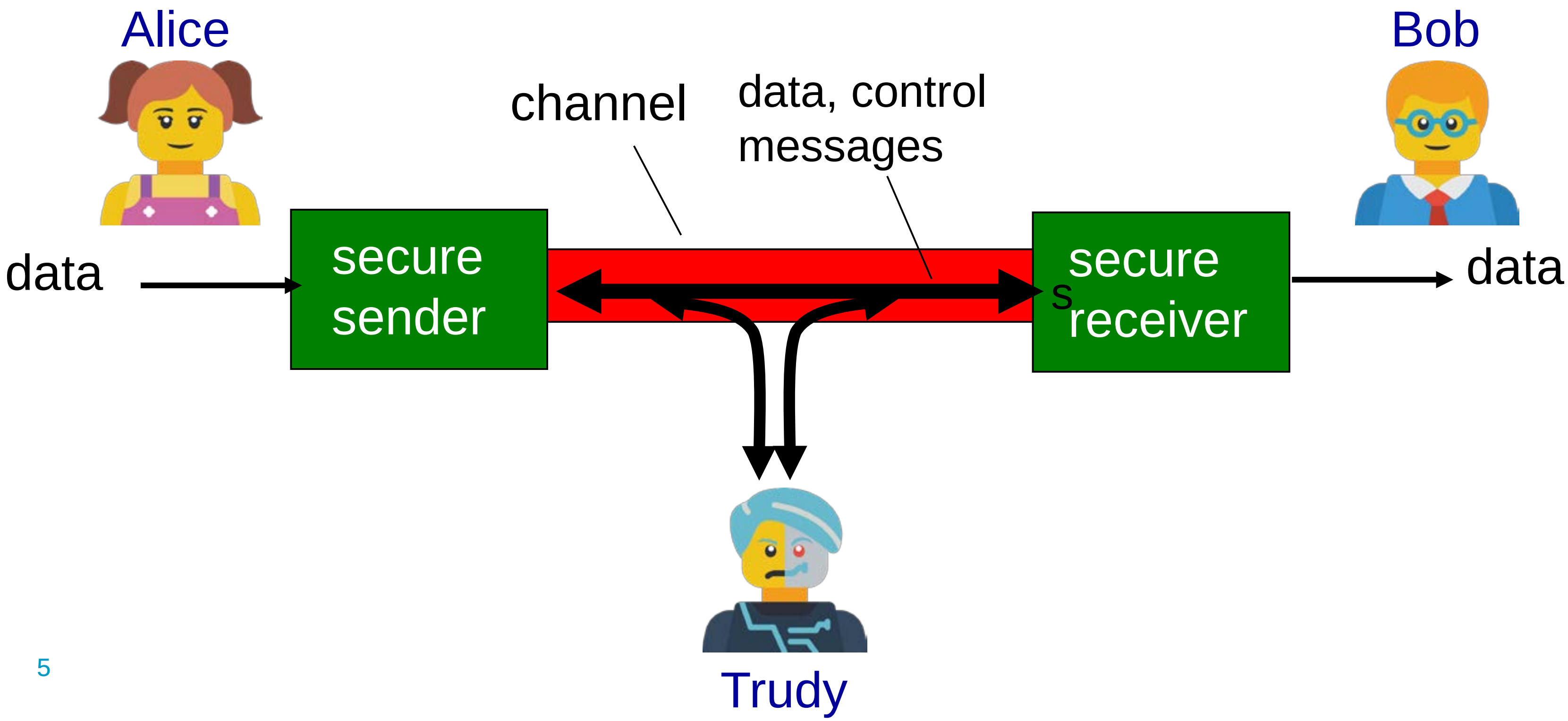
Access and availability: services must be accessible and available to users

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

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)

Purpose of Security

Prevent unwanted (malicious) actions on the network

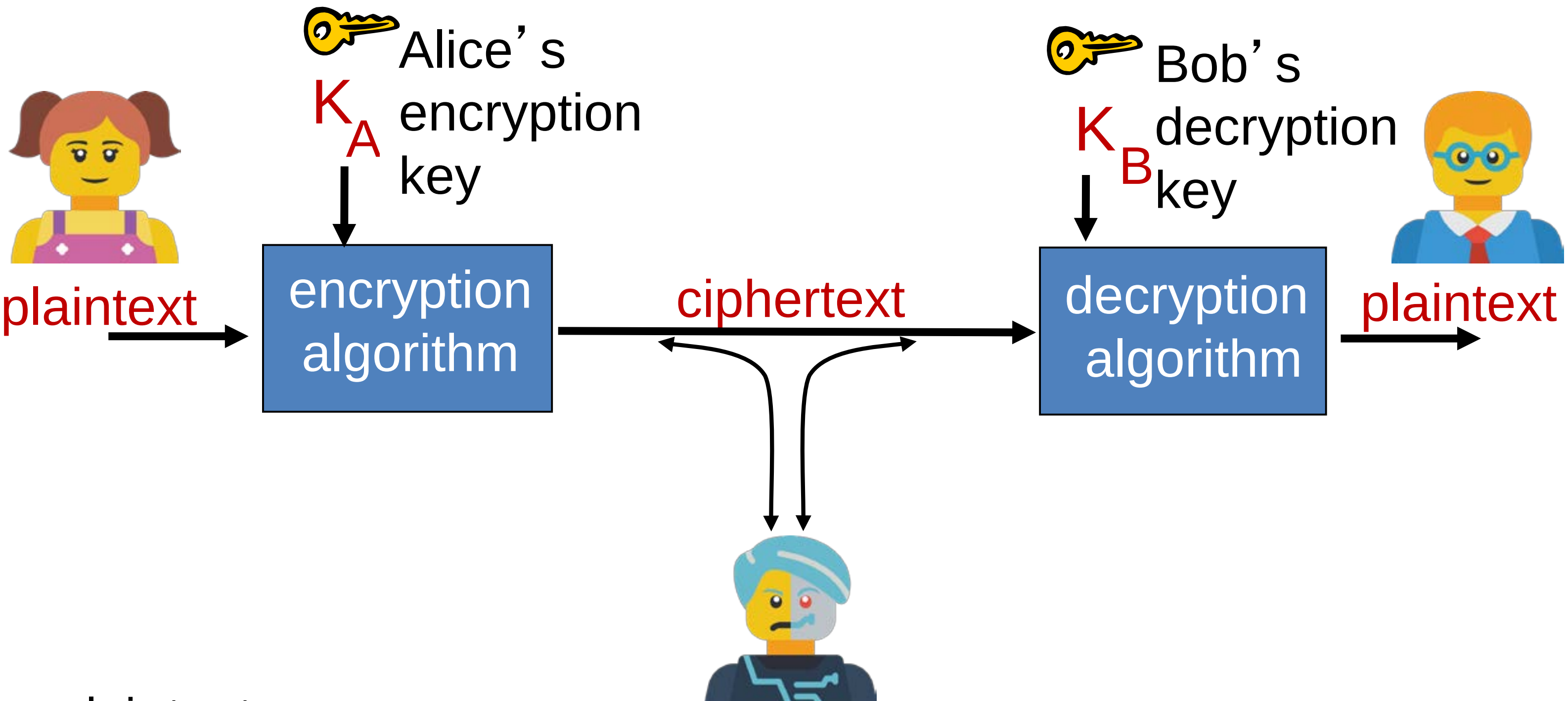
Malicious actions:

- Eavesdropping
- Message insertion
- Impersonation
- Hijacking
- Denial of service

Key terms:

- Authentication
- Integrity
- Confidentiality
- Availability

The language of cryptography



m plaintext message

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

$m = K_B(K_A(m))$

Breaking an encryption scheme

Cipher-text only attack: Trudy has ciphertext she can analyze

Two approaches:

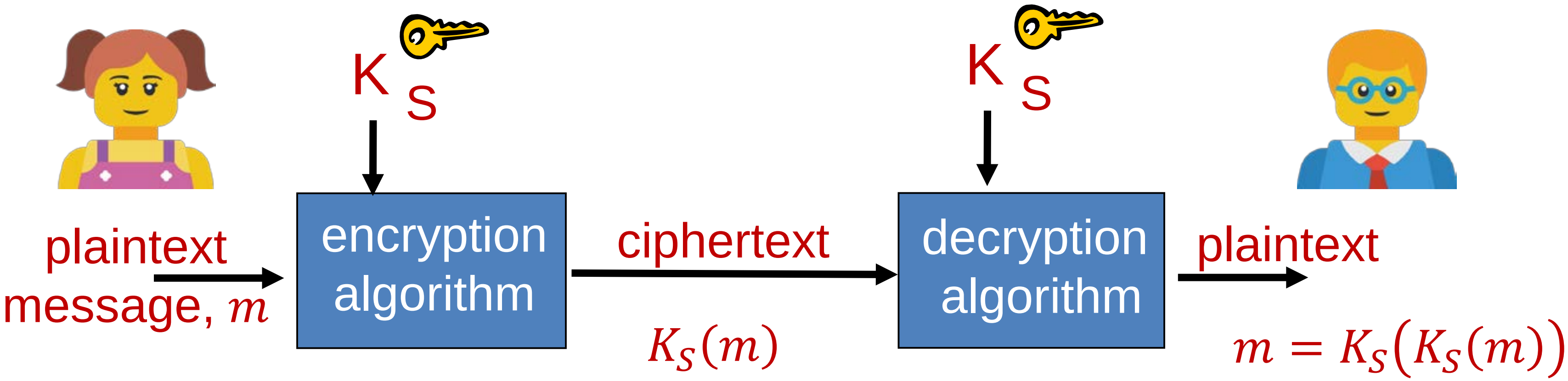
- Brute force: search through all keys
- Statistical analysis

Known-plaintext attack: Trudy has plaintext corresponding to ciphertext

- E.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,

Chosen-plaintext attack: Trudy can get ciphertext for chosen plaintext

Symmetric key cryptography



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

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

Q: 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

Q: how secure is this cypher?

Think about cypher-text only, known plain-text, chosen plain-text

A more sophisticated encryption approach

- n substitution ciphers, $M_1,M_2,\ldots,M_n$
- Cycling pattern:
  - e.g., n=4: M_1,M_3,M_4,M_3,M_2; M_1,M_3,M_4,M_3,M_2; ..
- For each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - dog: d from M_1, o from M_2, g from M_3

Encryption key: n substitution ciphers, and cyclic pattern
- Key need not be just n-bit pattern

Q: how secure is this cypher?

Think about cypher-text only, known plain-text, chosen plain-text

Symmetric key crypto: DES

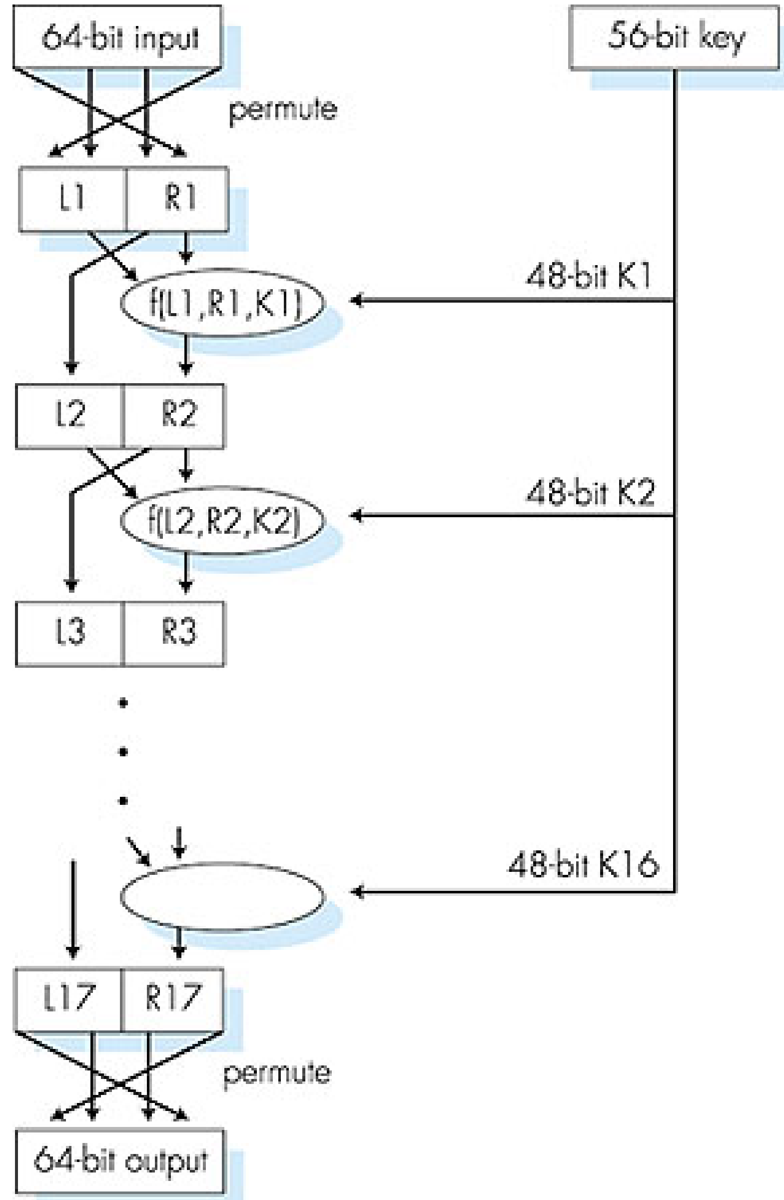
DES: Data Encryption Standard

- 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)

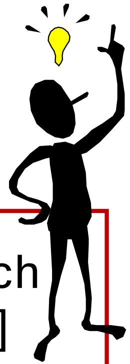
Processes data in 128 bit blocks

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

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
ENGINEERING



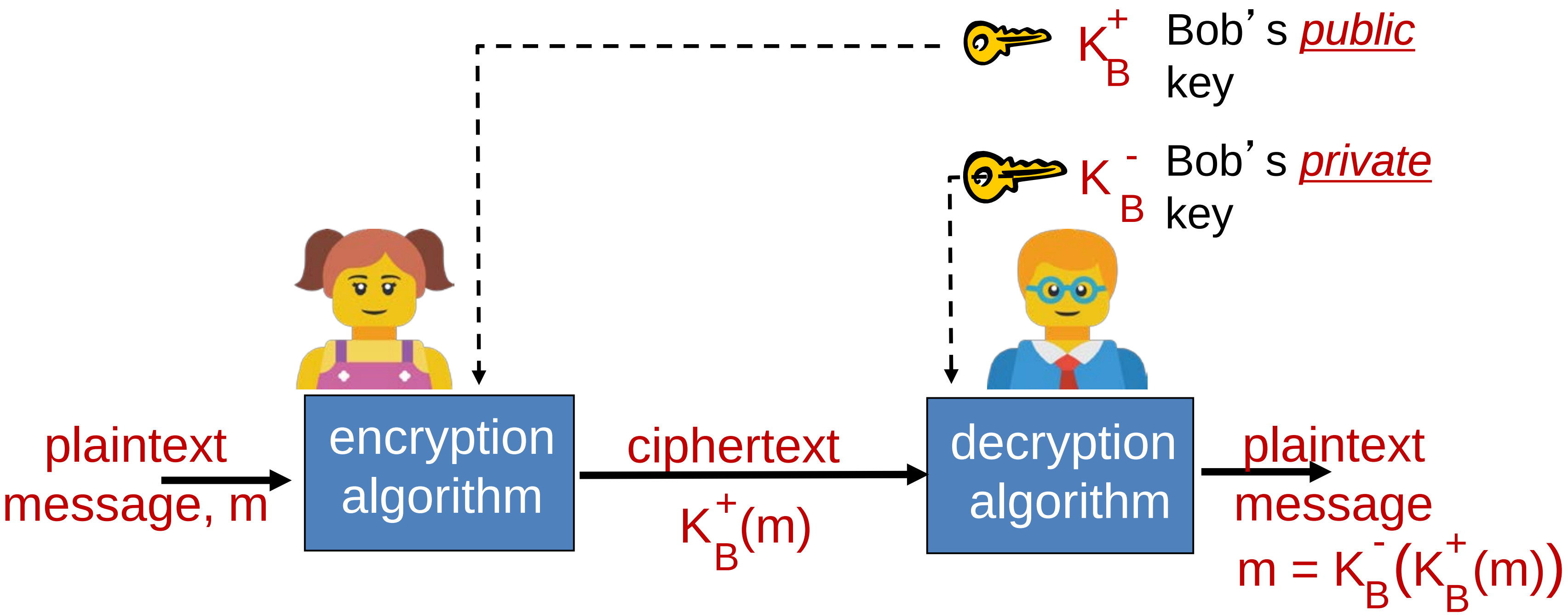
Symmetric key crypto

- 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 $K_B^+(.)$ and $K_B^-(.)$ such that

$$K_B^-\left(K_B^+(m)\right) = m$$

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

RSA: Rivest, Shamir, Adelson algorithm

Prerequisite: modular arithmetic

x mod n = remainder of x when divide by n

Facts:

- [(a mod n) + (b mod n)] mod n = (a+b) mod n
- [(a mod n) - (b mod n)] mod n = (a-b) mod n
- [(a mod n) × (b mod n)] mod n = (a×b) mod n

Thus

- $(a \bmod n)^d \bmod n = a^d \bmod n$

Example: x=14, n=10, d=2:

- $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
- $x^d = 14^2 = 196$ $x^d \bmod 10 = 6$

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 $ed\text{-}1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).
5. *Public key* is $\underbrace{(n,e)}_{K_B^+}$. *Private key* is $\underbrace{(n,d)}_{K_B^-}$.

RSA: encryption, decryption

1. Given (n,e) and (n,d) as computed previously
2. To encrypt message m ($< n$), compute

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

3. 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.

encrypt: $\overbrace{0000l000}^{\text{bit pattern}}$ $\overbrace{12}^{m}$ $\overbrace{24832}^{m^e}$ $\overbrace{17}^{c\,=\,m^e \bmod n}$

decrypt: $\overbrace{17}^{c}$ $\overbrace{481968572106750915091411825223071697}^{c^d}$ $\overbrace{12}^{m\,=\,c^d \bmod n}$

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:

$$\underbrace{K_B^-\left(K_B^+(m)\right)} = m = \underbrace{K_B^+\left(K_B^-(m)\right)}$$

Use public key first, followed by private key

Use private key first, followed by public key

Result is the same!

Why does $K_B^-\left(K_B^+(m)\right) = m = K_B^+\left(K_B^-(m)\right)$?

Follows directly from modular arithmetic:

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

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 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

Digital signatures

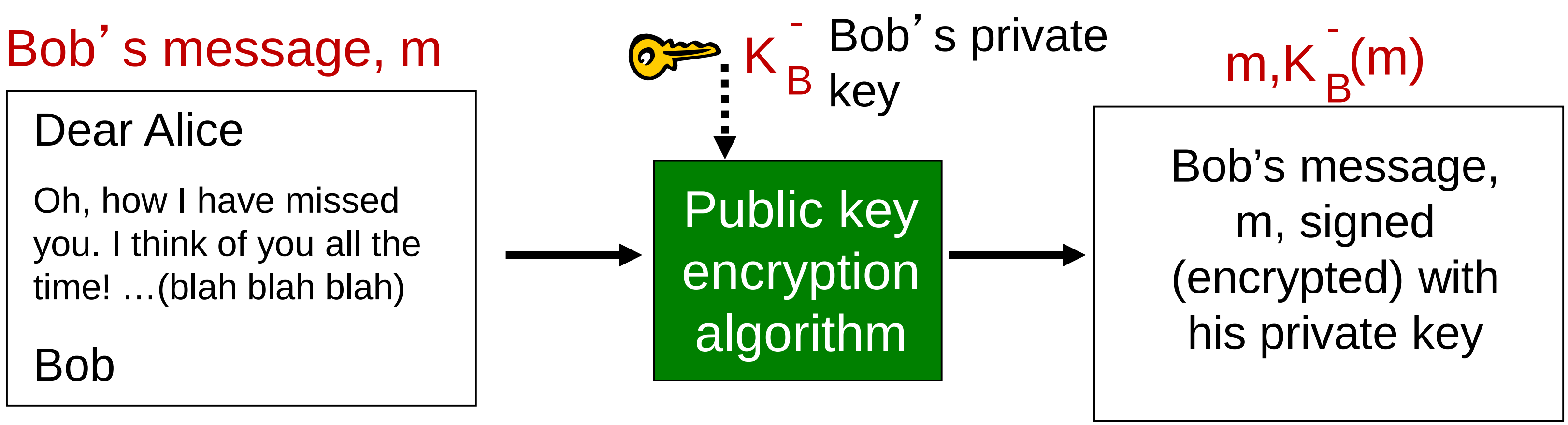
Cryptographic technique analogous to hand-written signatures:

- Sender (Bob) digitally signs document, establishing he is document owner/creator.
- *Verifiable, non-forgeable*: 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, $K_B^-(m)$



Digital signatures

1. Suppose Alice receives msg m, with signature: m, $K_B^-(m)$
2. Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+\left(K_B^-(m)\right) = m$.
3. If $K_B^+\left(K_B^-(m)\right) = m$, 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'

Non-repudiation:

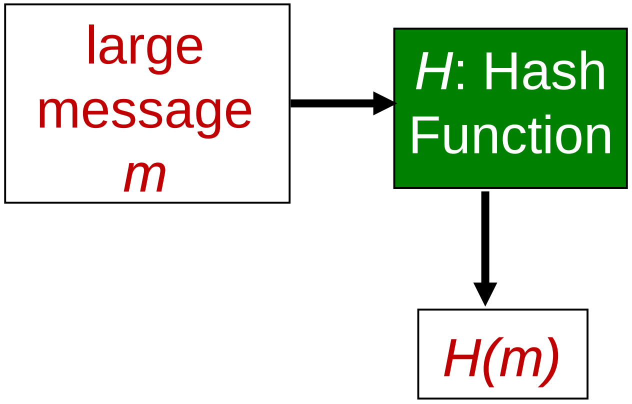
- Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m

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 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:

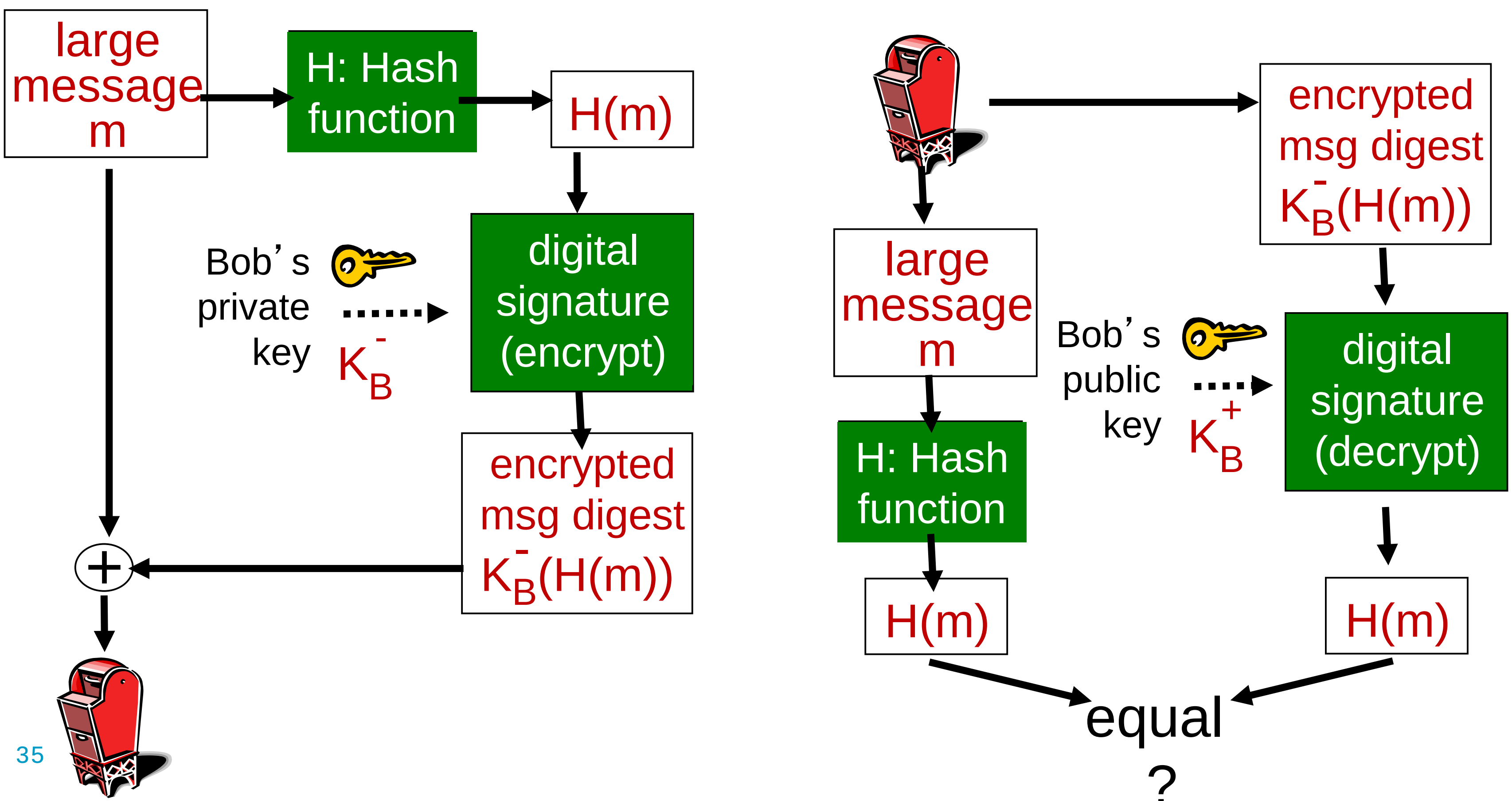
message	ASCII format	message	ASCII format
I O U 1	49 4F 55 31	I O U 9	49 4F 55 39
0 0 . 9	30 30 2E 39	0 0 . 1	30 30 2E 31
9 B O B	39 42 D2 42	9 B O B	39 42 D2 42
	B2 C1 D2 AC		B2 C1 D2 AC

Different messages but identical checksums!

Digital signature = signed message digest

Bob sends digitally signed message:

Alice verifies signature, integrity of digitally signed message:



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 msg m whose MD5 hash is equal to x

SHA-1 is also used

- US standard [NIST, FIPS PUB 180-1]
- 160-bit message digest

Public-key certification

Motivation: Trudy plays pizza prank on Bob

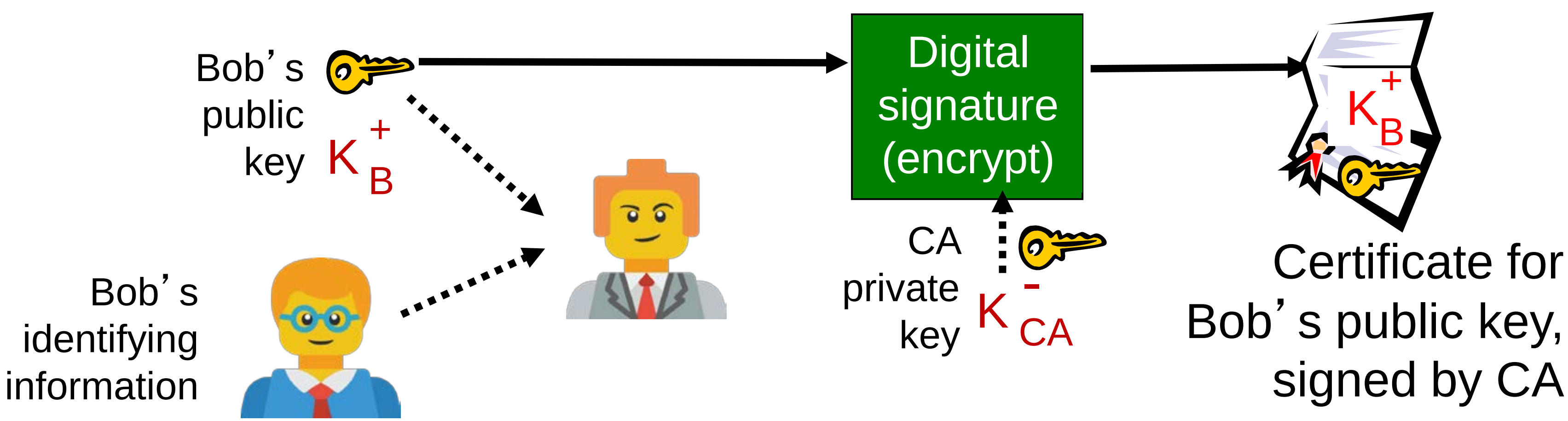
- Trudy creates e-mail order:
  Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob
- Trudy signs order with her private key
- Trudy sends order to Pizza Store
- Trudy sends to Pizza Store her public key, but says it's Bob's public key
- Pizza Store verifies signature; then delivers four pepperoni 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.

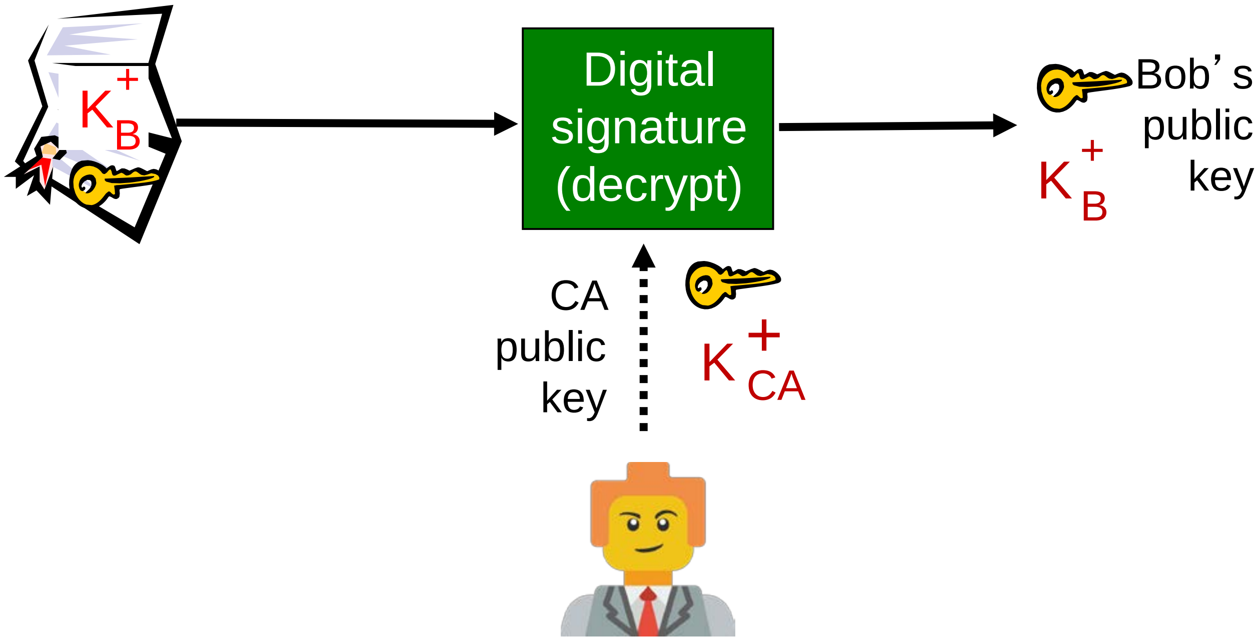
- E provides “proof of identity” to CA.
- CA creates certificate binding E to its public key.
- Certificate containing E’s public key digitally signed by CA – CA says “this is E’s public key”



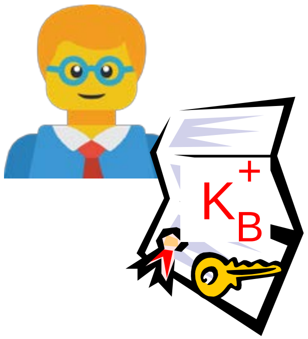
Certification authorities

When Alice wants Bob’s public key:

- Gets Bob’s certificate (Bob or elsewhere).
- Apply CA’s public key to Bob’s certificate, get Bob’s public key



Why do we need CAs?



Can’t anyone issue a certificate?

Adds a validation step

- When checking a certificate, check the CA signature is correct first
- If the signature is valid, then the certificate was issued by the CA
- Otherwise…

How do you trust the certification authority?

Some Common Certification Authorities

Let’s Encrypt (Open Source)

Comodo

DigiCert

Symantec* (formerly VeriSign)

GeoTrust*

* *Backed by DigiCert*

Plus Amazon, Google, Microsoft

And several governments!

Authentication

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

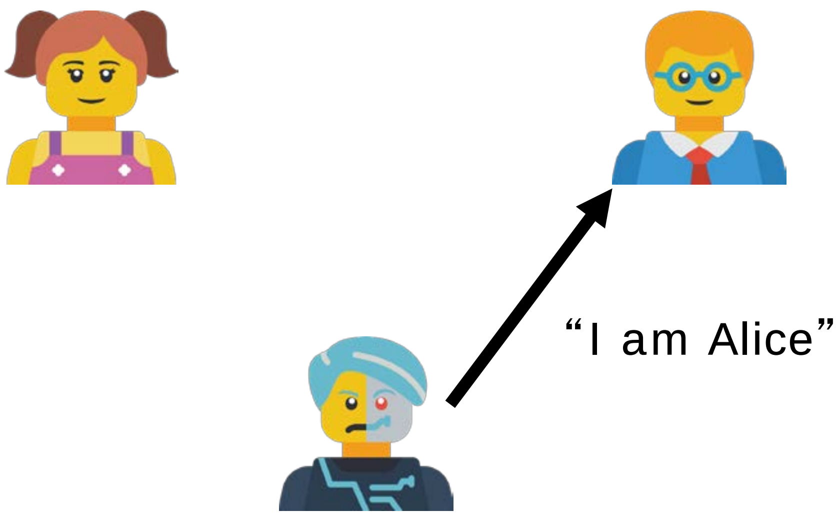
Protocol ap1.0: Alice says “I am Alice”

Failure scenario??

Authentication

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

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



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

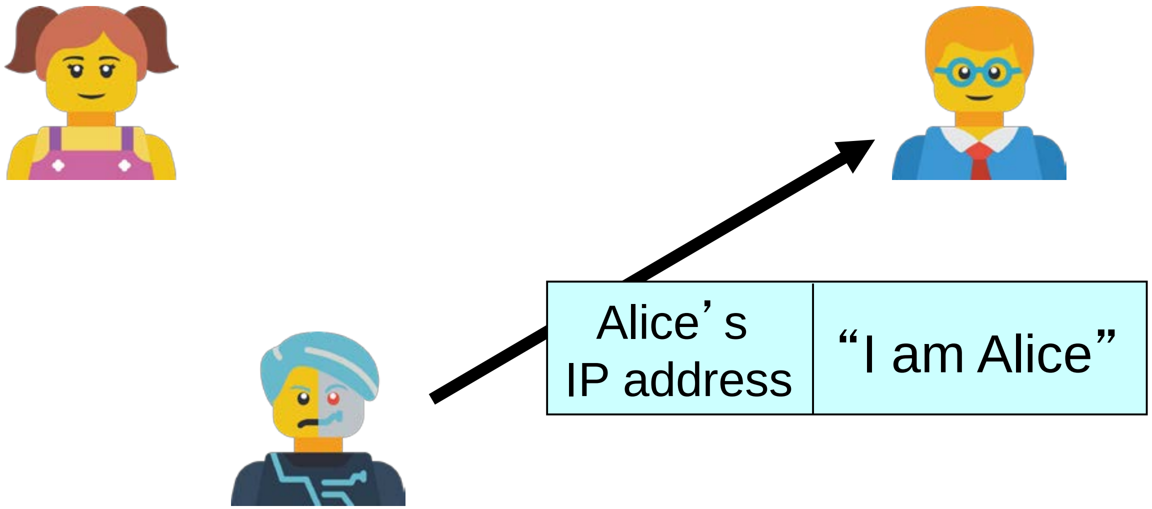
Authentication: another try

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

Failure scenario??

Authentication: another try

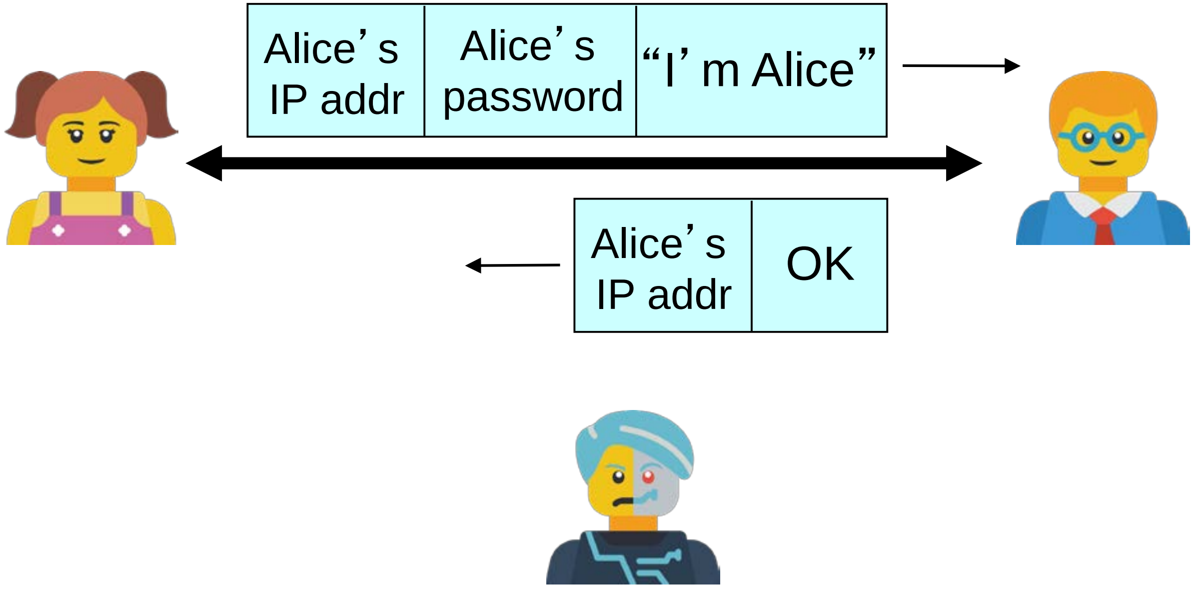
Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



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

Authentication: another try

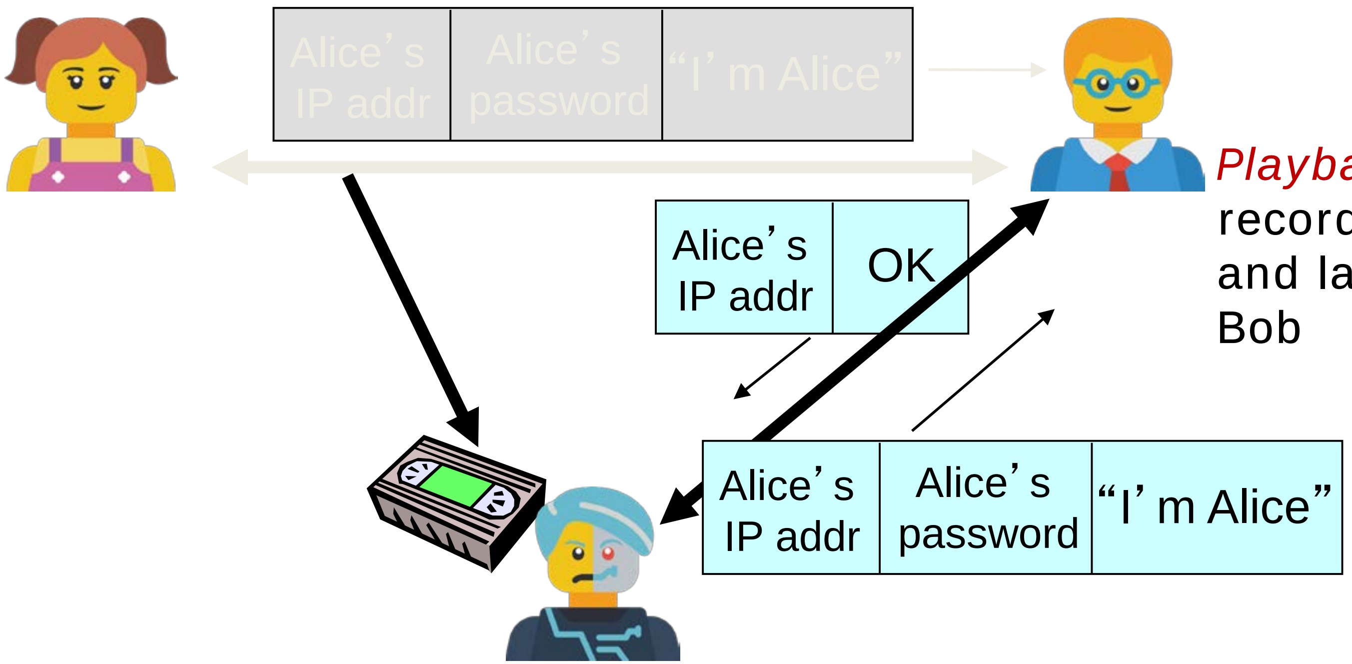
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



Failure scenario??

Authentication: another try

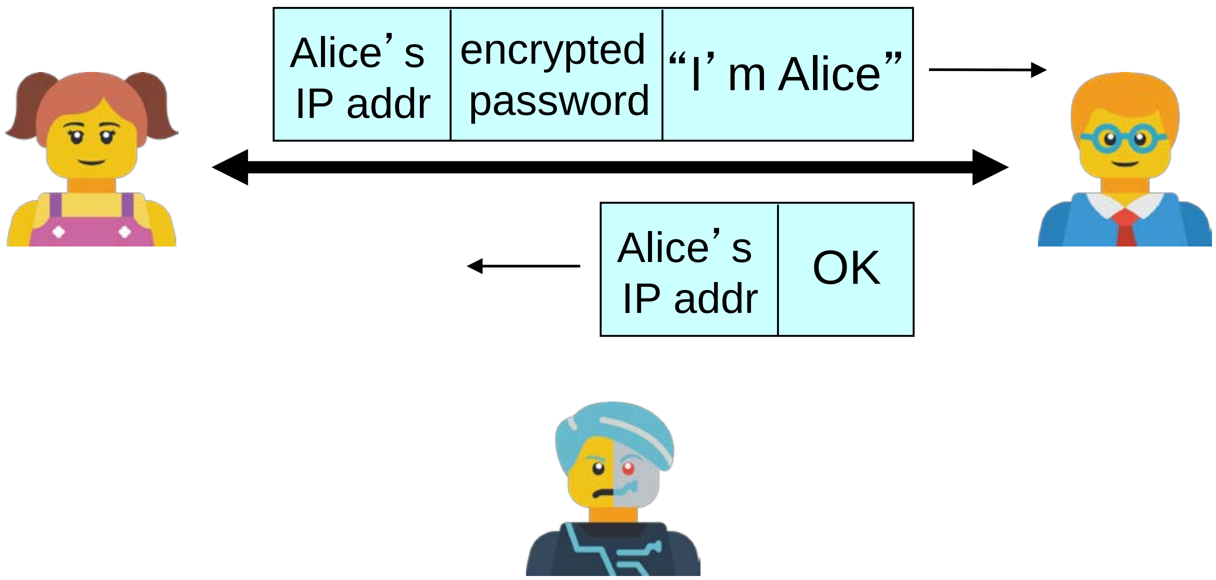
Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.



Playback attack: Trudy records Alice’s packet and later plays it back to Bob

Authentication: yet another try

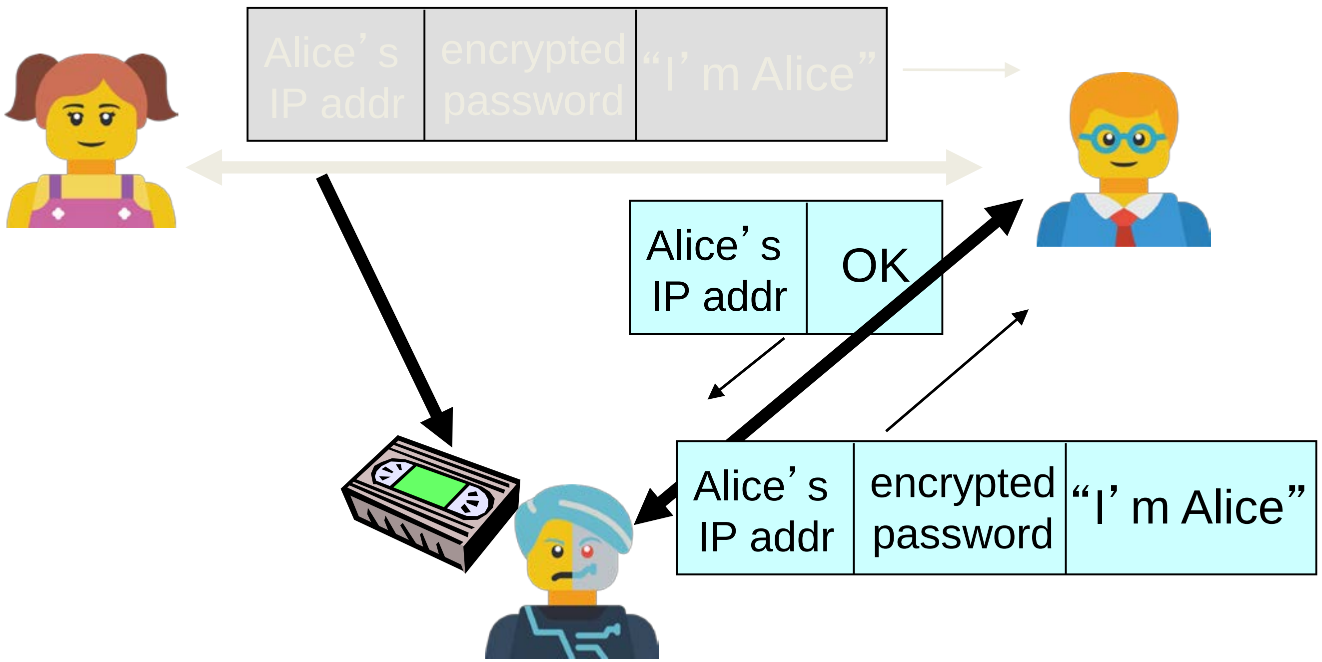
Protocol ap3.1: Alice says “I am Alice” and sends her *encrypted* secret password to “prove” it.



Failure scenario??

Authentication: yet another try

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



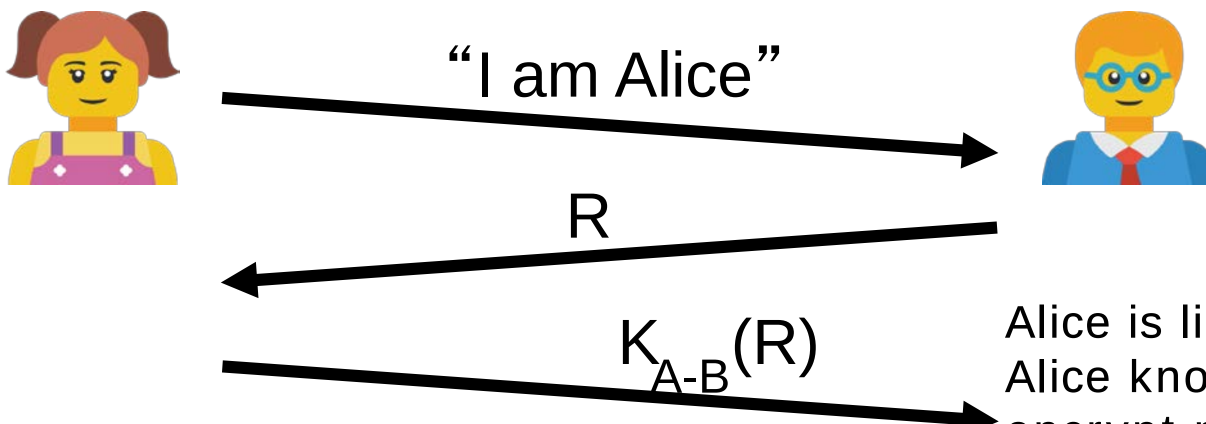
Record and playback *still* works!

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



Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

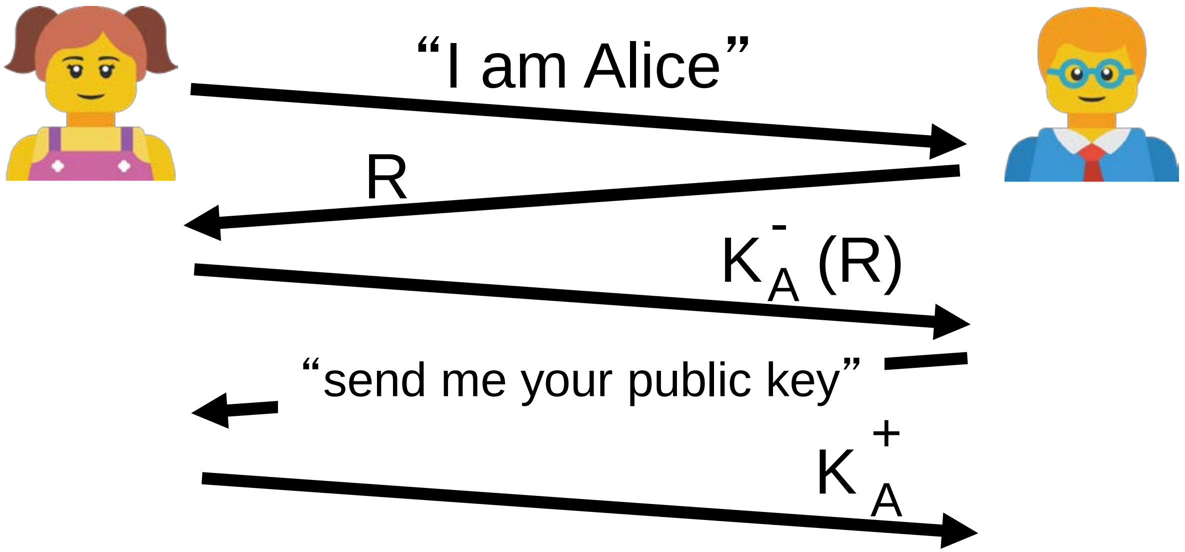
Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

- Can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



Bob computes

$$K_A^+\left(K_A^-(R)\right) = R$$

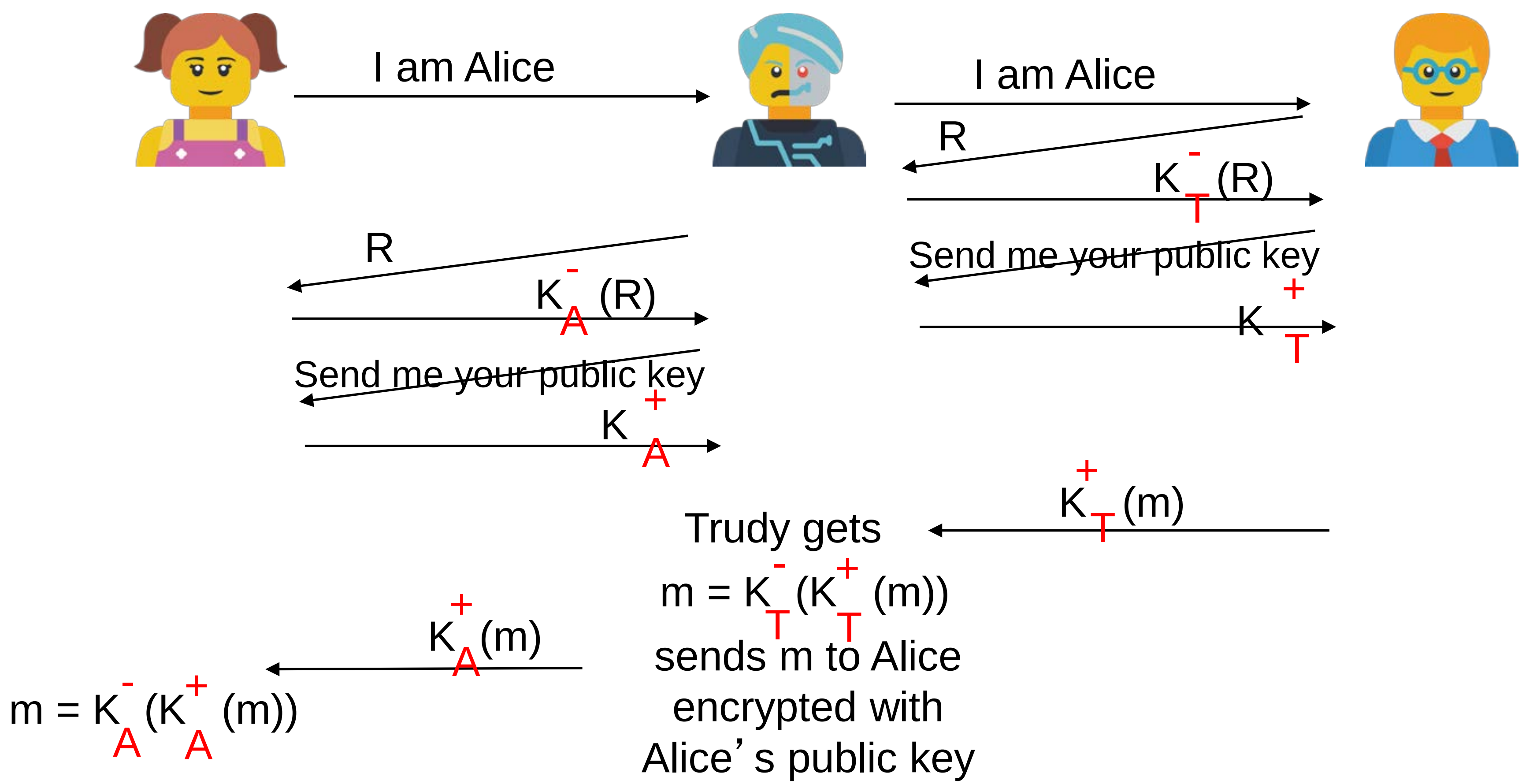
and knows only Alice could have the private key, that encrypted R such that

$$K_A^+\left(K_A^-(R)\right) = R$$

THE UNIVERSITY OF AUCKLAND Te Whare Wānanga o Tāmaki Makaurau NEW ZEALAND | ENGINEERING

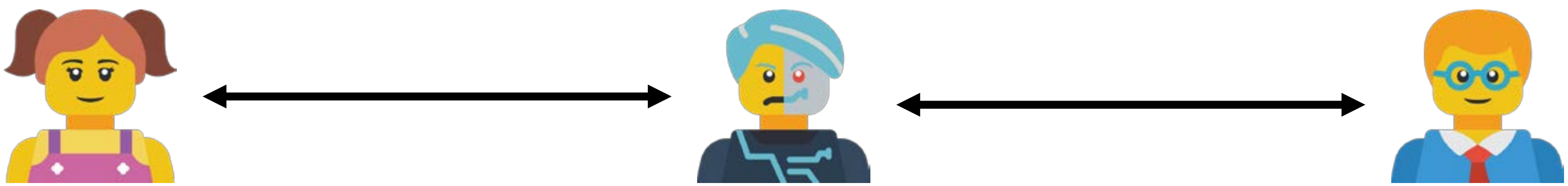
ap5.0: security hole

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)

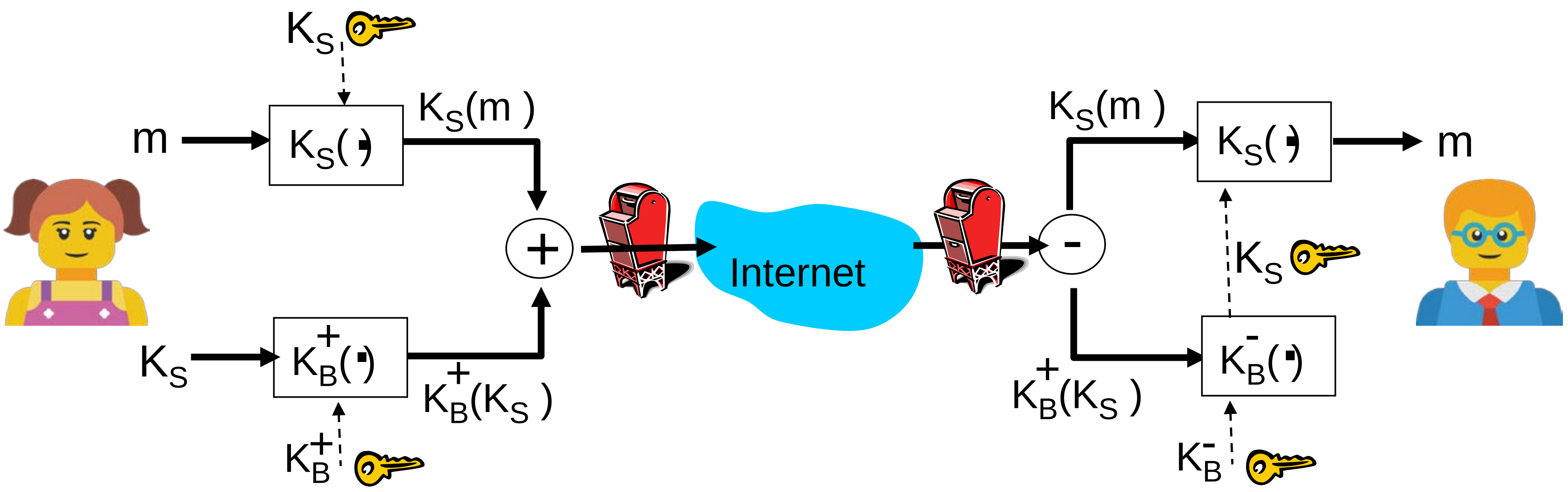


Difficult to detect:

- 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!

Secure e-mail

Alice wants to send confidential e-mail, m, to Bob.

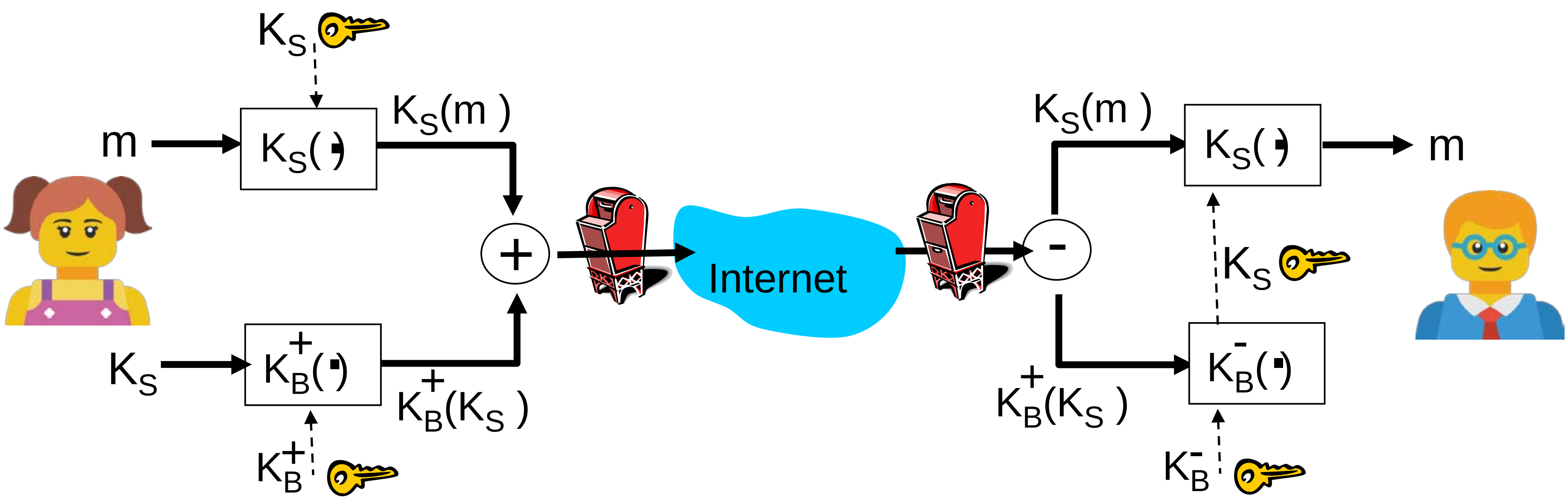


Alice:

- Generates random *symmetric* private key, K_S
- Encrypts message with K_S (for efficiency)
- Also encrypts K_S with Bob's public key
- Sends both $K_S(m)$ and $K_B(K_S)$ to Bob

Secure e-mail

Alice wants to send confidential e-mail, m, to Bob.

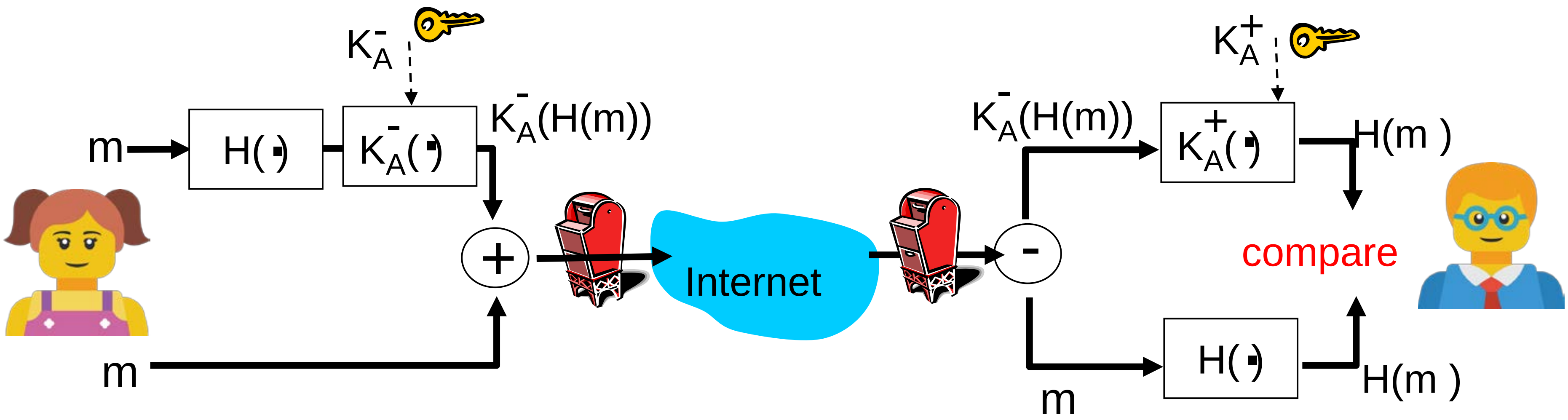


Bob:

- Uses his private key to decrypt and recover K_S
- Uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail (continued)

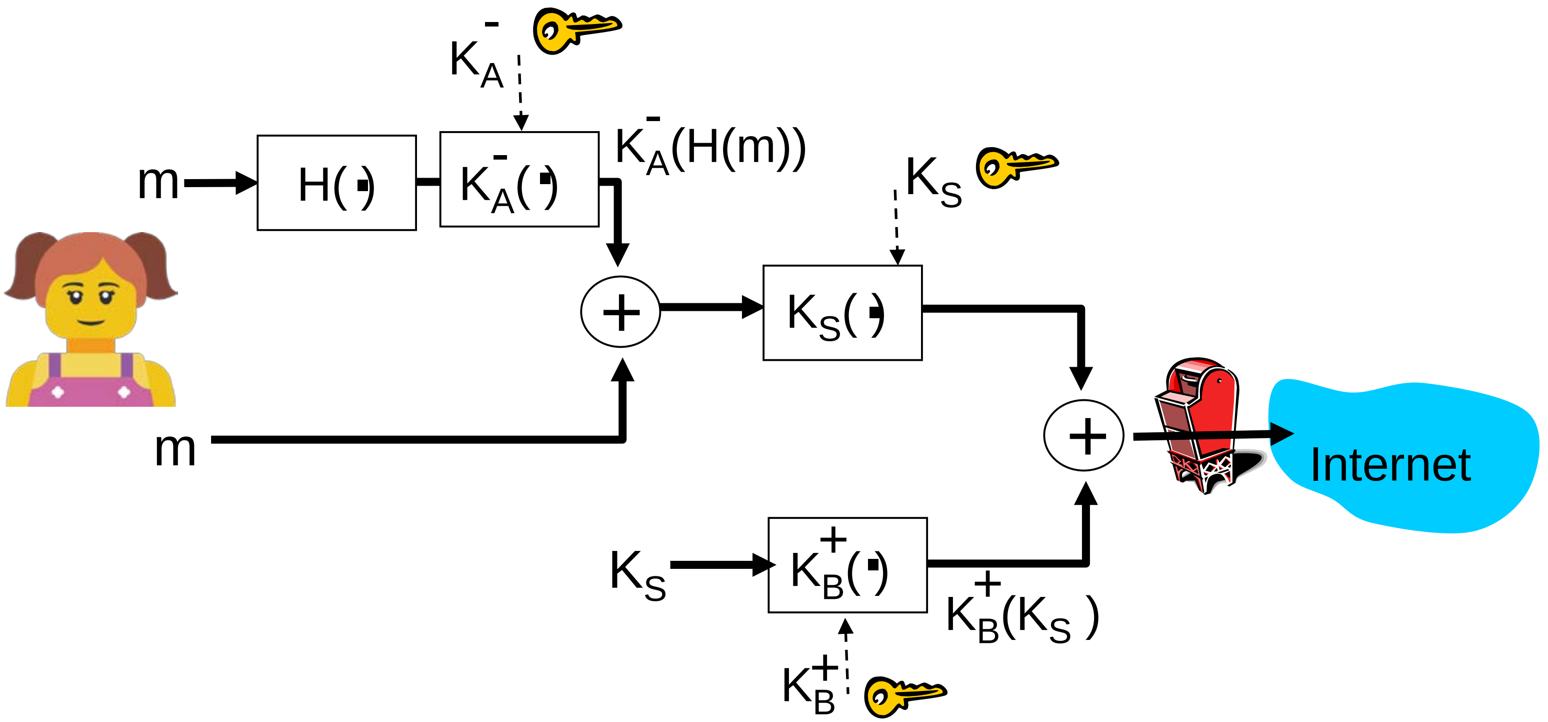
Alice wants to provide sender authentication message integrity



- Alice digitally signs message
- sends both message (in the clear) and digital signature

Secure e-mail (continued)

Alice wants to provide secrecy, sender authentication, message integrity.



Alice uses three keys: her private key, Bob's public key, newly created symmetric key

Secure Sockets Layer

Widely deployed security protocol

- Supported by almost all browsers, web servers
- https
- Billions $/year over SSL

Mechanisms: [Woo 1994], implementation: Netscape

Variation -TLS: transport layer security, RFC 2246

Provides

- Confidentiality
- Integrity
- Authentication

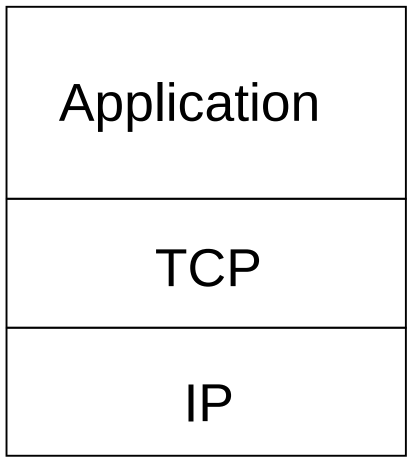
Original goals:

- Web e-commerce transactions
- Encryption (especially credit-card numbers)
- Web-server authentication
- Optional client authentication
- Minimum hassle in doing business with new merchant

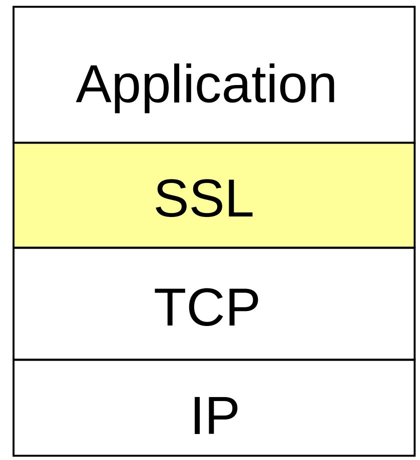
Available to all TCP applications

- Secure socket interface

SSL and TCP / IP



Normal application

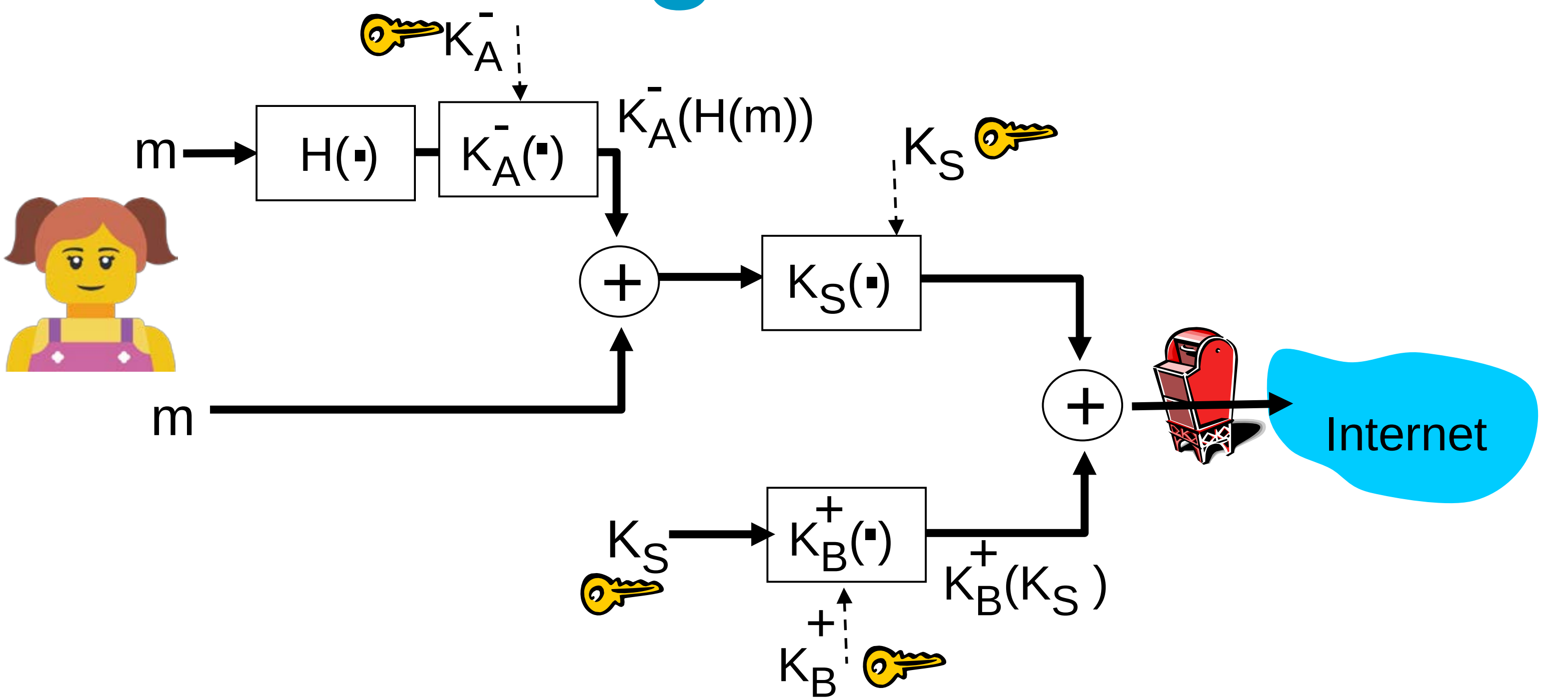


Application with SSL

SSL provides application programming interface (API) to applications

C and Java SSL 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 SSL: a simple secure channel

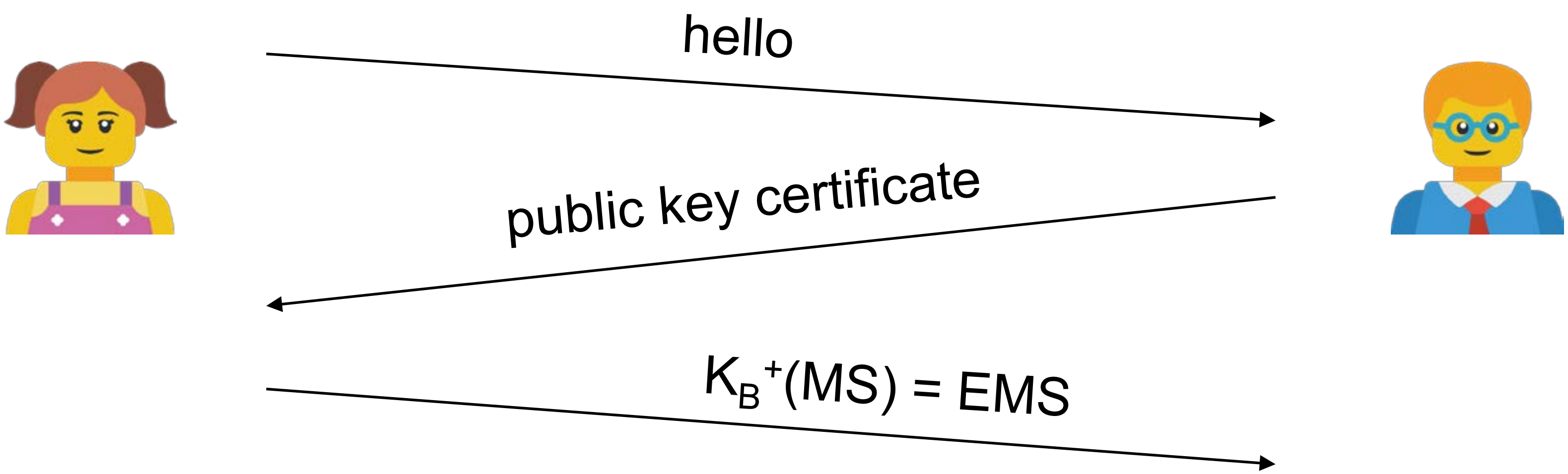
Handshake: Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret

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



MS: master secret

EMS: encrypted master secret

Key derivation

Considered bad to use same key for more than one cryptographic operation

- Use different keys for *Message Authentication Code* (MAC) and encryption

Four keys:

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

Keys derived from key derivation function (KDF)

- Takes master secret and (possibly) some additional random data and creates the keys

Data records

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

- Where would we put the MAC? If at end, no message integrity until all data processed.
- E.g., with instant messaging, how can we do integrity check over all bytes sent before displaying?

Instead, break stream in series of records

- Each record carries a MAC
- Receiver can act on each record as it arrives

Issue: in record, receiver needs to distinguish MAC from data

- Want to use variable-length records

Sequence numbers

Problem: attacker can capture and replay record or re-order records

Solution: put sequence number into MAC:

- MAC = MAC(M_x, sequence||data)
- Note: no sequence number field

Problem: attacker could replay all records

Solution: use nonce

Control information

Problem: truncation attack:

- Attacker fores TCP connection close segment
- One or both sides thinks there is less data than there actually is.

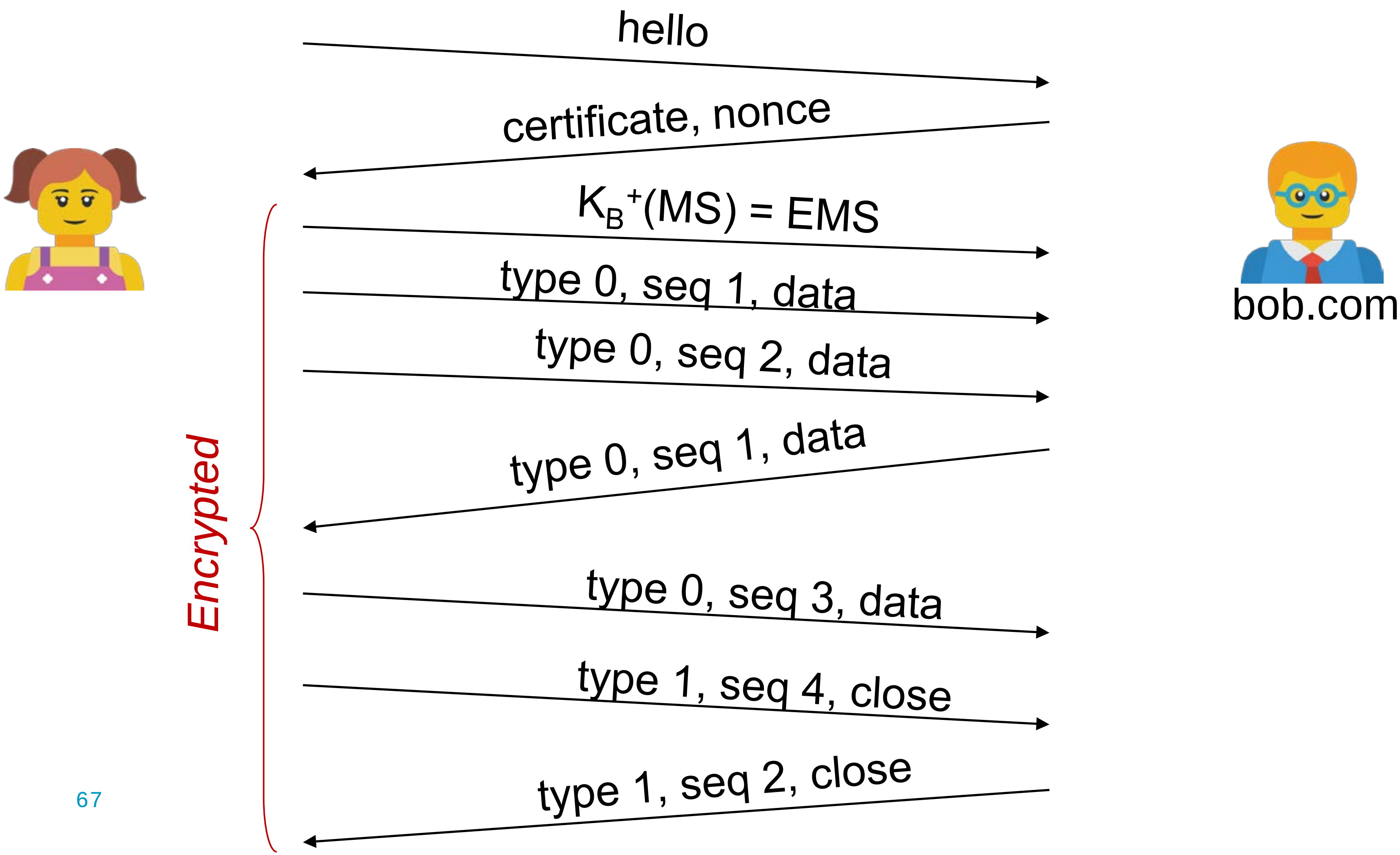
Solution: record types, with one type for closure

- Type 0 for data; type 1 for closure

MAC = MAC(M_x, sequence||type||data)

length	type	data	MAC

Toy SSL: summary



Toy SSL isn’t complete

How long are fields?

Which encryption protocols?

Want negotiation?

- Allow client and server to support different encryption algorithms
- Allow client and server to choose together specific algorithm before data transfer

SSL cipher suite

Cipher suite:

- Public-key algorithm
- Symmetric encryption algorithm
- MAC algorithm

SSL supports several cipher suites

Negotiation: client, server agree on cipher suite

- Client offers choice
- Server picks one

Common SSL symmetric ciphers

- DES – Data Encryption Standard: block
- 3DES – Triple strength: block
- RC2 – Rivest Cipher 2: block
- RC4 – Rivest Cipher 4: stream

SSL Public key encryption

- RSA

Real SSL: handshake (1)

Purpose

1. Server authentication
2. Negotiation: agree on crypto algorithms
3. Establish keys
4. Client authentication (optional)

Real SSL: handshake (2)

1. Client sends list of algorithms it supports, along with client nonce
2. Server chooses algorithms from list; sends back: choice + certificate + server nonce
3. Client verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. Client and server independently compute encryption and MAC keys from pre_master_secret and nonces
5. Client sends a MAC of all the handshake messages
6. Server sends a MAC of all the handshake messages

Real SSL: handshake (3)

Last 2 steps protect handshake from tampering

- Client typically offers range of algorithms, some strong, some weak
- Man-in-the middle could delete stronger algorithms from list
- Last two messages are encrypted

Real SSL: handshaking (4)

Why two random nonces?

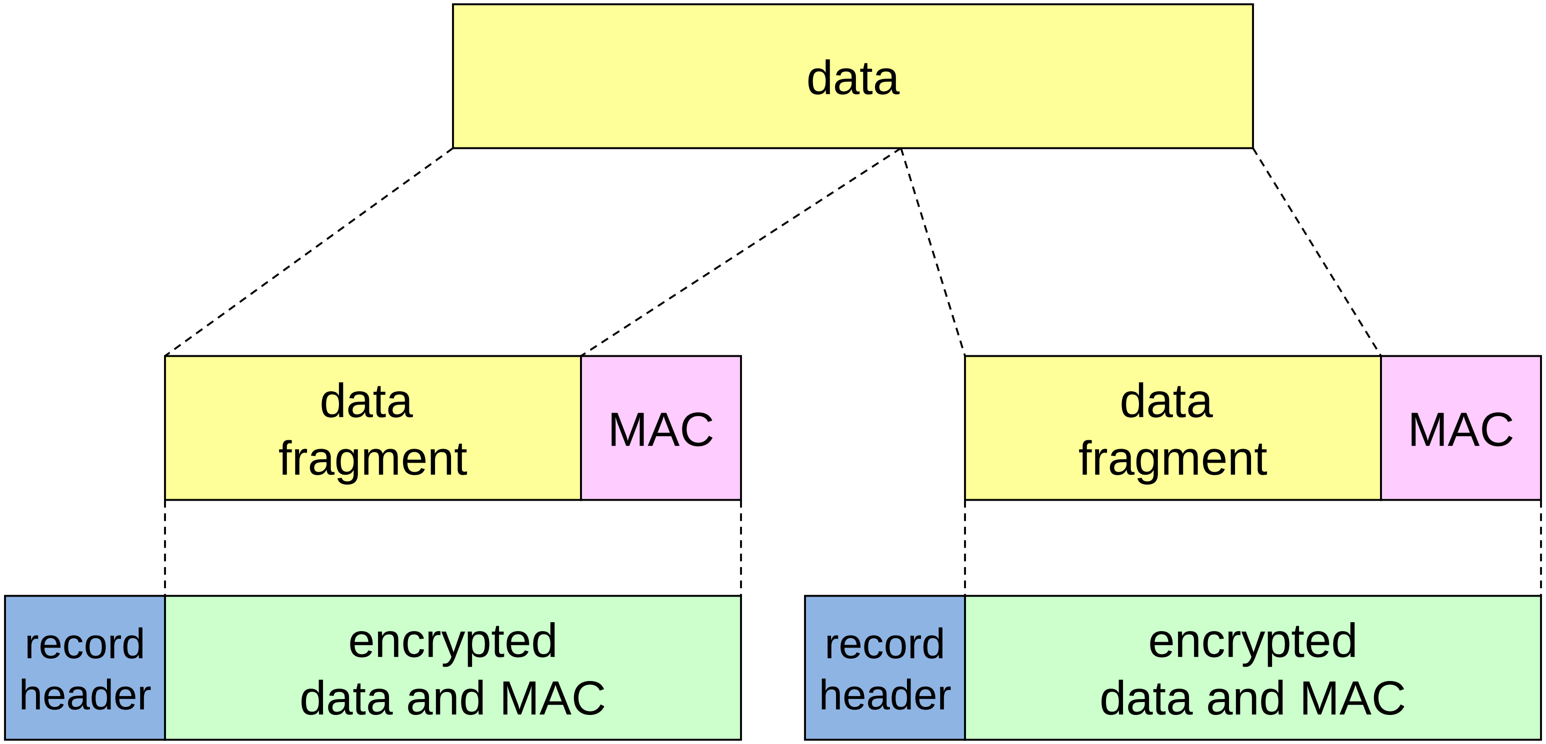
Suppose Trudy sniffs all messages between Alice & Bob
Next day, Trudy sets up TCP connection with Bob, sends exact same sequence of records

- Bob (Amazon) thinks Alice made two separate orders for the same thing

Solution: Bob sends different random nonce for each connection. This causes encryption keys to be different on the two days

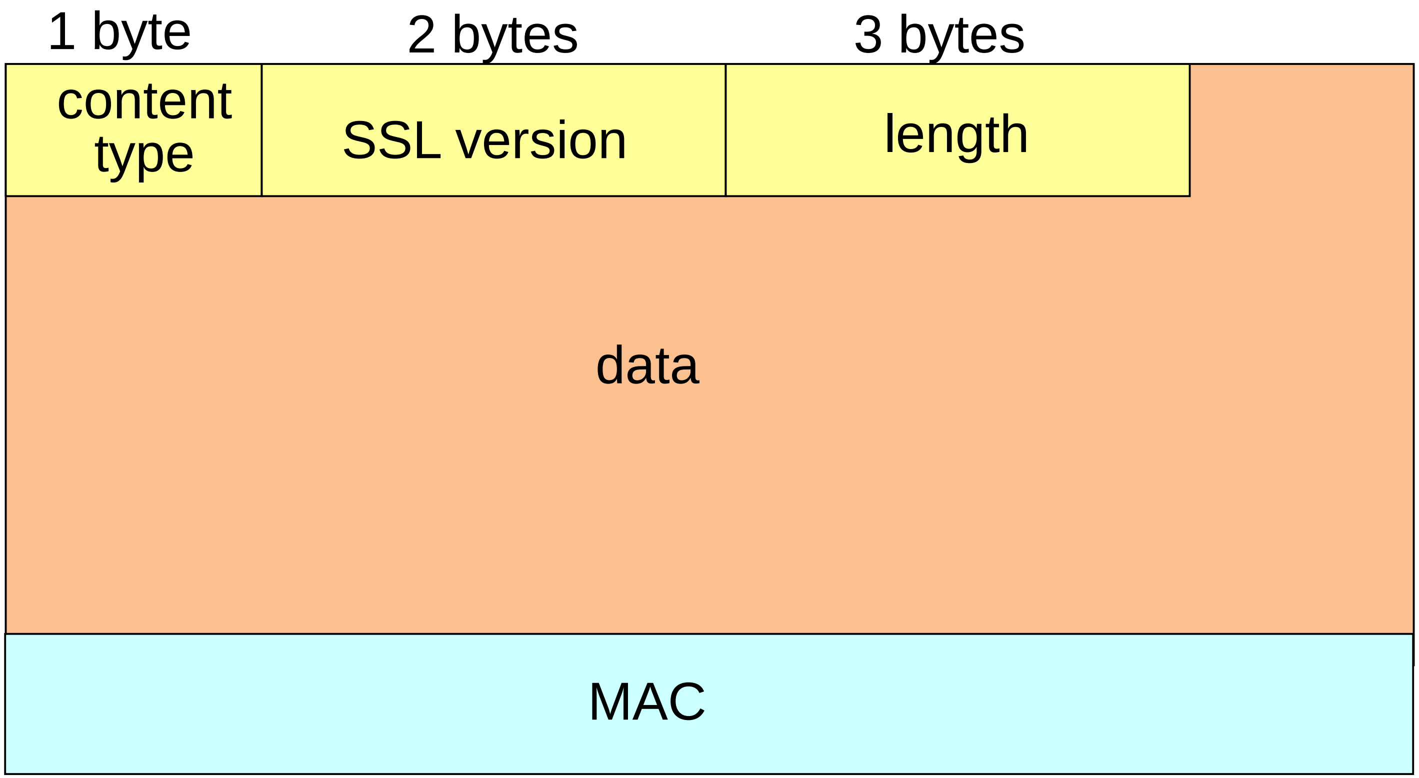
- Trudy's messages will fail Bob's integrity check

SSL record protocol



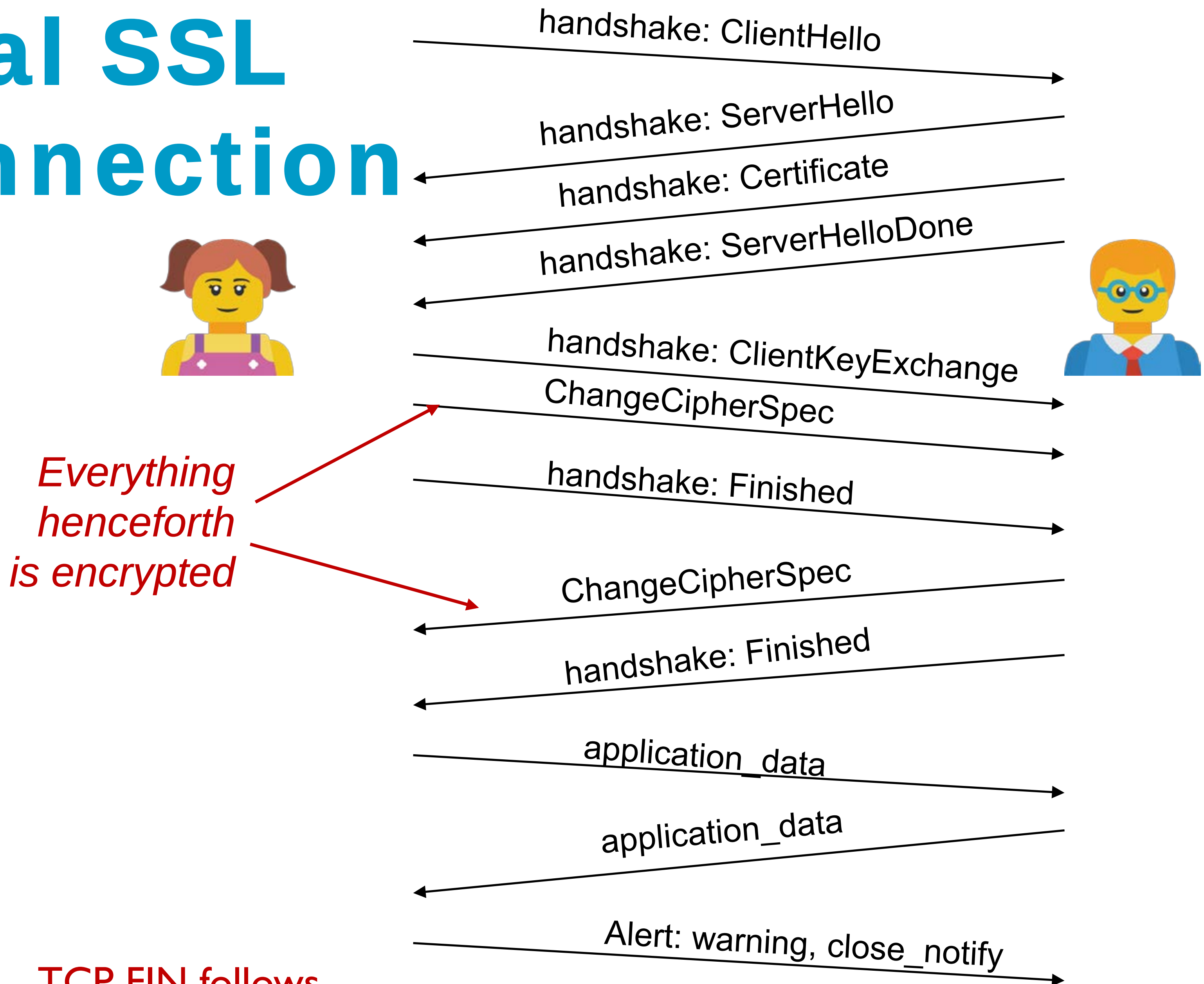
Record header: content type; version; length
MAC: includes sequence number, MAC key M_x
Fragment: each SSL fragment 2^{14} bytes (~16 Kbytes)

SSL record format



Data and MAC encrypted (symmetric algorithm)

Real SSL connection



TCP FIN follows

Key derivation

Client nonce, server nonce, and pre-master secret input into pseudo random-number generator.

- Produces master secret

Master secret and new nonces input into another random-number generator: “key block”

- Because of resumption: TBD

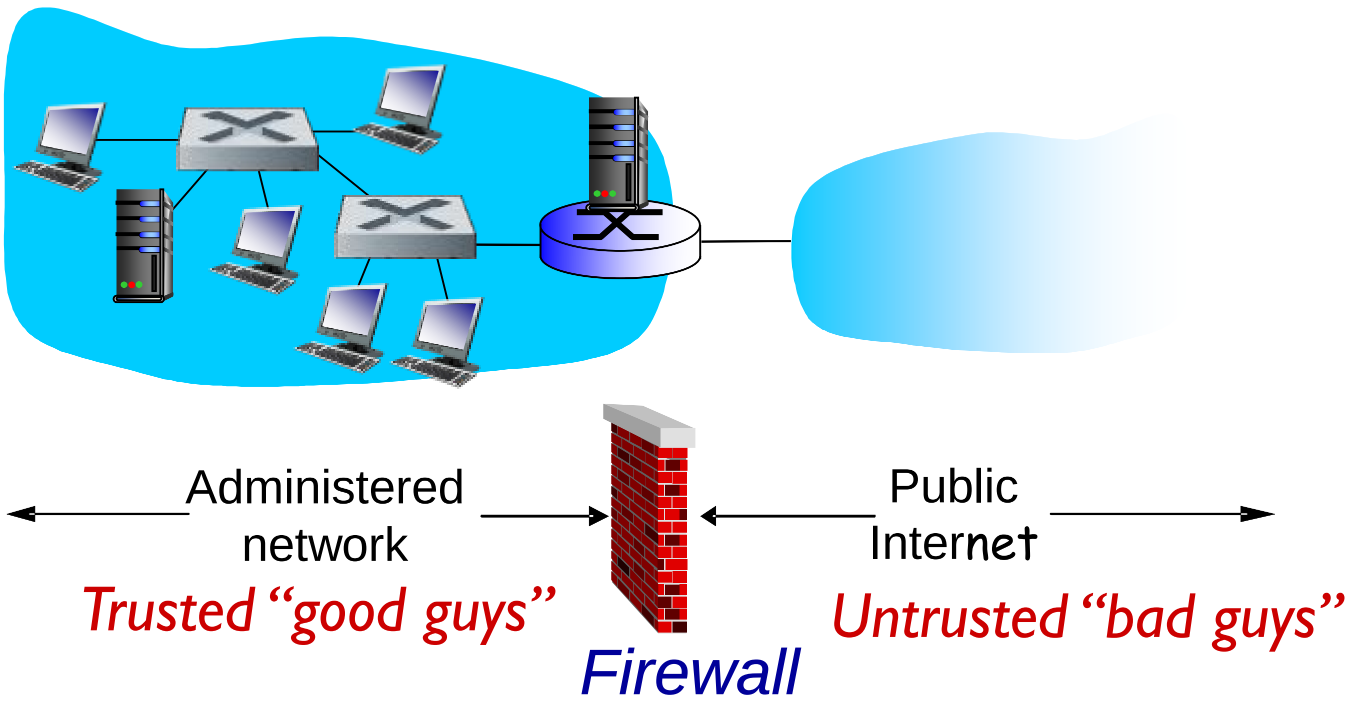
Key block sliced and diced:

- Client MAC key
- Server MAC key
- Client encryption key
- Server encryption key
- Client initialization vector (IV)
- Server initialization vector (IV)

Firewalls

Firewall

Isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others



Firewalls: why

Prevent denial of service attacks:

- SYN flooding: attacker establishes many bogus TCP connections, no resources left for “real” connections

Prevent illegal modification/access of internal data

- E.g., attacker replaces CIA’s homepage with something else

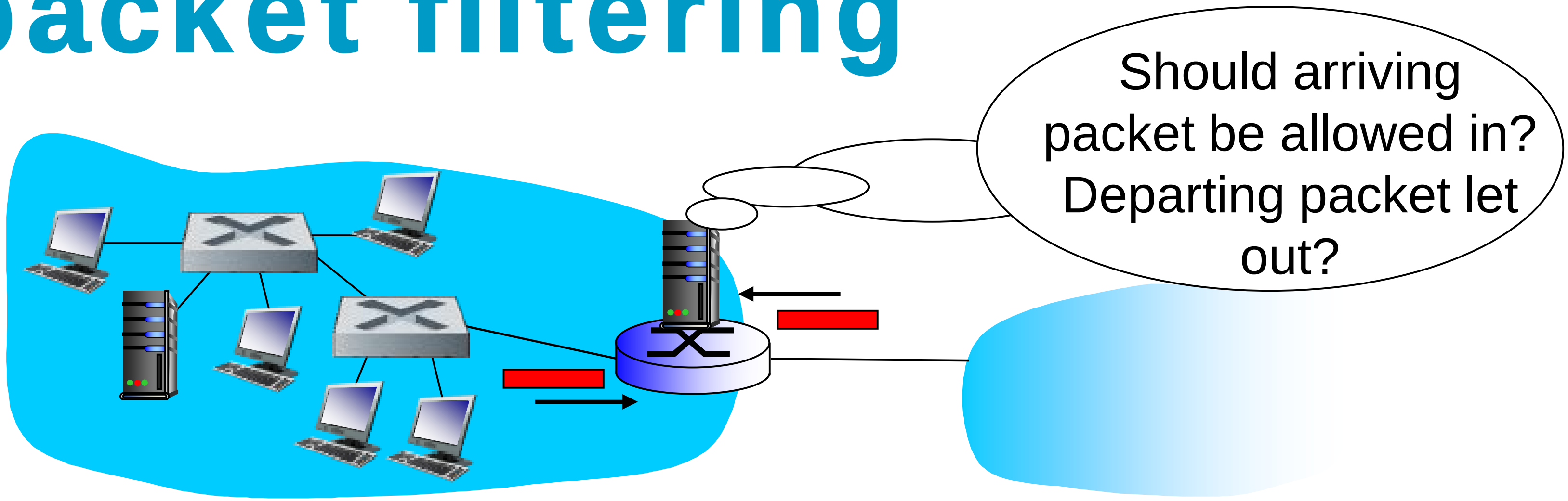
Allow only authorized access to inside network

- Set of authenticated users/hosts

Three types of firewalls:

- Stateless packet filters
- Stateful packet filters
- Application gateways

Stateless packet filtering



Internal network connected to Internet via *router firewall*

Router *filters packet-by-packet*, decision to forward/drop packet based on:

- Source IP address, destination IP address
- TCP/UDP source and destination port numbers
- ICMP message type
- TCP SYN and ACK bits

Stateless packet filtering: example

Example 1: block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23

- *Result*: all incoming, outgoing UDP flows and telnet connections are blocked

Example 2: block inbound TCP segments with ACK=0.

- *Result*: prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

Stateless packet filtering: examples

Policy	*Firewall Setting*
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (e.g. 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

Access Control Lists

ACL: table of rules, applied top to bottom to incoming packets: (action, condition) pairs: looks like OpenFlow forwarding (Ch. 4)!

Action	Source Address	Dest. Address	Protocol	Source Port	Dest. Port	Flag Bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

Stateful packet filtering

Stateless packet filter: heavy handed tool

- Admits packets that "make no sense," e.g., dest port = 80, ACK bit set, even though no TCP connection established:

Action	Source Address	Dest. Address	Protocol	Source Port	Dest. Port	Flag Bit
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

Stateful packet filter: track status of every TCP connection

- Track connection setup (SYN), teardown (FIN): determine whether incoming, outgoing packets "makes sense"
- Timeout inactive connections at firewall: no longer admit packets

Stateful packet filtering

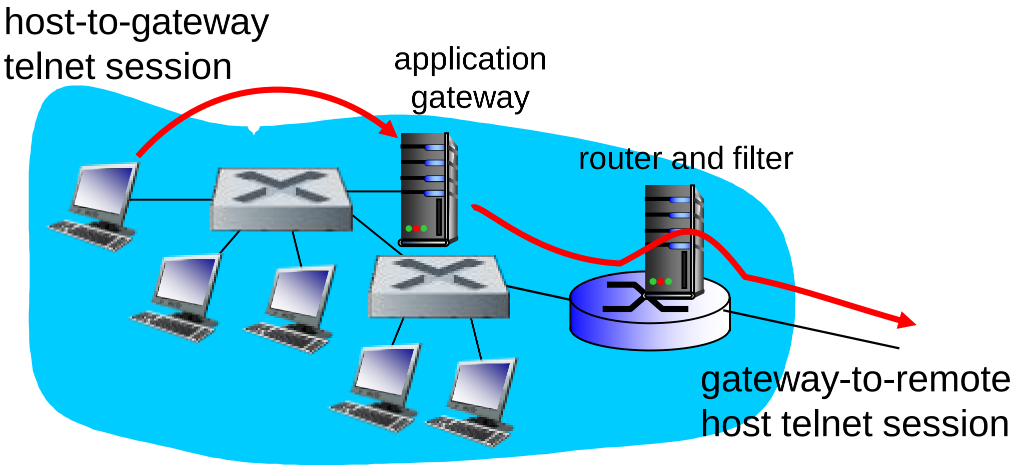
ACL augmented to indicate need to check connection state table before admitting packet

Action	Source Address	Dest Address	Proto	Source Port	Dest Port	Flag Bit	Check Conn.
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any	
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK	x
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---	
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----	x
deny	all	all	all	all	all	all	

Application gateways

Filter packets on application data as well as on IP/TCP/UDP fields.

Example: allow select internal users to telnet outside



1. Require all telnet users to telnet through gateway.
2. For authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
3. Router filter blocks all telnet connections not originating from gateway.

Limitations of firewalls & gateways

IP spoofing: router can't know if data "really" comes from claimed source

If multiple app's. need special treatment, each has own app. gateway

Client software must know how to contact gateway.

- E.g., must set IP address of proxy in Web browser

Filters often use all or nothing policy for UDP

Tradeoff: degree of communication with outside world, level of security

Many highly protected sites still suffer from attacks

Intrusion Detection Systems

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
ENGINEERING

Packet filtering:

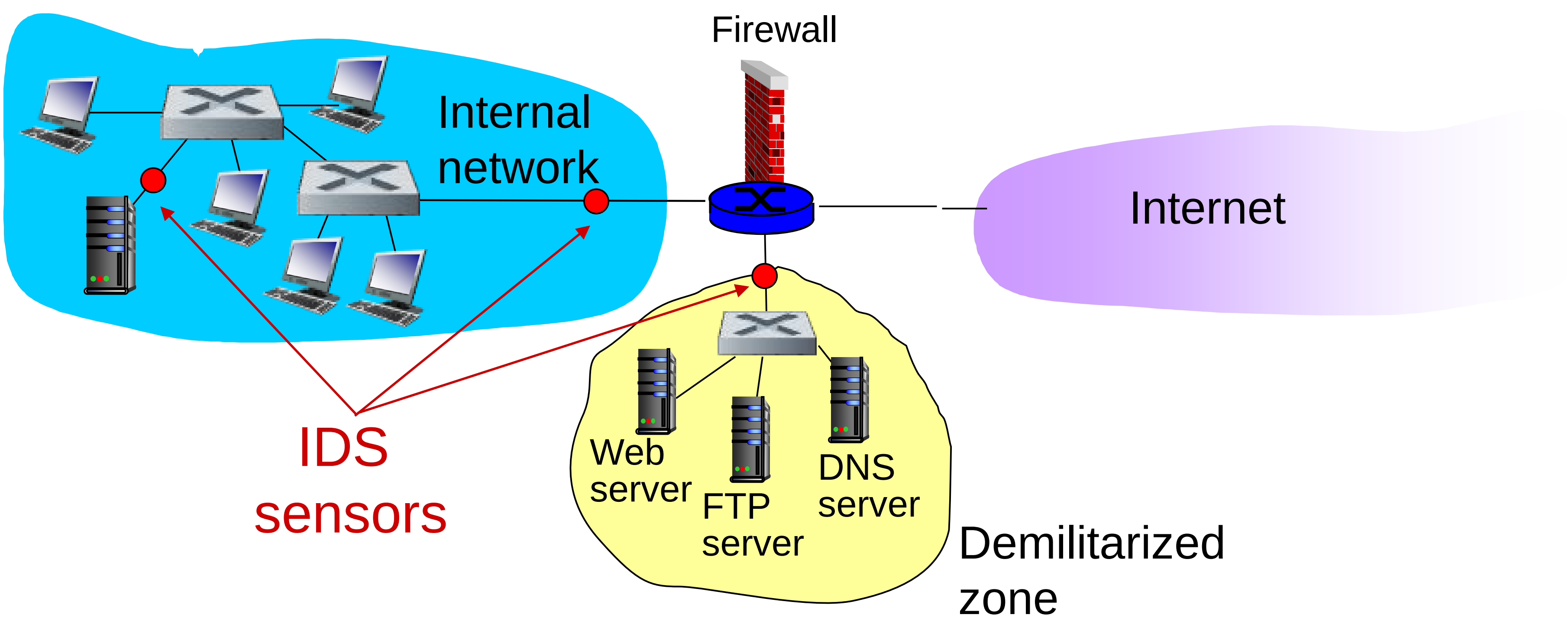
- Operates on TCP/IP headers only
- No correlation check among sessions

Intrusion Detection System (*IDS*)

- *Deep packet inspection*: look at packet contents (e.g., check character strings in packet against database of known virus, attack strings)
- *Examine correlation* among multiple packets
    - Port scanning
    - Network mapping
    - DoS attack

Intrusion Detection Systems

Multiple IDSs: different types of checking at different locations



IDS Detection Types

Network intrusion detection systems (NIDS): A system that analyzes incoming network traffic.

Host-based intrusion detection systems (HIDS): A system that monitors important operating system files.

IDS Limitations

- Noise
- False alarms
- Out-of-date signatures
- Skips encrypted packages
- Fake IP addresses

What we haven’t covered

Authorisation:
- Is the person allowed to perform an action
- Typically application level

IPSec
- Network layer confidentiality
- Protects upper level protocols
- Allows Virtual Private Networks

Wireless security!
- Haven’t covered wireless at all
- WEP: Wired Equivalent Privacy – *avoid!*
- WPA: Wireless Protected Access (and WPA2)

Summary

Defined key terms in network security

Defined the basics of how cryptography works

Introduced symmetric cryptography:

- DES
- AES

Introduced asymmetric cryptography:

- RSA

Digital signatures and cryptographic hashes

Certification authorities

The challenges of authentication

Described how to secure email

Introduced Secure Sockets Layer and its place in network transport

Conceptually built a secure protocol for communications

Described how SSL has been implemented in reality

Introduced firewalls and their purpose

Described three types of firewall

Explained what Intrusion Detection Systems are and what they are used for

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

ENGINEERING