CS 118 Discussion Week 10: Mobility and Security

Slides by Eric Newberry, UCLA

Winter 2021

Reminders and Announcements

- Project 2 and Homework 4 are due at 11:59pm today!
  - Please turn something in for partial credit even if it doesn't check every box!
  - "Triage" your remaining time – get regular forwarding working before trying ICMP (and save a copy of your code before this point as a backup)
  - We are allowing late submissions through Sunday at 11:59pm Pacific (note that daylight saving time begins Sunday as well!)
  - Lateness penalty is -15% per day or partial day late.
- Course evals are due tomorrow at 8am PST
- Final exam will be assigned on Thursday, March 18
  - Similar format to midterm (X hours to do it within a 24-hour period)

General Mobility Approaches

- Cellular
  - Register with home carrier → tracks your general location
  - When visiting other carrier network w/ agreement with your home carrier
    - Register using your home network credentials
- Traditional computing environments
  - No concept of "home" network for your average laptop
  - Have to use different credentials to connect to each network (e.g., WiFi passwords)
    - Sometimes there is unified authentication infrastructure, e.g., Eduroam

Mobility Approach: Indirect Routing

- As you move, register your current location (IP address) with your "home network"

- Senders will send data to home network

- Then, home network will forward data on to your current location

- When you respond to sender, send packet directly to them
    - Use your home network address as source address
- Also called "triangle routing"
- A bit inefficient since traffic must be forwarded twice
- But transparent (and therefore easier) to outside senders

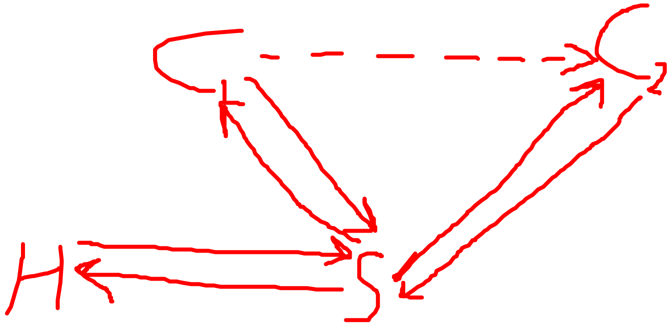
Mobility Approach: Direct Routing

- When sender attempts to communicate with a mobile host, host's home network will inform of host's current IP address

- Sender will then send traffic directly to host's current IP address

- More efficient routing (send directly instead of indirectly)
- However, sender must learn location of mobile host

- Additionally, if mobile host moves, correspondent must be able to respond by getting new mobile host IP address



Securing Computer Networks

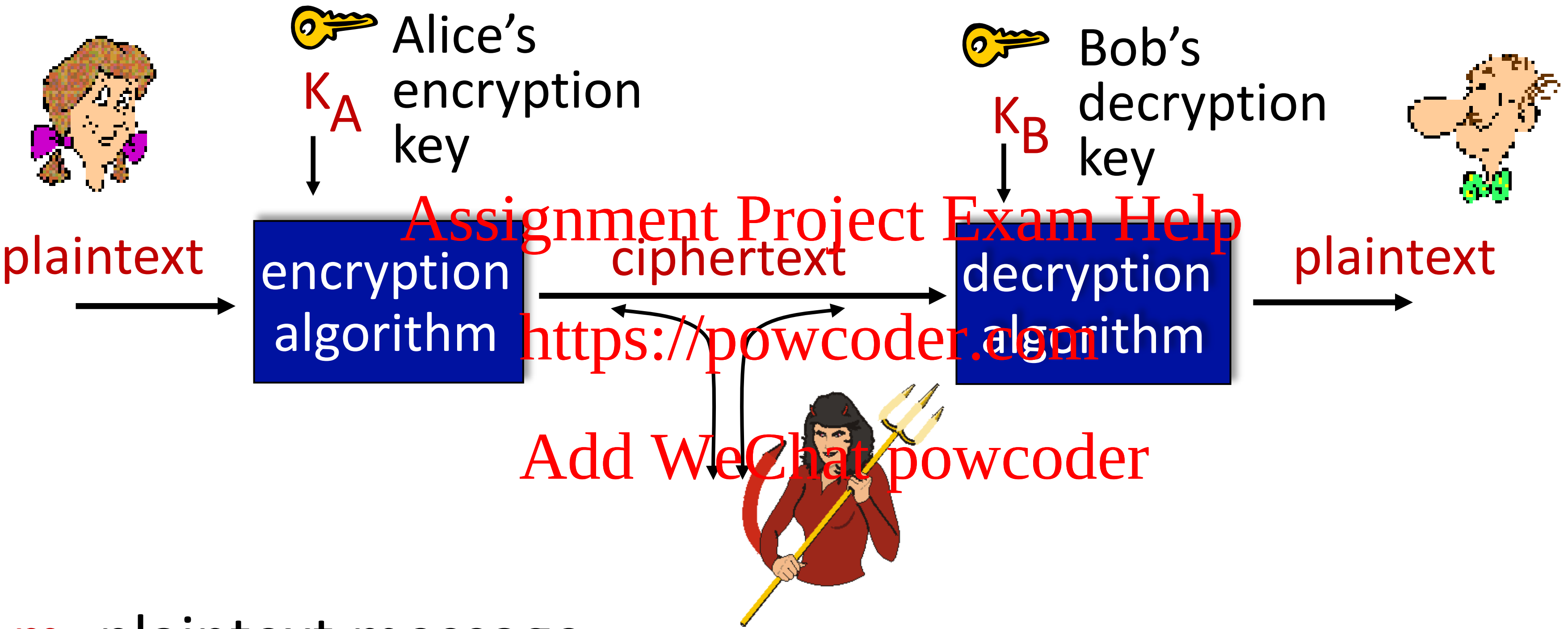
"CIA" – The Core Principles of Security

- Confidentiality
  - Only sender and receiver(s) should be able to known message contents

- Integrity
  - Message should not be able to be surreptitiously altered in transit

- Availability
  - Users must be able to use services
- (not a principle, but important) Authentication
  - Sender and receiver should be able to verify each other's identities

Symmetric Key Cryptography



m: plaintext message

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

$m = K_B(K_A(m))$

From slides by Kurose & Ross

Symmetric Key Cryptography

- Same key is used to encrypt and decrypt
- “Substitution Cipher”
  - Both parties have a pre-shared substitution table
  - Sender uses table to substitute letters one way
  - Receiver uses table to reverse substitution

Input	Output
A	Z
B	Y
C	X
D	W
E	V
F	U
…	…

Substitution Cipher

Encryption: Shift by two letters to the right

Decryption: Shift by two letters to the left ("symmetric")

Plaintext: T W O P L U S T W O E Q U A L S F O U R

Ciphertext:

Ciphertext: G V J G T P G V

Plaintext: E T H E R N E T

More Complex Symmetric Cryptograhpy

- Data Encryption Standard (DES)
  - Small key size (56-bits)
  - Very insecure with modern processing speeds
- Advanced Encryption Standard (AES)
  - Key size: 128-bits, 192-bits, or 256-bits
- Comparison:
  - Brute force DES key in approx. 1 second
  - Brute force AES key in approx. 149 x 10

Public Key Cryptography

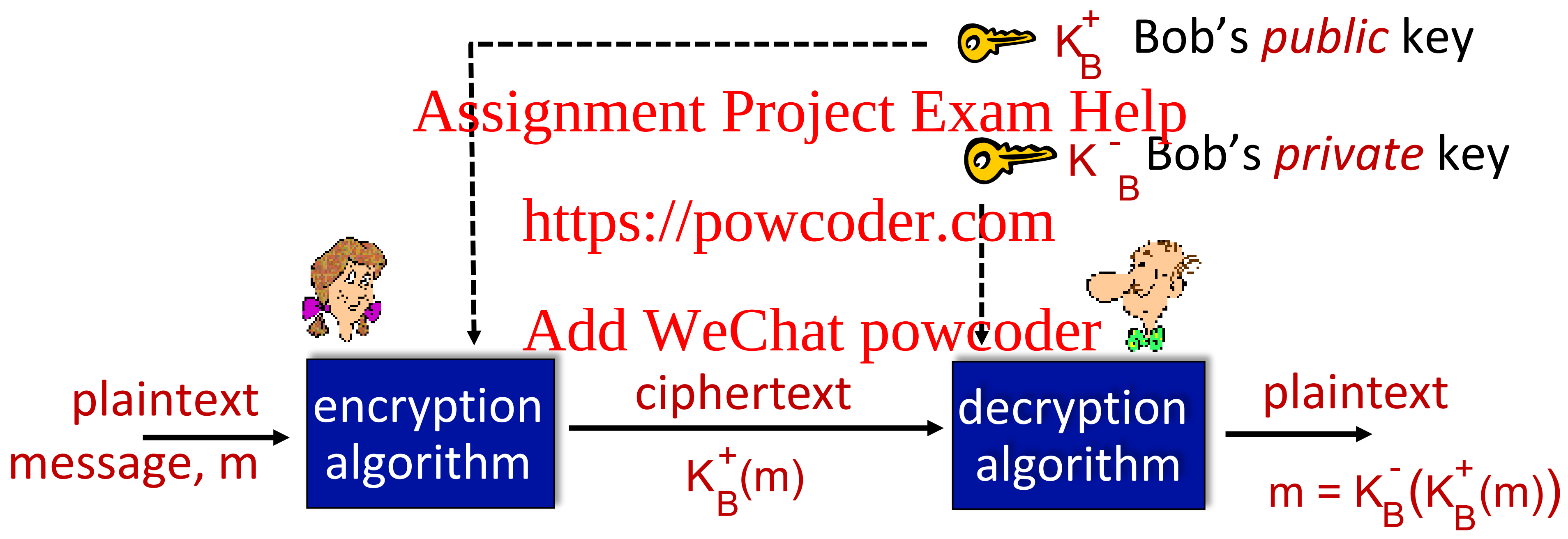
- Symmetric has one key for both encrypting and decrypting
- Instead, use a different key for each function!

- Give out public key, which can only encrypt

- Keep safe private key, which can only decrypt

- => Anyone can encrypt data to send to you, only you can decrypt it
- (Side note: digital signatures use reverse: sign w/ private key, validate w/ public key)

Public Key Cryptography



From slides by Kurose & Ross

Public Key vs. Symmetric

- Public key never needs to “move” a secret key to the other end
  - Meanwhile, need a mechanism to securely share the secret key in symmetric

- Public key keeps communication between all pairs of parties secret (only recipient can decrypt communications directly to them)

  - Meanwhile, anyone with the key can decrypt in symmetric

- However, public key is significantly slower (more mathematically complex)
- Real world solution: use public-key to securely share a symmetric key
  - Then use this symmetric key for the communication session
  - RSA!

RSA: Rivest-Shamir-Adelson

- How do we construct a key pair so that the public key cannot be used to compute the private key?
- Essentially:
  - Choose two very large (e.g., 1024-bit) prime numbers p and q
  - Compute $n = pq$, $z = (p-1)(q-1)$
  - Choose $e<n$ s.t. e,z are relatively prime (no common factors)
  - Choose d s.t. $ed-1$ is divisible by z (ed mod $z = 1$)
- From these, the public key is (n,e) and the private key is (n,d)
- Security comes from difficulty of factoring very large prime numbers
  - However, quantum computing is making this easier every day…

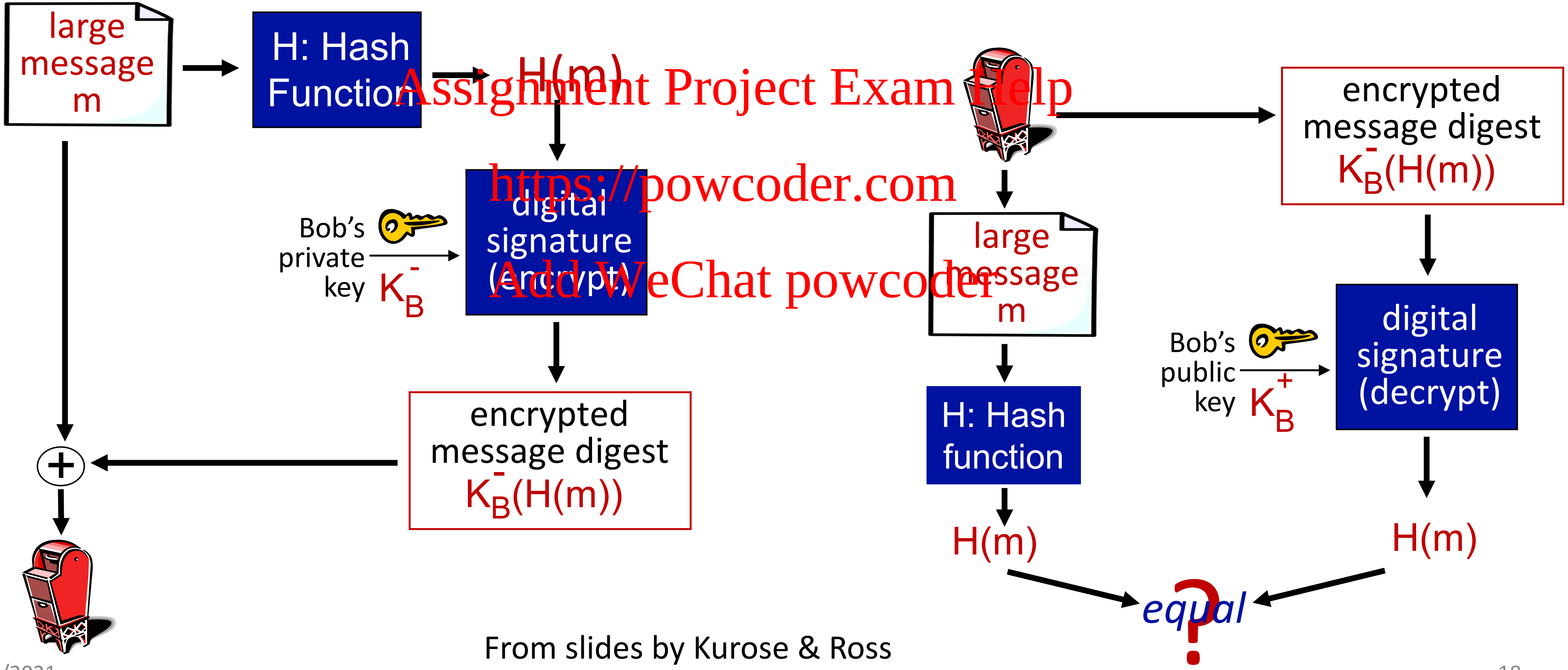
Realistic RSA

- Actually encrypting and decrypting data with RSA is very computationally expensive
- Instead, create a public and private key pair with RSA
- Then, use to securely share a symmetric session key
- Only use this key for this session
  - Breaking key of one session doesn't break any other sessions

Hashing

- How do we verify that message as not modified (whether maliciously or through an error) in transit?
- Use a hash function!
  - Generates a small “fingerprint” from some large input document
  - Goal is for it to be difficult to find another message that hashes to same value
  - Can then use public key encryption to sign this hash (cheaper than signing large message)
- Common hash functions:
  - MD5 (no longer secure – too easy to break)
  - SHA1 (no longer secure – too easy to break)
  - SHA256 (secure! for now…)

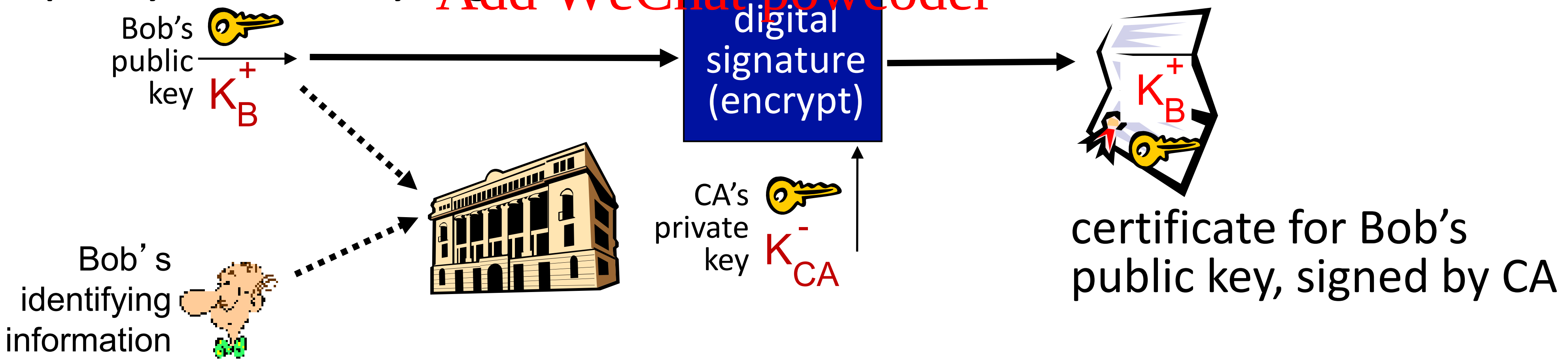
Hashing



From slides by Kurose & Ross

Public Key Infrastructure

- How do we know that we're talking to the right person/app ("entity")?
- "Certificate Authority" (CA) that is trusted by both sender and receiver creates a signature over entity's key to prove identity
- Other party can verify signature



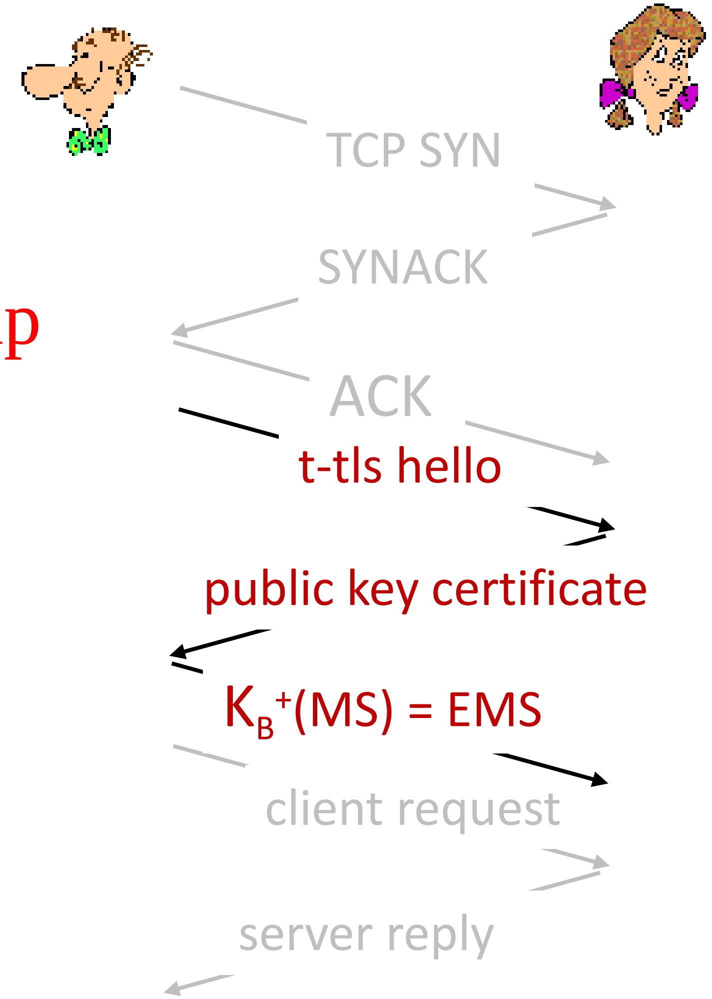
From slides by Kurose & Ross

TLS: Transport-Layer Security

- Protocol above transport layer to secure application-layer data
- Used by protocols such as HTTPS, IMAP, SMTP, SSH, etc. (most secure application-layer protocols)
  - Replaced SSL, which was deprecated in 2015
- Combination of:
  - Symmetric key encryption (to provide data confidentiality)
  - Cryptographic hashing (to provide data integrity)
  - Public key cryptography (to provide data authentication)

Classic TLS in Action

- We have a handshake like TCP
  - Happens after TCP handshake
- Client sends hello to server
- Server sends back certificate to verify identity
- Client sends the “master secret” (MS) key
  - Encrypted using server’s public key
  - This key is used to generate other keys over the span of this session
- However, is a bit slow (3 RTTs before data exchange can occur)



From slides by Kurose & Ross

TLS in Action

- Data is encrypted *and* hashed (latter is known as message authentication code or “MAC”)

- TCP sends data as an endless “stream”, but we can only encrypt data in finite blocks

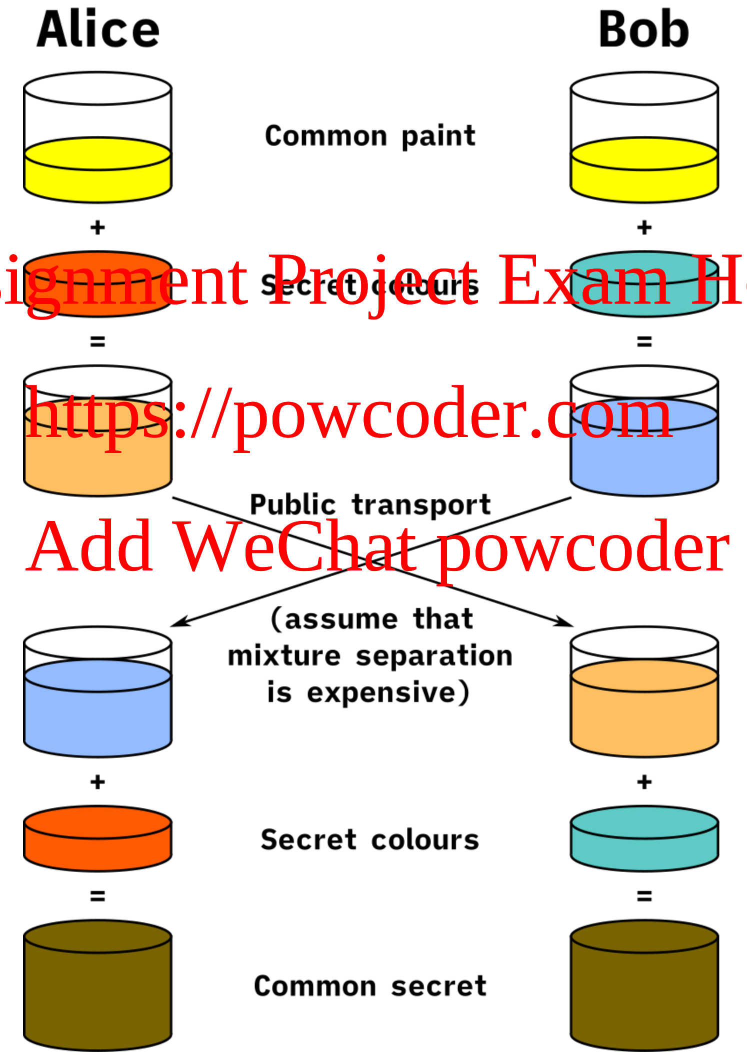
  - Solution: Break up stream of data into finite-sized “records”

- How to avoid reordering and replay attacks?
  - Use sequence numbers (included in data hashed in MAC)
  - Use a nonce (random value) to change MAC values
- How to avoid truncation attacks (closing connection)?
  - Use a special message type to close (include type in data hashed in MAC)

TLS 1.3

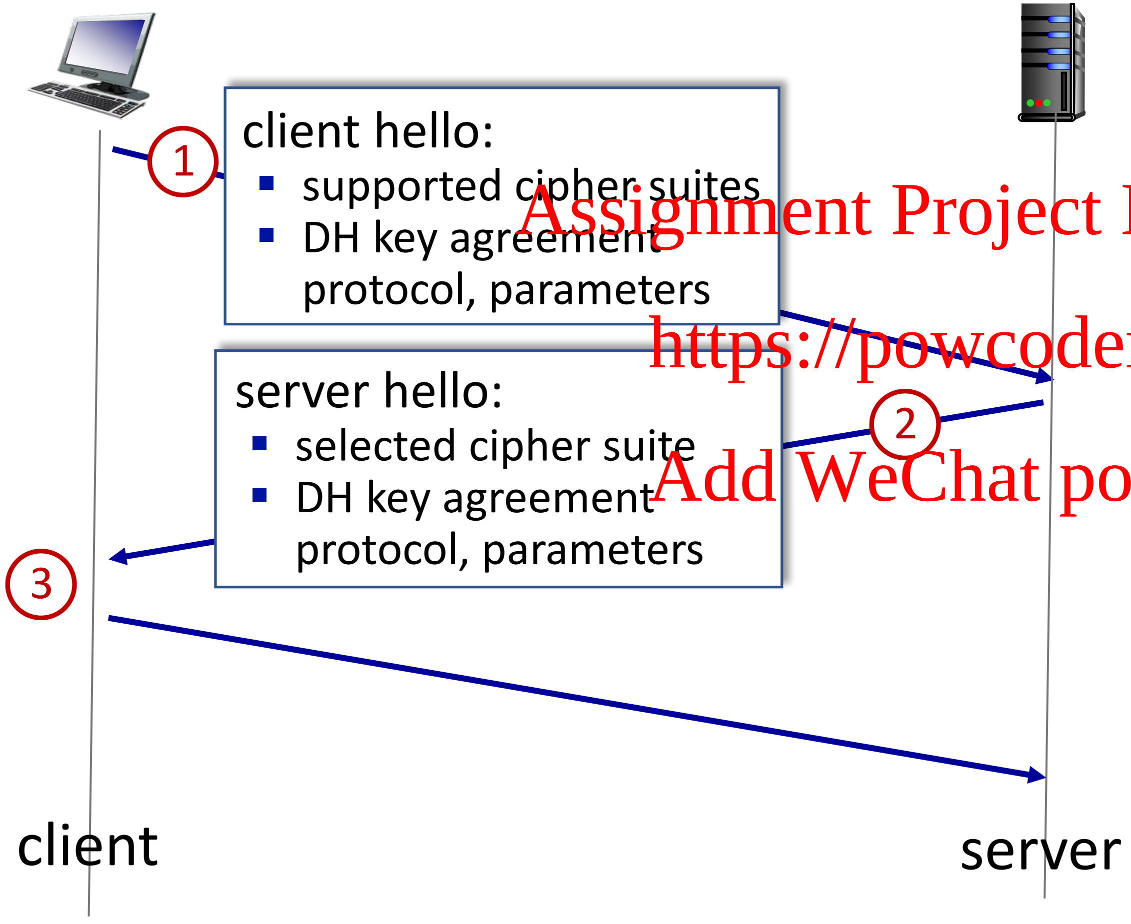
- Improve upon TLS 1.2
- Simplicity: Number of cryptographic ciphers reduced from 37 to 5
- Simplicity: Require Diffie-Hellman (DH) instead of RSA
- Security: Require SHA256 or SHA284 cryptographic hash function
- Efficiency: Use combined encryption and authentication algorithm instead of encrypting and then authenticating
- Efficiency: 1-RTT and 0-RTT handshakes

Diffie-Hellman Key Exchange



https://commons.wikimedia.org/wiki/File:Diffie-Hellman_Key_Exchange.svg

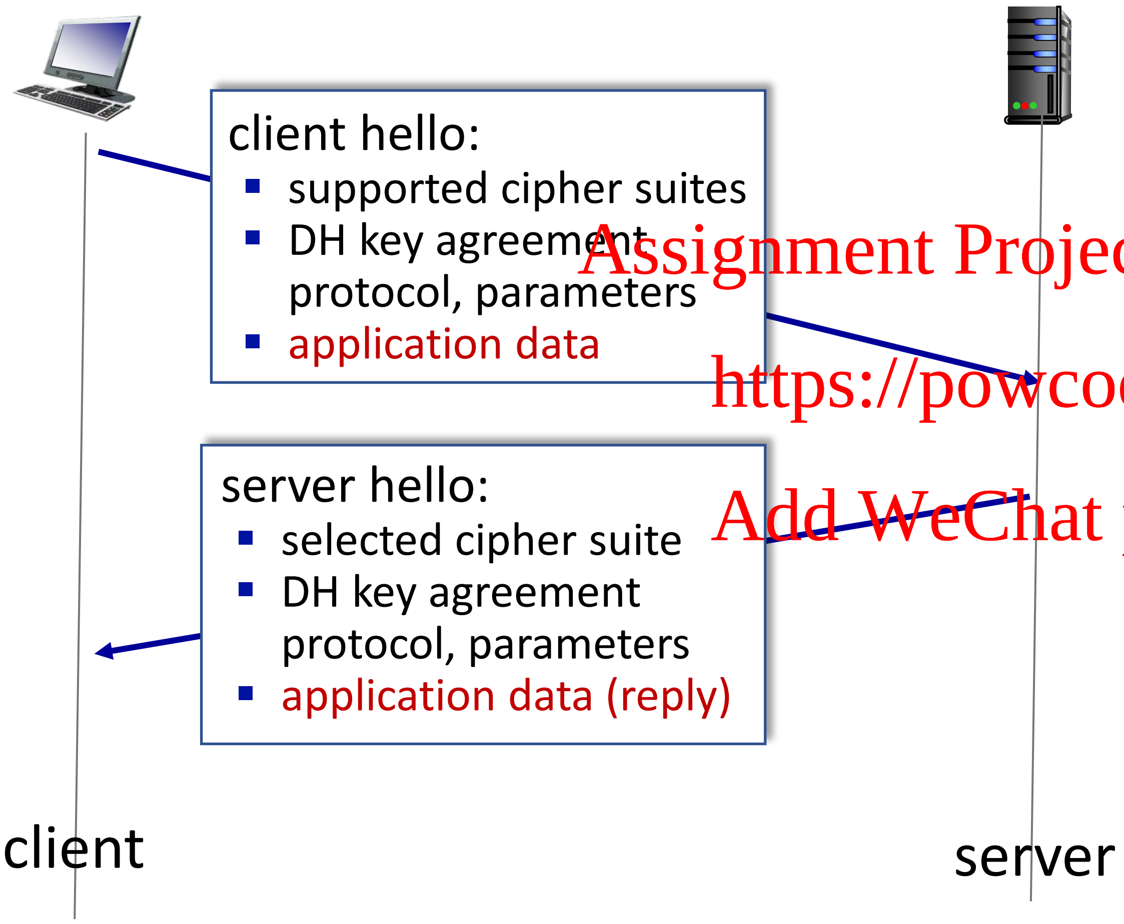
TLS 1-RTT Handshake



- Data can actually be sent in packet 3

From slides by Kurose & Ross

TLS 0-RTT Handshake



- Send application data in client hello message!
- However, now vulnerable to replay attacks!
  - Only really suitable for requests that don't modify server state
  - E.g., HTTP GET requests

From slides by Kurose & Ross

IPsec: Securing the Network Layer

- Encrypt IP datagrams directly
- Two modes:
  - "Transport mode": Only encrypt payload, headers still visible to passing hosts
  - "Tunnel mode": Encrypt entire datagram and encapsulate in another IP datagram when entering "tunnel", then decapsulate and decrypt at end of tunnel
- Two protocols:
  - Authentication Header (AH) protocol: authentication, integrity, but not confidentiality
  - Encapsulation Service Protocol (ESP): authentication, integrity, **and** confidentiality

IPsec Security Associations (SAs)

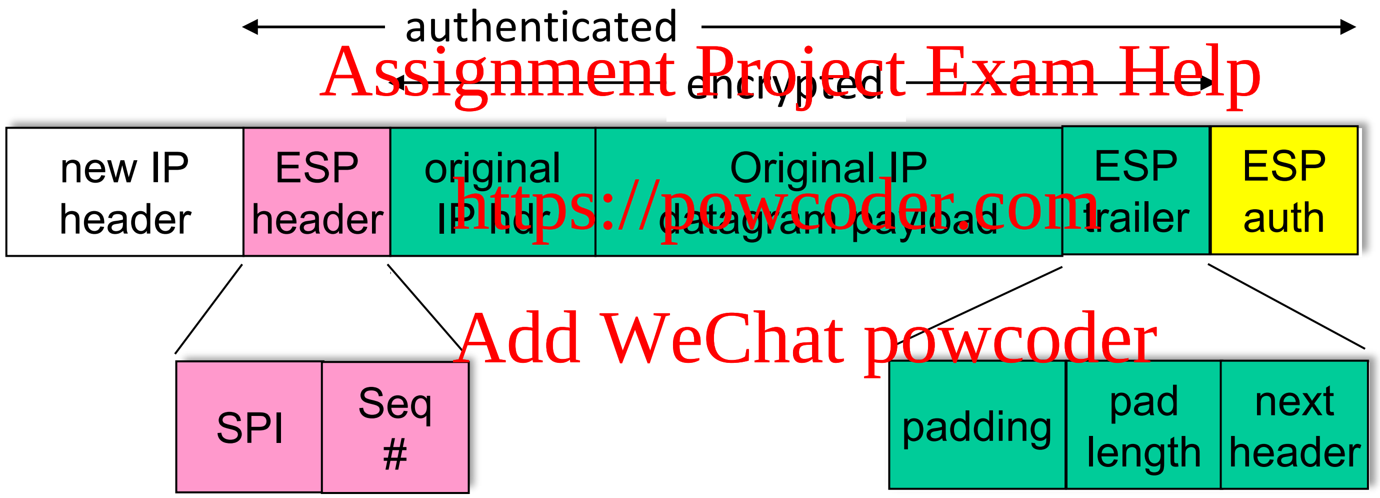
- IPsec is a stateful protocol, unlike IP!
- Need to establish state (Security Association) from sender to receiver

- What each endpoint stores:

  - Security Parameter Index (SPI) – 32-bit identifier of association
  - Origin host (sender)

  - Destination host (receiver)
  - Encryption type and key
  - Integrity validation mechanism
  - Authentication key

IPsec ESP Tunnel Mode Datagram



From slides by Kurose & Ross

Securing Wireless Networks

- Have you ever been asked for a password when you connected to a wireless network?

- When we connect to a wireless network, we must both associate **and** authenticate to the wireless network

- WiFi encryption is optional, but is generally used on most networks

  - Otherwise anyone could eavesdrop on your traffic!

802.11 (WiFi) Authentication and Encryption

- First, wireless access point (AP) advertises itself with a periodic “beacon” message
  - Also contains information about required security mechanisms of network
- Device tries to connect to AP, requesting specific security mechanisms from those available
- AP and device authenticate each other using shared secret, hashing, and nonces
- AP and device derive a symmetric session key
- Then, can proceed with encrypted communications

Firewalls

- Enforce security policies by selectively allowing, blocking, or modifying passing traffic
- Often sit between "trusted" (e.g., corporate) and "untrusted" (e.g., the Internet) networks
- Most often used to filter incoming traffic
  - But can also be used to block data from *leaving* the network
- Filtering can be stateful or stateless

Firewalls: Stateless Packet Filtering

- Examine each passing packet independently
- Apply firewalls rules to forward/drop packets based upon information in packet, such as:
  - Source and destination addresses
  - Source and destination ports
  - Protocol type (e.g., TCP, UDP, ICMP)
  - TCP bits set

Firewalls: Stateful Packet Filtering

- Like stateless filtering, but track TCP connections
- Can be used to, e.g., make sure that TCP connections are set up properly
- Or, e.g., prevent further communications on TCP connections that have been inactive for a while
- Requires more computational resources and stateful storage of connection status

Intrusion Detection Systems

- Perform packet filtering like firewalls, but perform “deep packet inspection”
- Look for evidence of known attack patterns
- Can be used to look at application-layer contents of packet
  - E.g., examine database queries for SQL injection attacks