Security Part I: Crypto Basics

15-441/641 Spring 2019

Profs Peter Steenkiste & **Justine Sherry**

Carnegie Mellon University

Midterms will be returned on Tuesday.

Normal Mindset

- No user would do that
- The odds of a router being misconfigured that way is too small to worry about

Security Mindset

- The **adversary** will do anything it can to break your system
- It will study your system and purposefully do the worst thing it can
- Might even disregard its own well being
- Will attack your implementation and your assumptions

How would you overwhelm my mailbox with letters?

How should I or the postal service protect against the attacks you considered?

What would you do after that?

What if you wanted to read my letters — but didn’t want me to know?

How should I or the postal service protect against the attacks you considered?

What other “attacks” might you leverage against the postal system?

Adversaries

- Possible adversaries include:
  - Competitors trying harm you
  - Governments trying to control you
  - Criminals who want to use your system for crime
  - Disgruntled employees (the *insider threat*)
  - Hackers who find it fun to break stuff
  - Others we didn't even think of …
- Assumptions about the adversary are dangerous
- Security is very hard

Unlimited resources

Knows your source code

Destructive with no "real" goals

“DARPA Internet Design Goals”

1. Interconnection
2. Failure resilience
3. Multiple types of service
4. Variety of networks
5. Management of resources
6. Cost-effective
7. Low entry-cost
8. Accountability for resources

Where is security?

Why did they leave it out?

- Designed for connectivity

- Network designed with implicit trust
  - Origin as a small and cooperative network
  - No “bad” guys (adversaries)

- Can’t security be provided at the edge?
  - Encryption, Authentication etc
  - End-to-end arguments in system design

Many of you have already noticed some security problems that snuck in to the Internet's design…

Internet Design Decisions and Security

- Connection-less datagram service
    - (=> can't verify source, hard to protect bandwidth)

Internet Usage and Security

- Anyone can connect (=> ANYONE can connect)
- Millions of hosts run nearly identical software (=> single exploit can create epidemic)
- Most Internet users know about as much as Senator Stevens aka “the tubes guy”
  (=> help us all…)

The problem of anyone

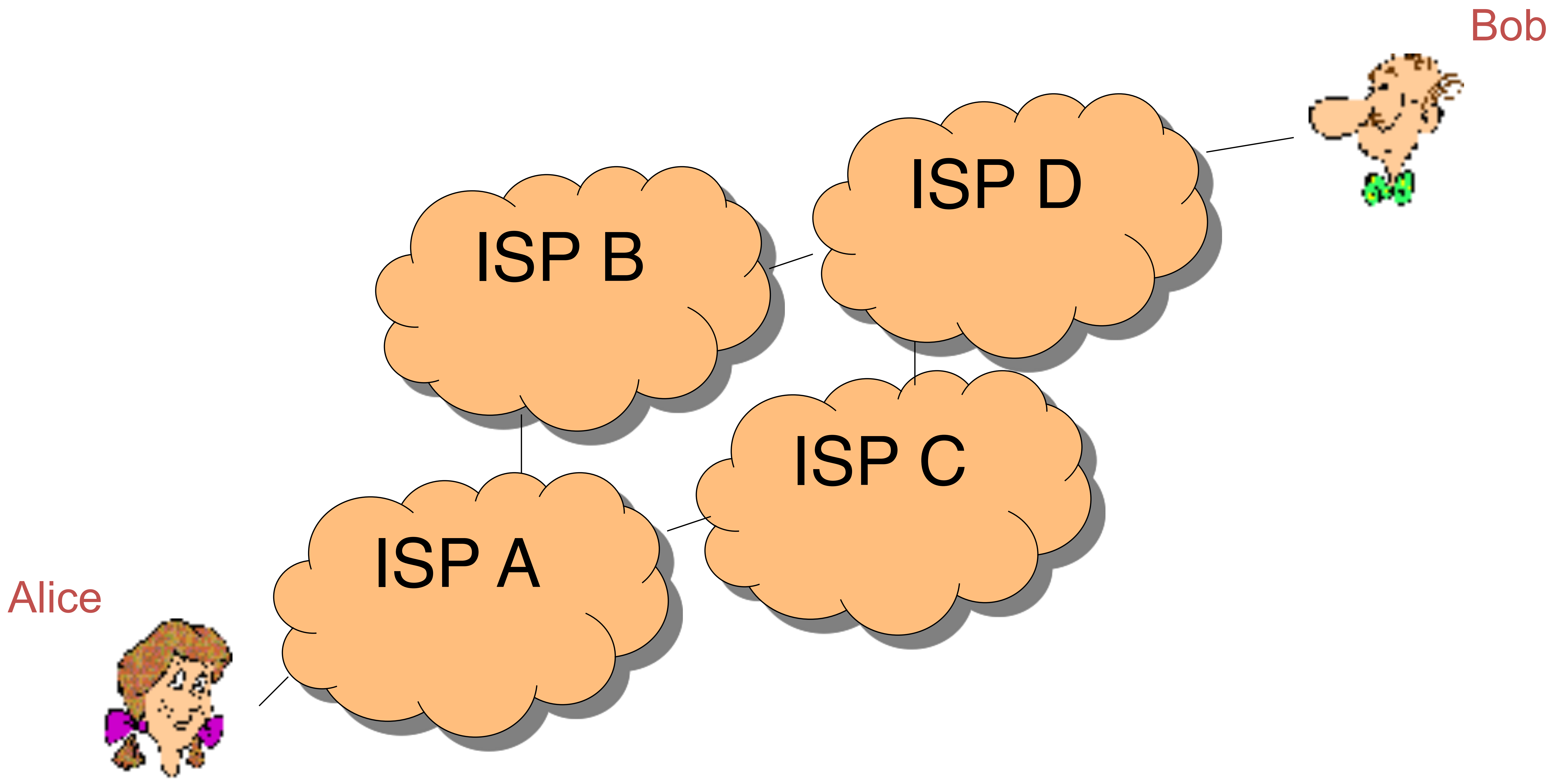
- The Internet — unlike other systems — allows *anyone* to use it.
  - Is this agent (IP address, connection, user) allowed to access this server?
  - Are they who they say they are?
  - Is this data from who I think it is from? Has it been read or modified?

Our “Narrow” Focus in Networking

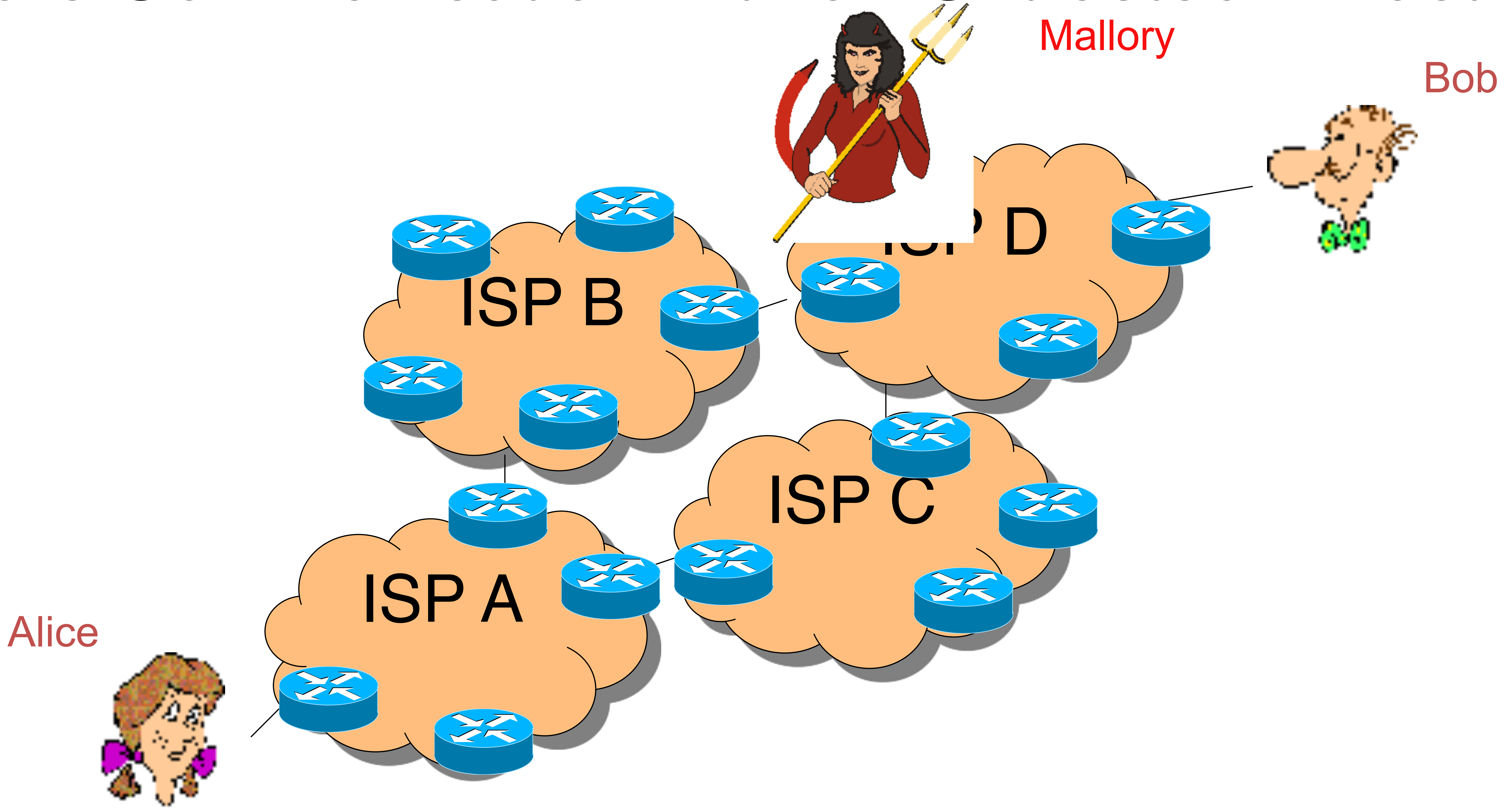
- Yes:
  - Creating a “secure channel” for communication (Part I)
    - End-to-end
  - Protecting network resources and limiting connectivity (Part II, III)
    - Accountability for resources (largely not end-to-end)

- No:
  - Preventing software vulnerabilities & malware, or “social engineering”.

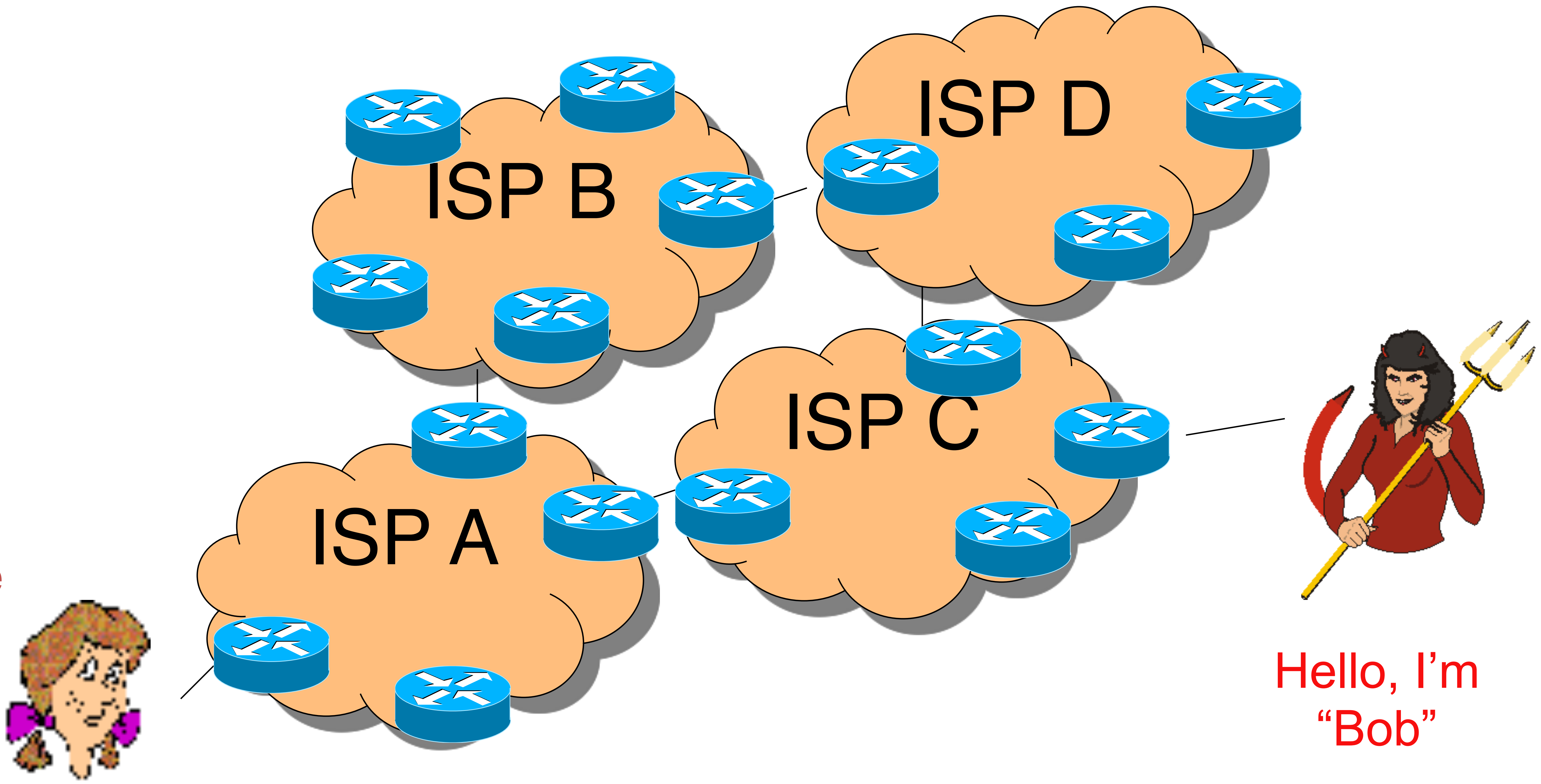
Secure Communication with an Untrusted Infrastructure



Secure Communication with an Untrusted Infrastructure



Secure Communication with an Untrusted Infrastructure



What do we need for a secure comm channel?

- Authentication (Who am I talking to?)
- Confidentiality (Is my data hidden?)
- Integrity (Has my data been modified?)
- Availability (Can I reach the destination?)

When you go to the bank, how do they implement authentication?

When you go to the bank, how do they implement confidentiality?

When you go to the bank, how do they implement integrity?

What is cryptography?

"cryptography is about communication in the presence of adversaries."

- Ron Rivest

“cryptography is using math and other crazy tricks to approximate magic”

- Unknown 441 TA

What is cryptography?

Mathematical tools to help us build secure communication channels that provide:

1) Authentication

2) Integrity

3) Confidentiality

ITS MAGIC
memegen.com

Cryptography As a Tool

- Using cryptography securely is not simple
- Designing cryptographic schemes correctly is so hard it's near impossible.

Today we want to give you an idea of what can be done with cryptography.

Go talk to Professor Goyal (https://www.cs.cmu.edu/~goyal/) or take a security course if you want to know more about crypto!

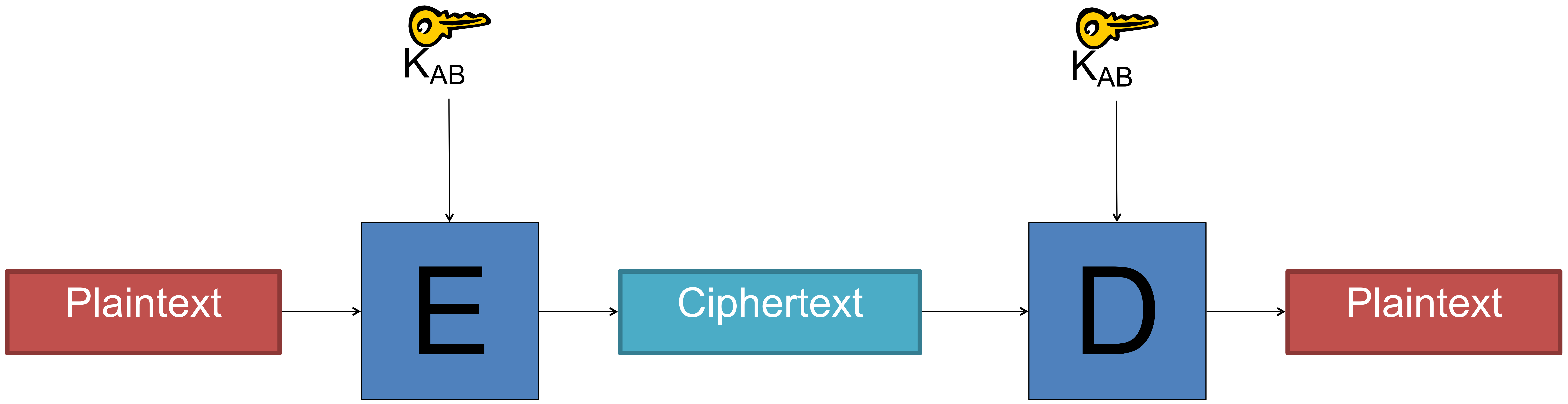
The Great Divide

	Symmetric Crypto (Private key) (E.g., AES)	Asymmetric Crypto (Public key) (E.g., RSA)
Shared secret between parties?	Yes	No
Speed of crypto operations	Fast	Slow

Cryptography Overview

	Symmetric	**Asymmetric**
Confidentiality		
Integrity		
Authentication		

Symmetric Key: Confidentiality



Symmetric Key: Confidentiality

One-Time Pad

<u>Motivating Example:</u>

You and a friend share a key K of L random bits, and want to secretly share message M also L bits long.

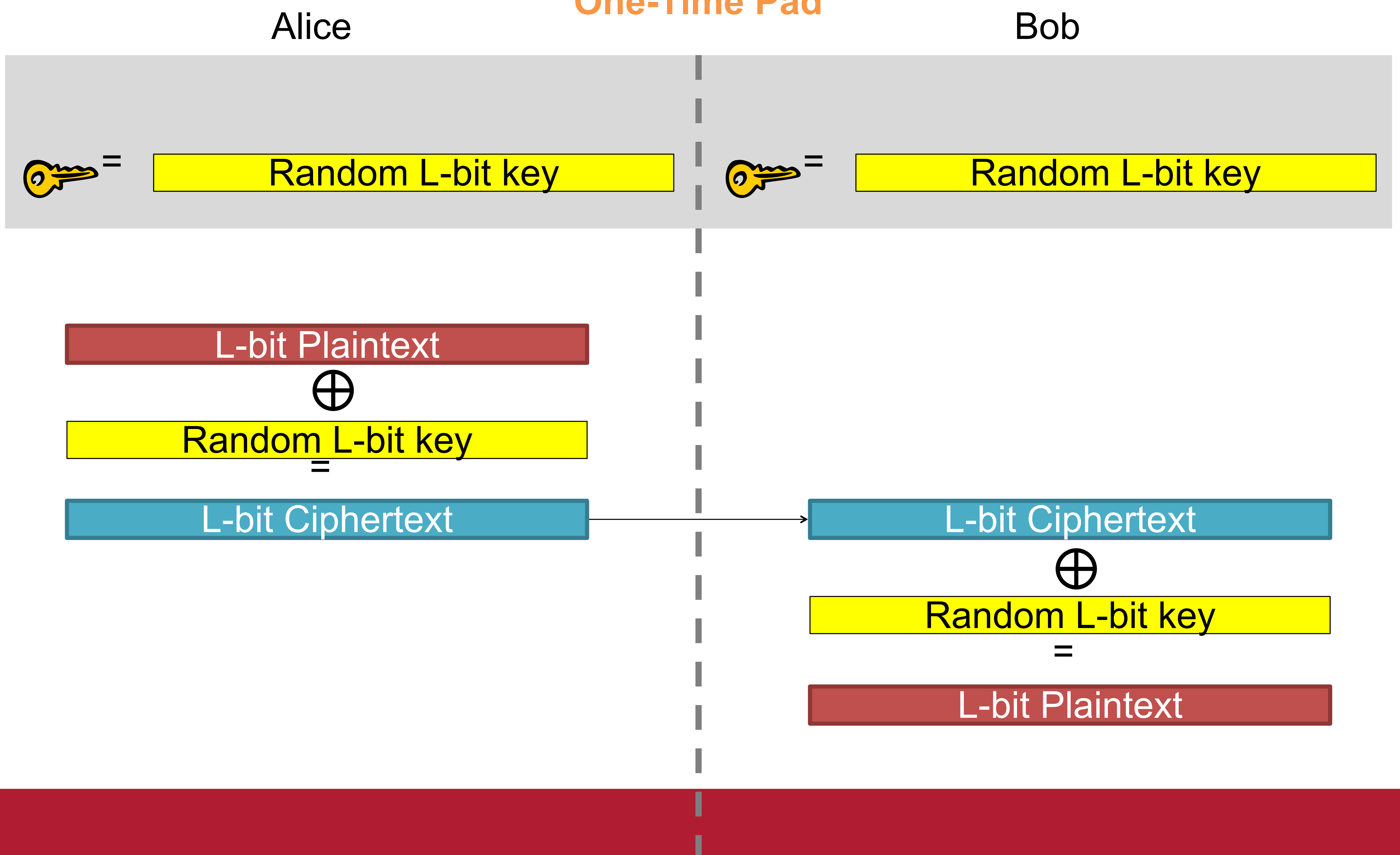
<u>Scheme:</u>

You send her the *xor(M,K)* and then she “decrypts” using *xor(M,K)* again.

1) Do you get the right message to your friend?
2) Can an adversary recover the message M?
3) Can adversary recover the key K?

Symmetric Key: Confidentiality

One-Time Pad



Symmetric Key: Confidentiality

One-Time Pad

SECURE?

•Yes! One-time Pad (OTP) is proven "information-theoretically secure" (Claude Shannon, 1949)

- Leaks no information about the message other than its length

BUT

•Assumptions:

- Perfectly random one-time pads (keys)
- One-time pad at least the length of the message
- Can *never* reuse a one-time pad
- Adversary can never know the one-time pad

Symmetric Key: Confidentiality

One-Time Pad

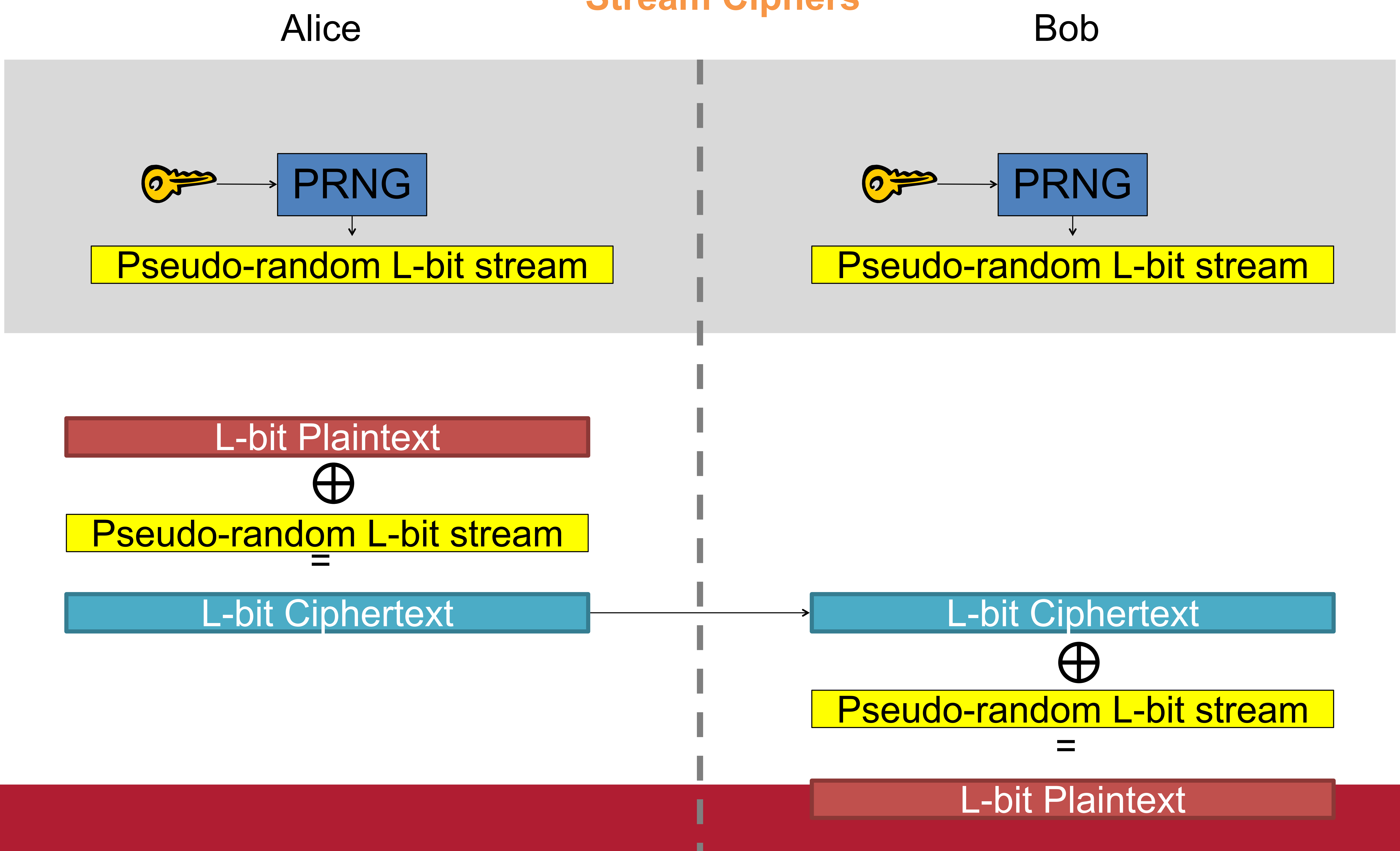
Symmetric Key: Confidentiality

- All ciphers suffer from assumptions, but one-time pad's are impractical to maintain
  - Key is as long at the message
  - Keys cannot be reused

- In practice, two types of ciphers are used that require constant length keys:
  - **Stream Ciphers**
    Ex: RC4, A5
  - **Block Ciphers**
    Ex: DES, AES, Blowfish

Big Idea: Small amount of shared random info and use a deterministic function to generate the rest

Symmetric Key: Confidentiality


Stream Ciphers
Alice
Bob
PRNG
Pseudo-random L-bit stream
PRNG
Pseudo-random L-bit stream
L-bit Plaintext
⊕
Pseudo-random L-bit stream
=
L-bit Ciphertext
L-bit Ciphertext
⊕
Pseudo-random L-bit stream
=
L-bit Plaintext

Symmetric Key: Confidentiality

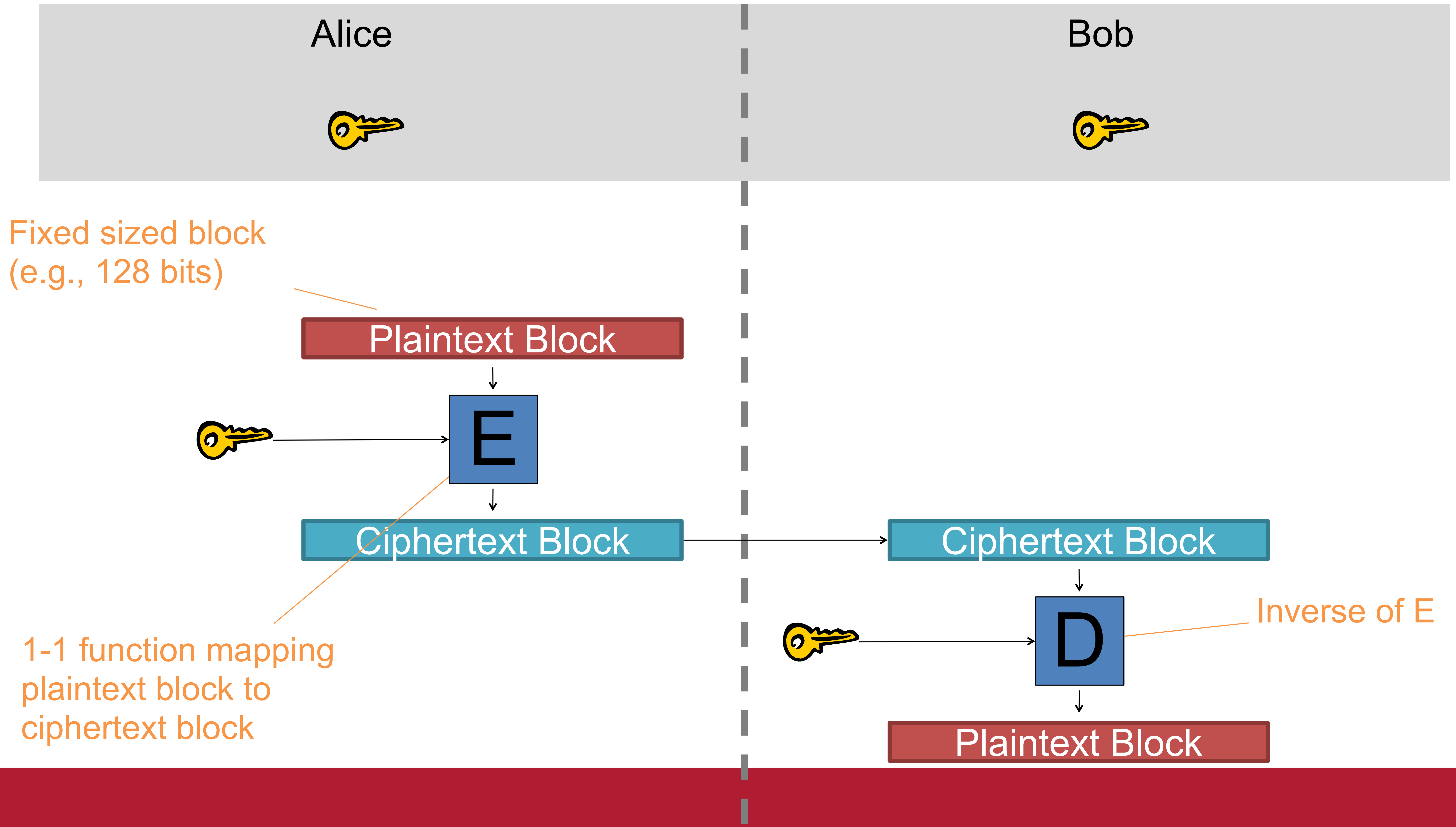
Stream Ciphers

SECURE?

- Key stream reuse attack
  - $C_1 = P_1 \oplus K$ and $C_2 = P_2 \oplus K$ where $K = \text{PRNG}(\text{key})$
  - $C_1 \oplus C_2 = P_1 \oplus P_2$
  - Easier to analyze since random key is removed
  - Solution: **Initialization Vector**. $K = \text{PRNG}(\text{key}, \text{IV})$
- Ciphertext modification attack
  - Flipping a bit in ciphertext results in the corresponding bit flipped in decrypted plaintext
  - Particularly dangerous if attacker knows message format/content
  - Solution: Check integrity.

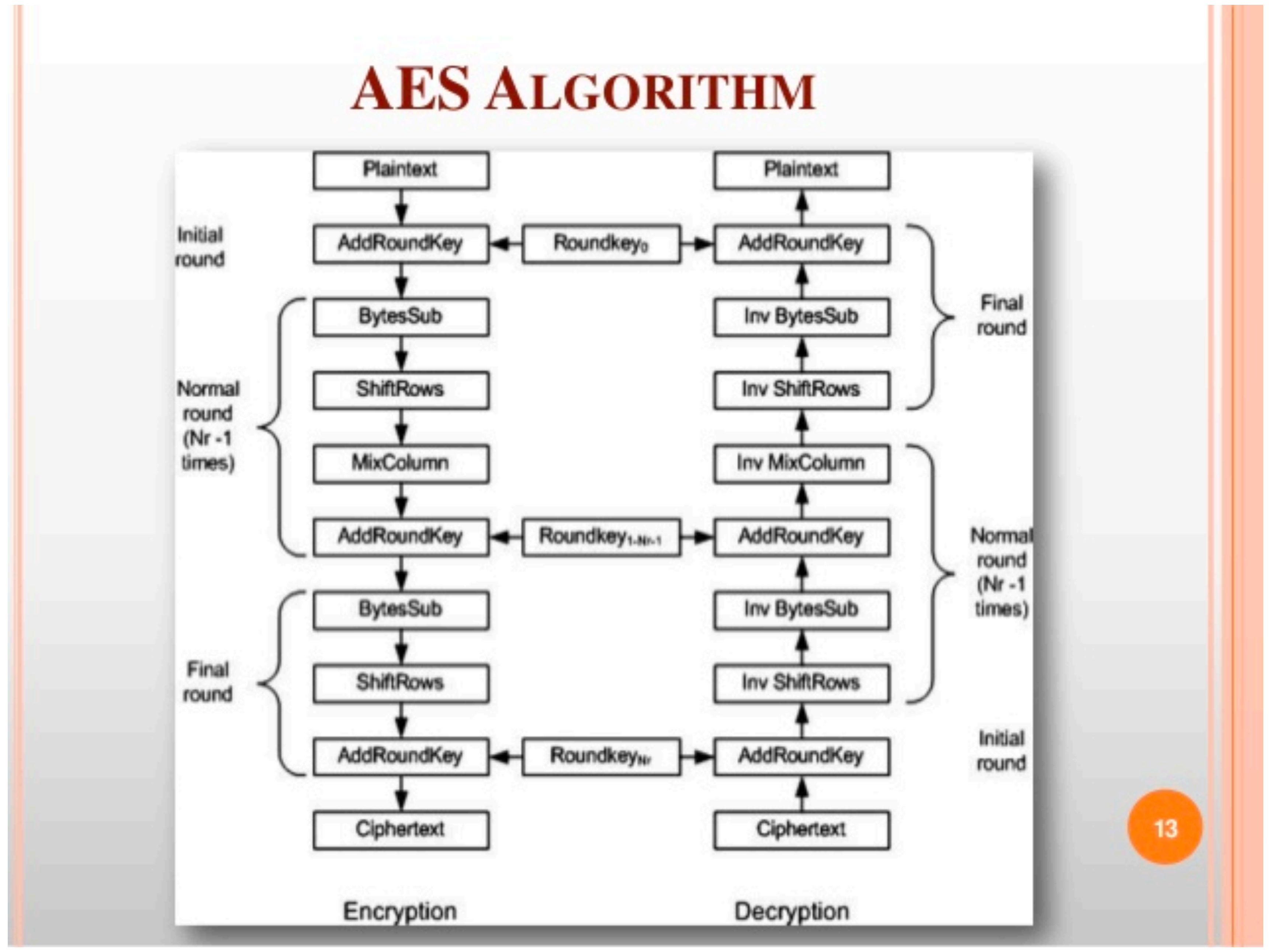
Symmetric Key: Confidentiality

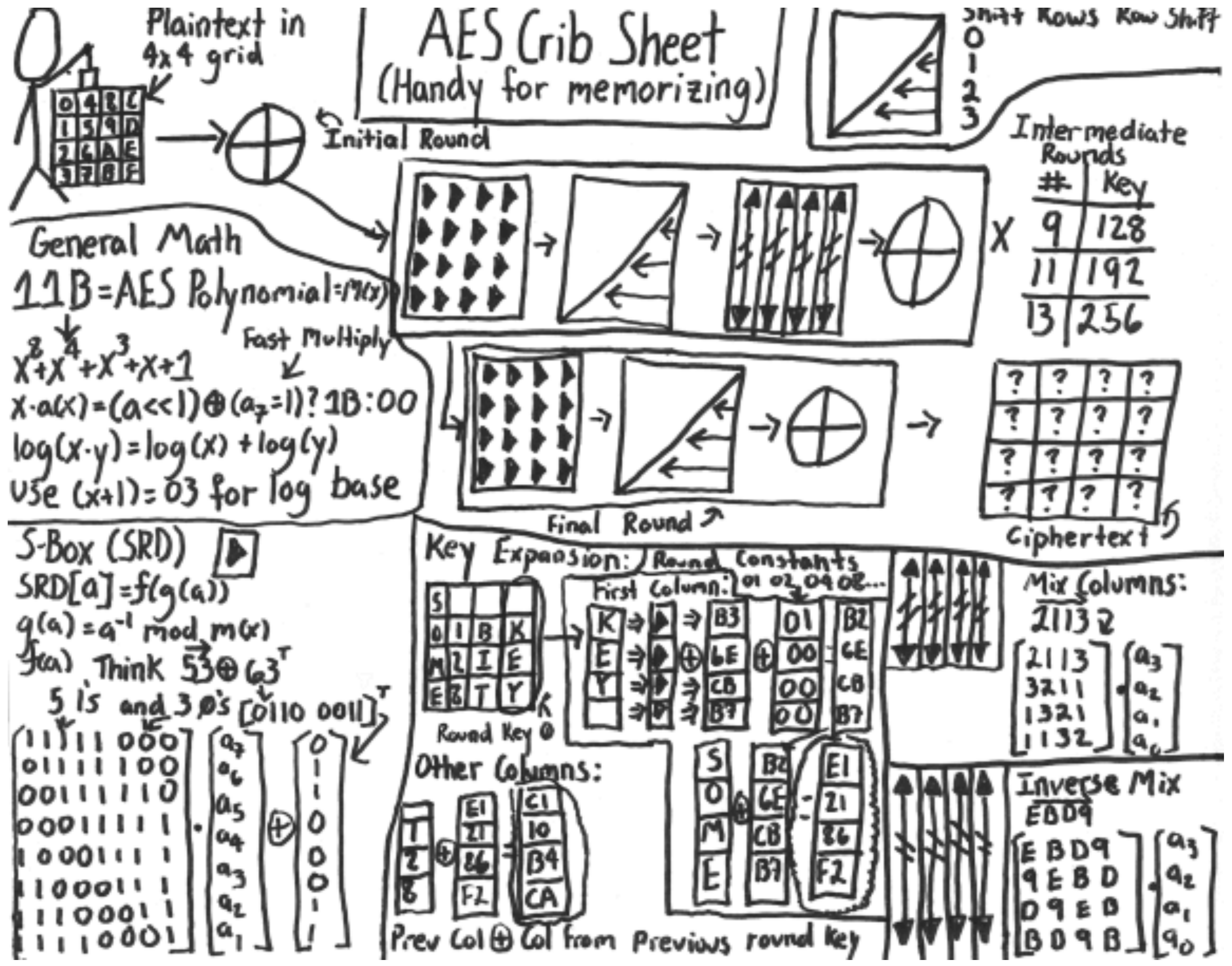
Block Ciphers



Symmetric Key: Confidentiality

Block Ciphers

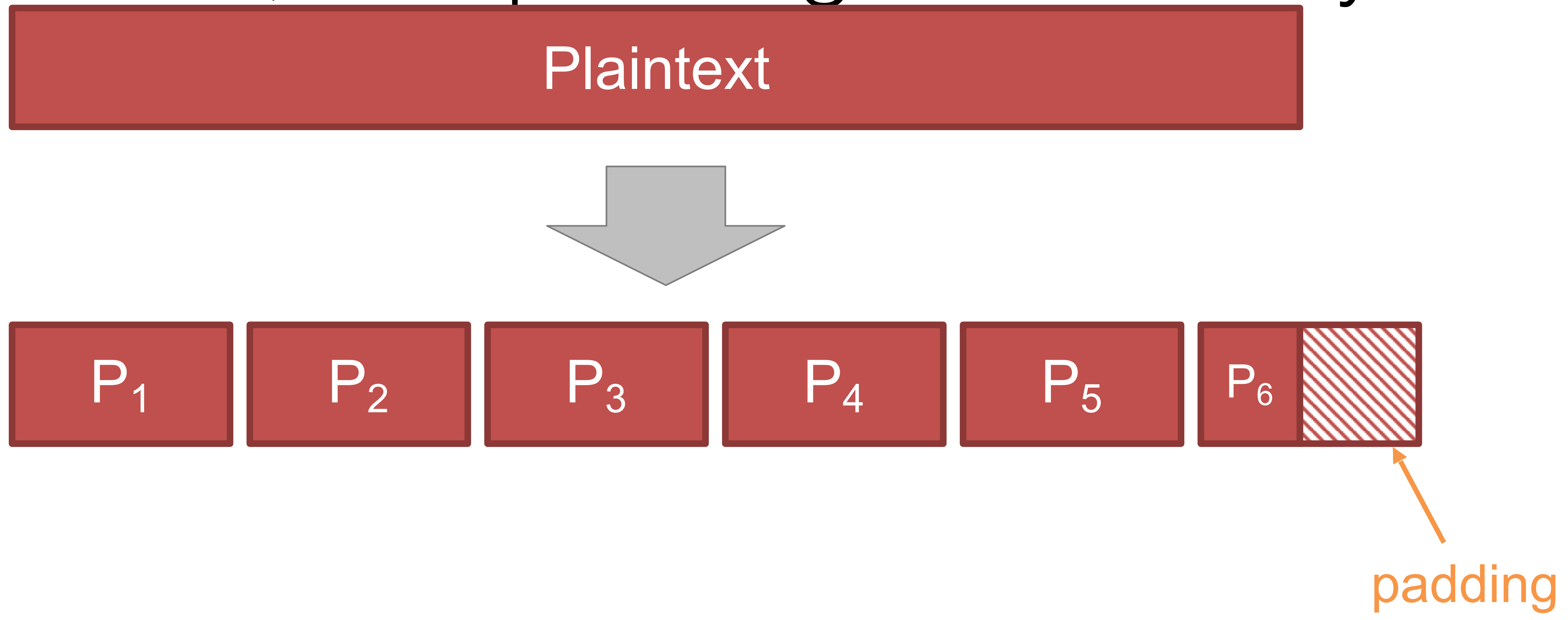




Symmetric Key: Confidentiality

Block Ciphers

- What if your data is bigger than a block?
  - Break it into blocks, add padding if necessary

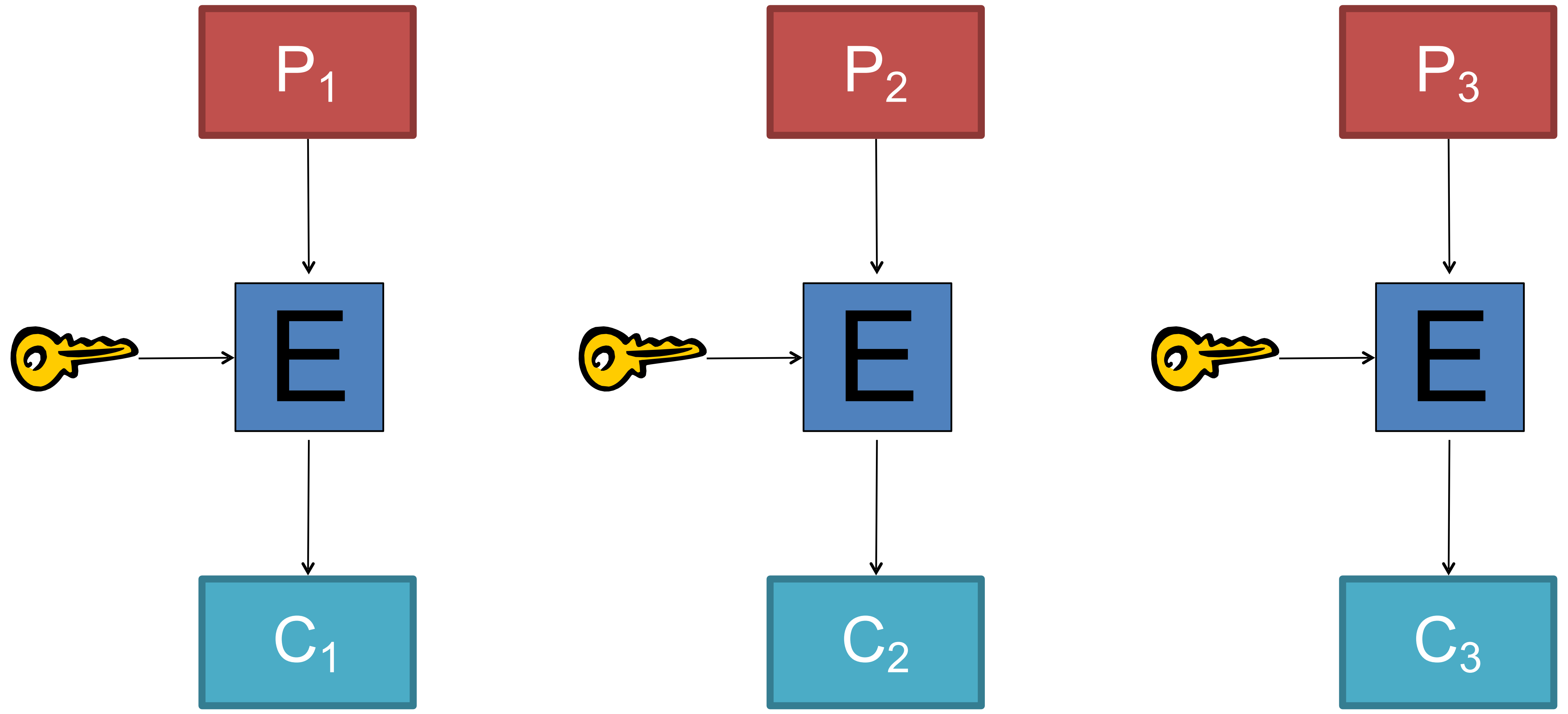


- Now what?
  - There are several **modes of operation**

Symmetric Key: Confidentiality

Block Ciphers

Electronic Code Book (**ECB Mode**)

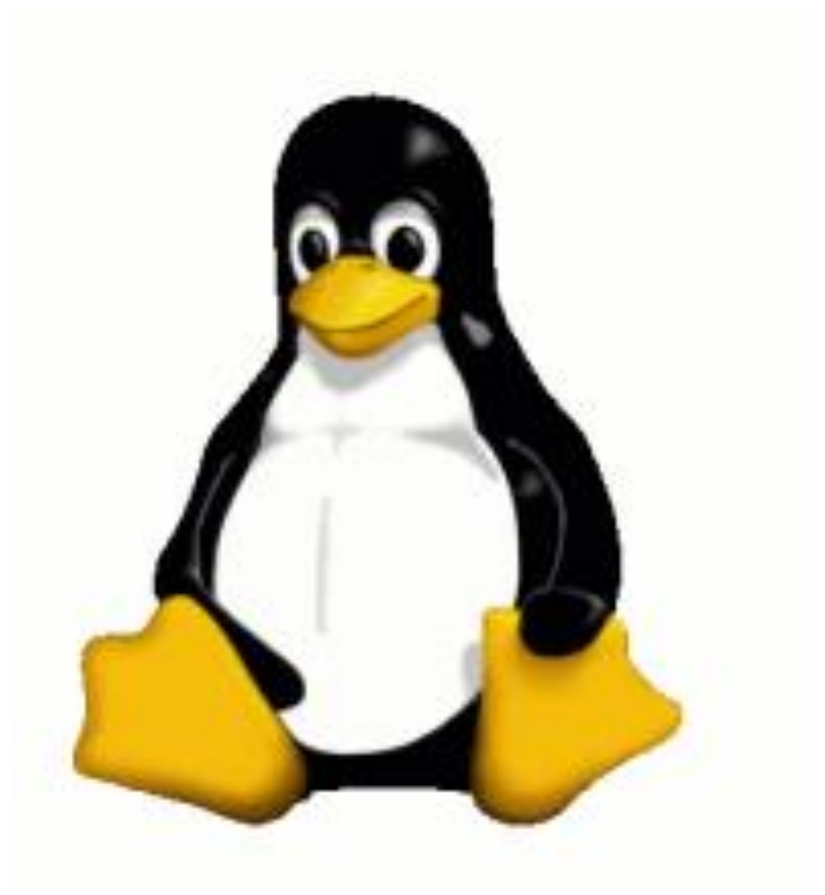


DONT LOOK AT THE NEXT SLIDE
THAT IS CHEATING
What is wrong with this algorithm?

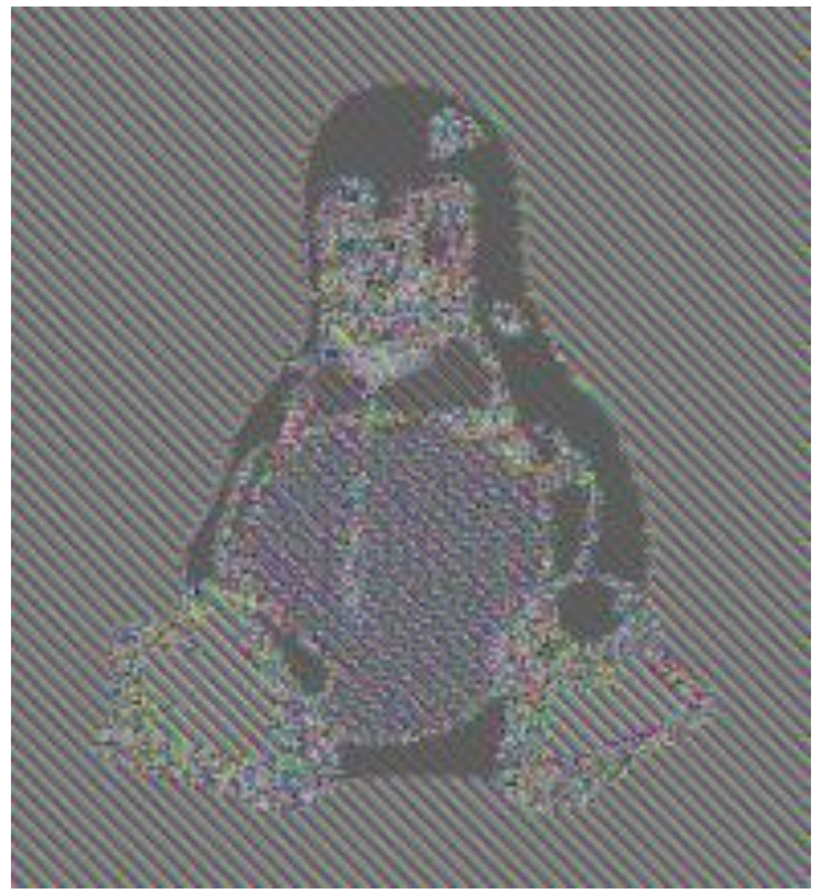
Symmetric Key: Confidentiality

SECURE? **Block Ciphers**

- Identical plaintext blocks -> identical ciphertext blocks!



Plaintext



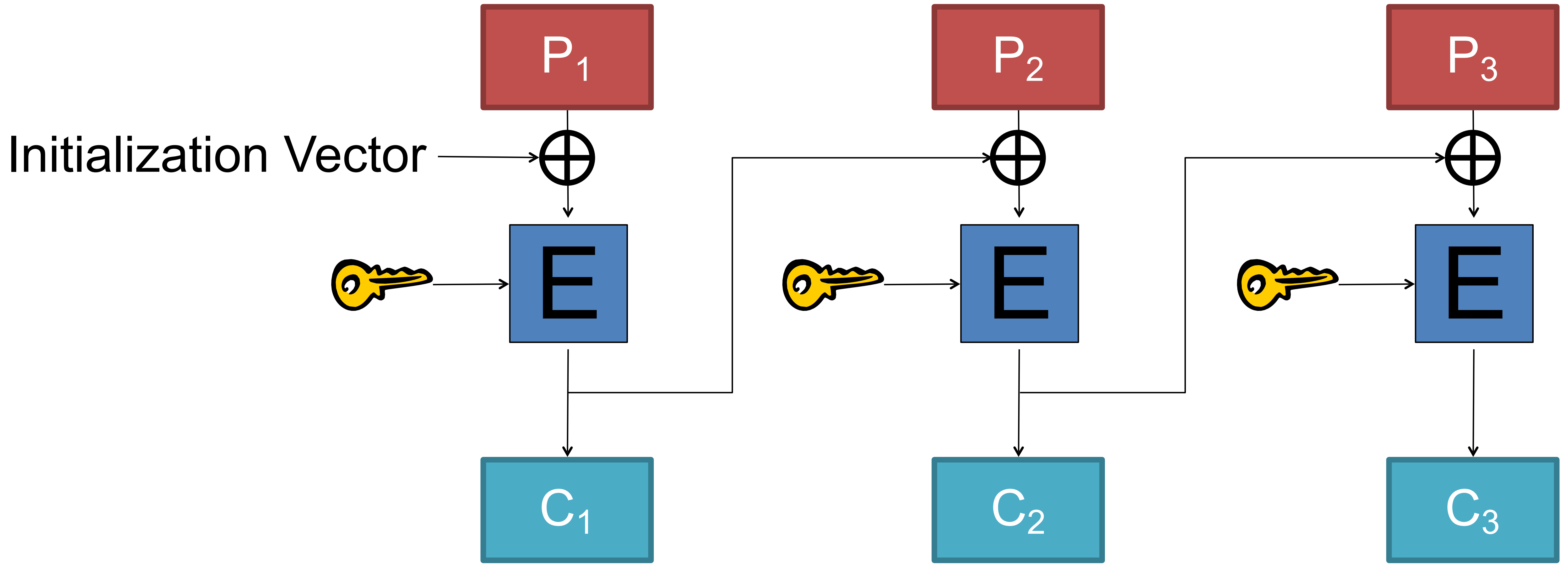
ECB

- Also: attacker can delete/re-order ciphertext blocks without affecting other blocks

Symmetric Key: Confidentiality

Block Ciphers

Cipher Block Chaining (**CBC Mode**)

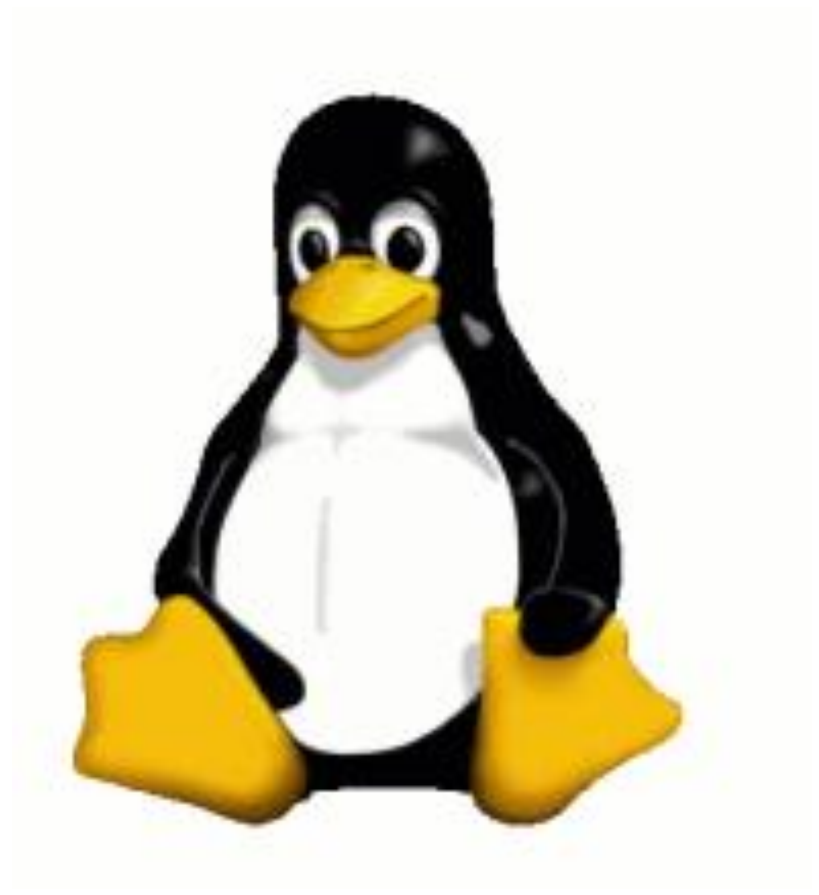


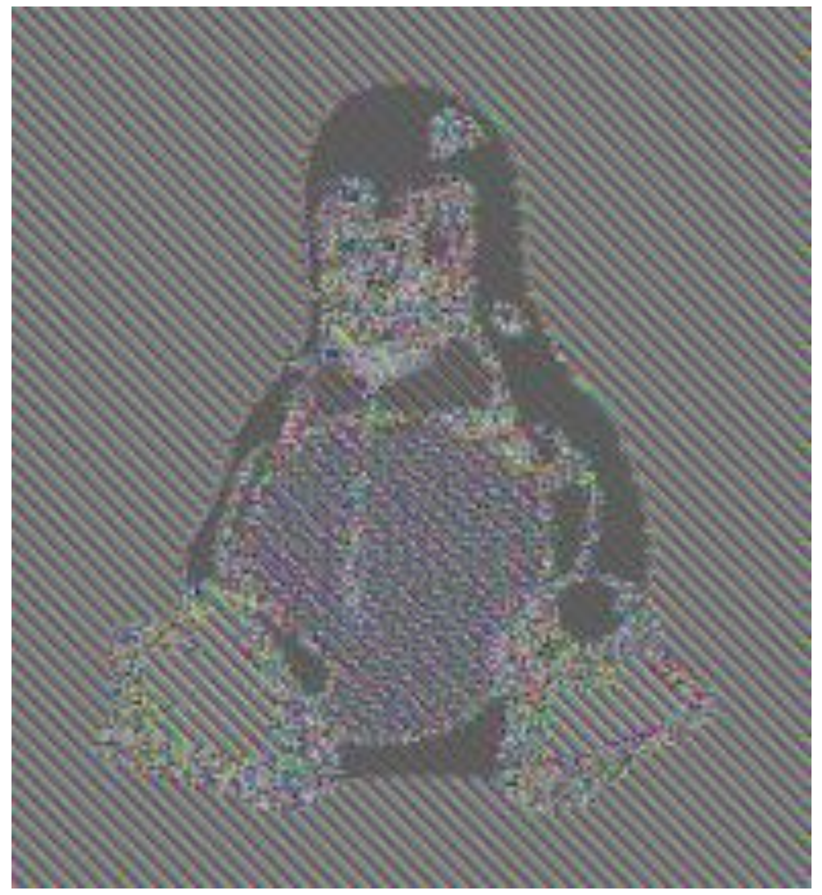
Symmetric Key: Confidentiality

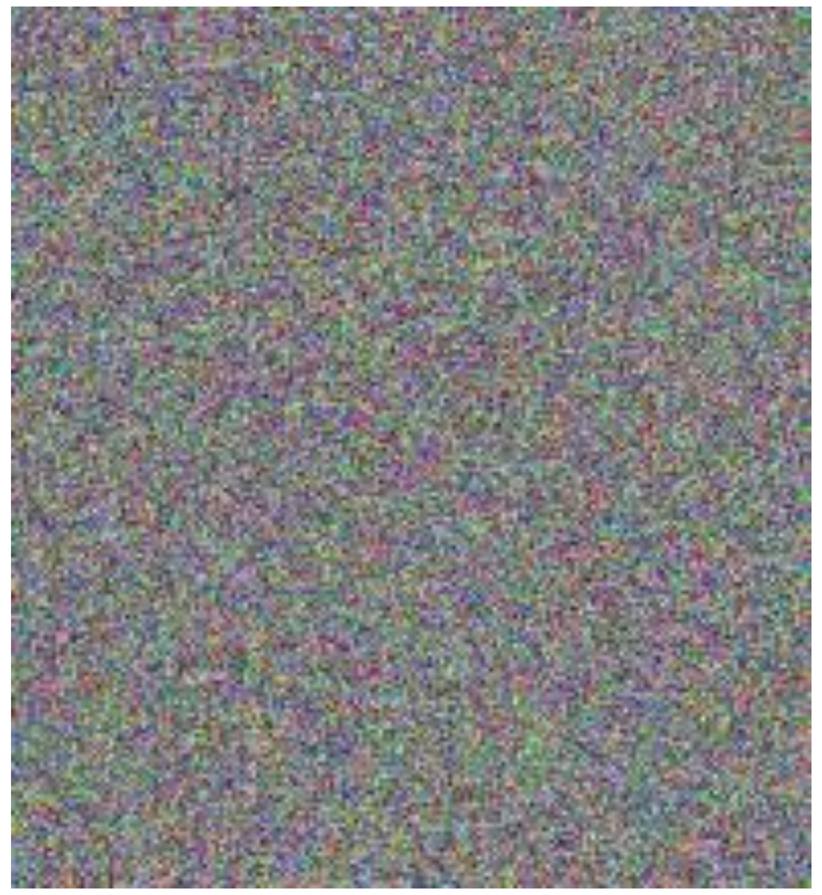
SECURE?

Block Ciphers

- Fixed ECB's big problem


Plaintext


ECB

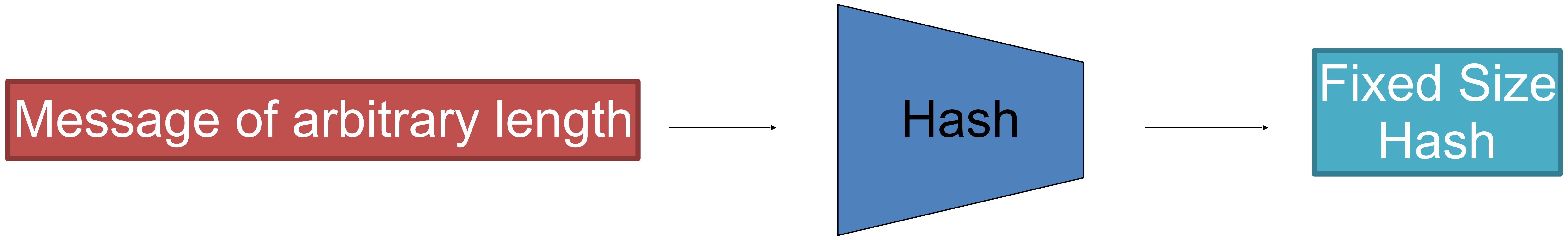

CBC

•

Cryptography Overview

	Symmetric	Asymmetric
Confidentiality	One-Time Pad Stream Ciphers Block Ciphers	
Integrity		
Authentication		

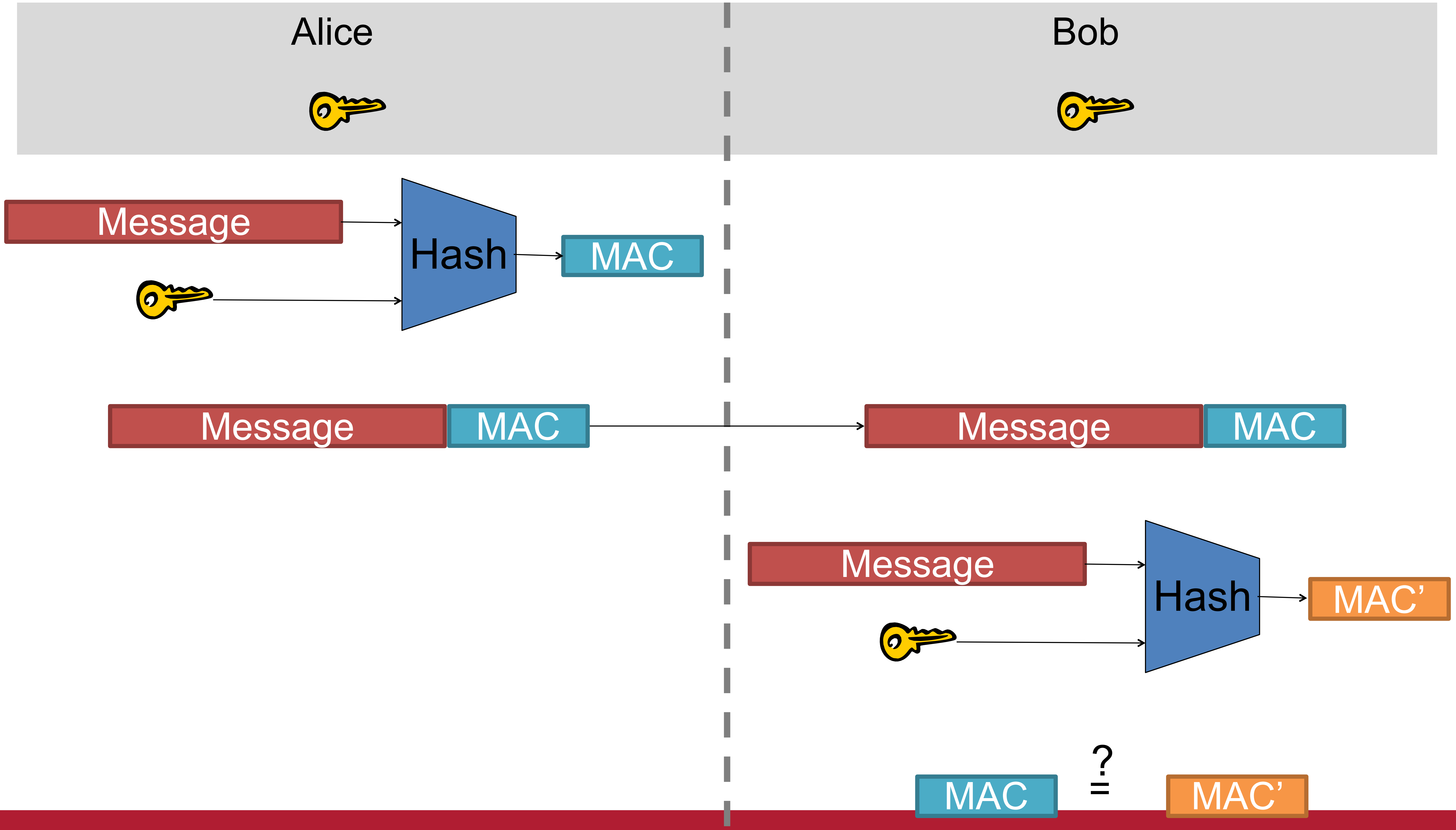
Cryptographic Hash Functions



- One-Way
  - Given $y = H(x)$, can't find x' s.t. $H(x') = y$
- Weak Collision Resistance
  - Given x, can't find $x' \neq x$ s.t. $H(x) = H(x')$
- Strong Collision Resistance
  - Can't find $x \neq x'$ s.t. $H(x) = H(x')$

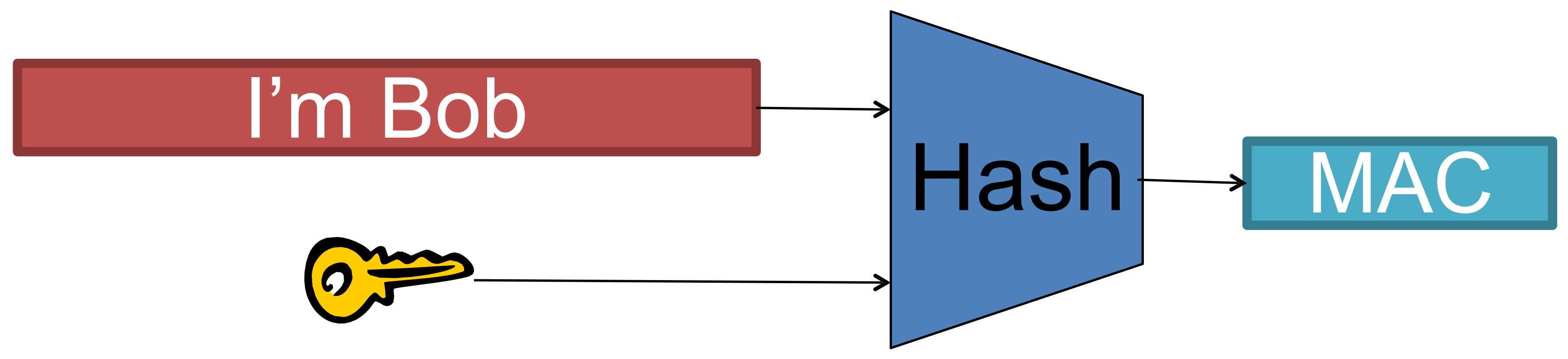
Symmetric Key: Integrity

Hash Message Authentication Code



Symmetric Key: Authentication

- You already know how to do this!
  - *(Hint: Think how we verified integrity.)*



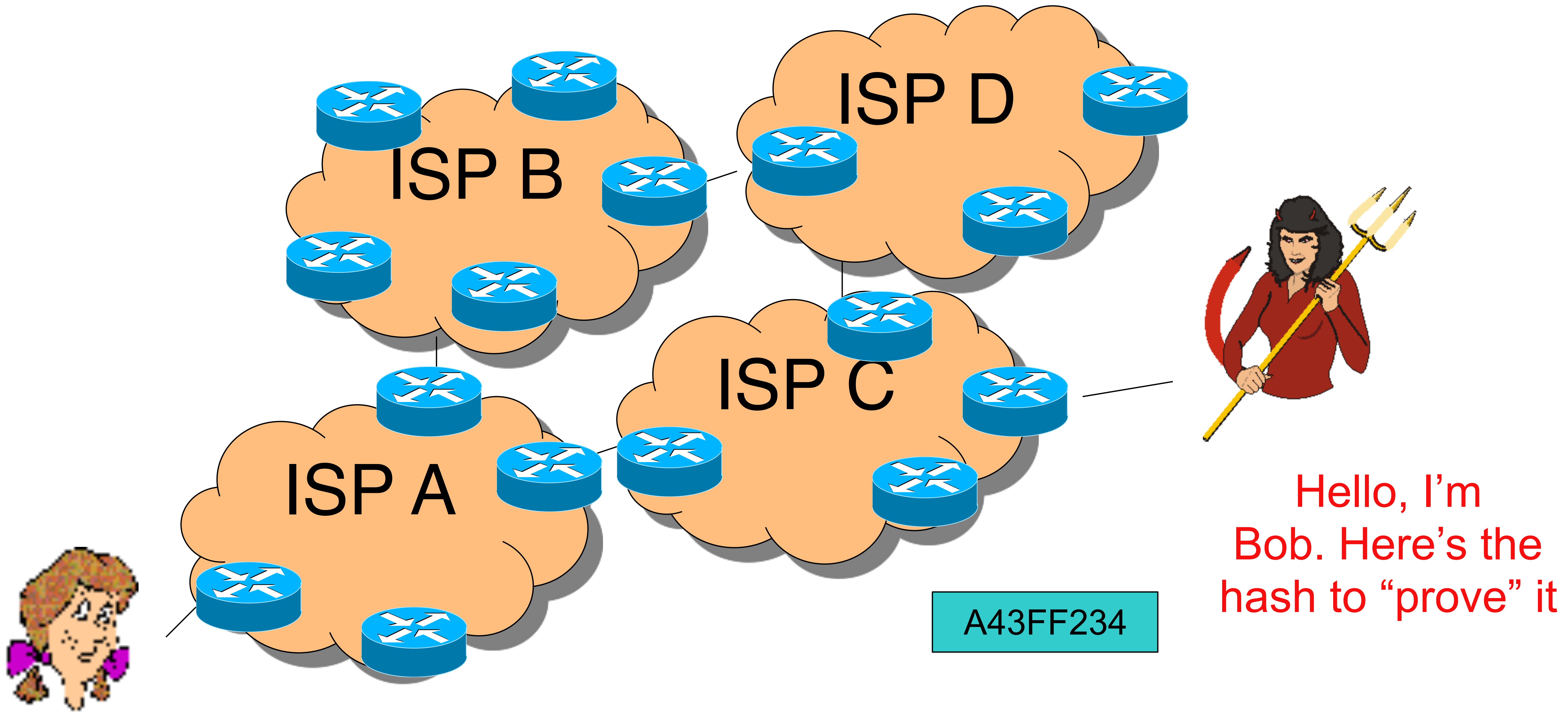
- Alice checks the MAC, knows sender is Bob

DONT LOOK AT THE NEXT SLIDE
THAT IS CHEATING
What is wrong with this algorithm?

Symmetric Key: Authentication

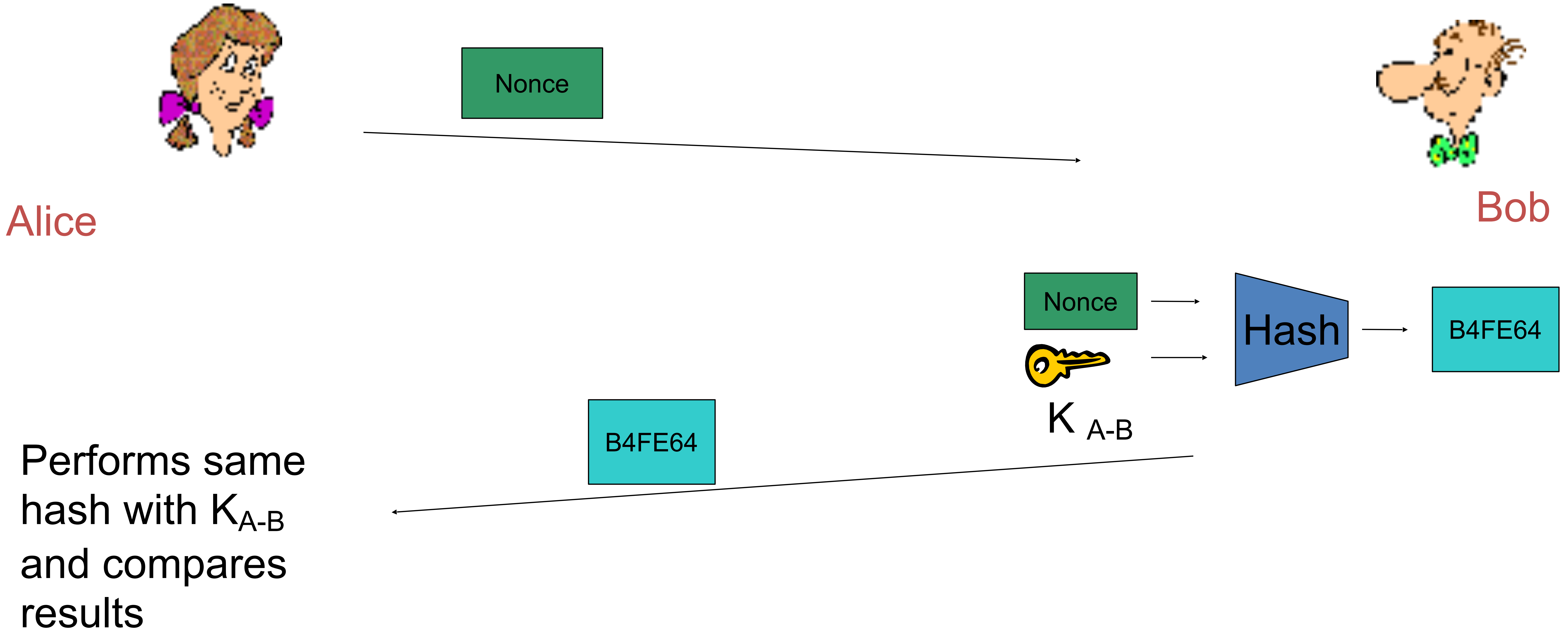
SECURE?

- What if Mallory overhears the MAC from Bob and replays it later?



Symmetric Key: Authentication

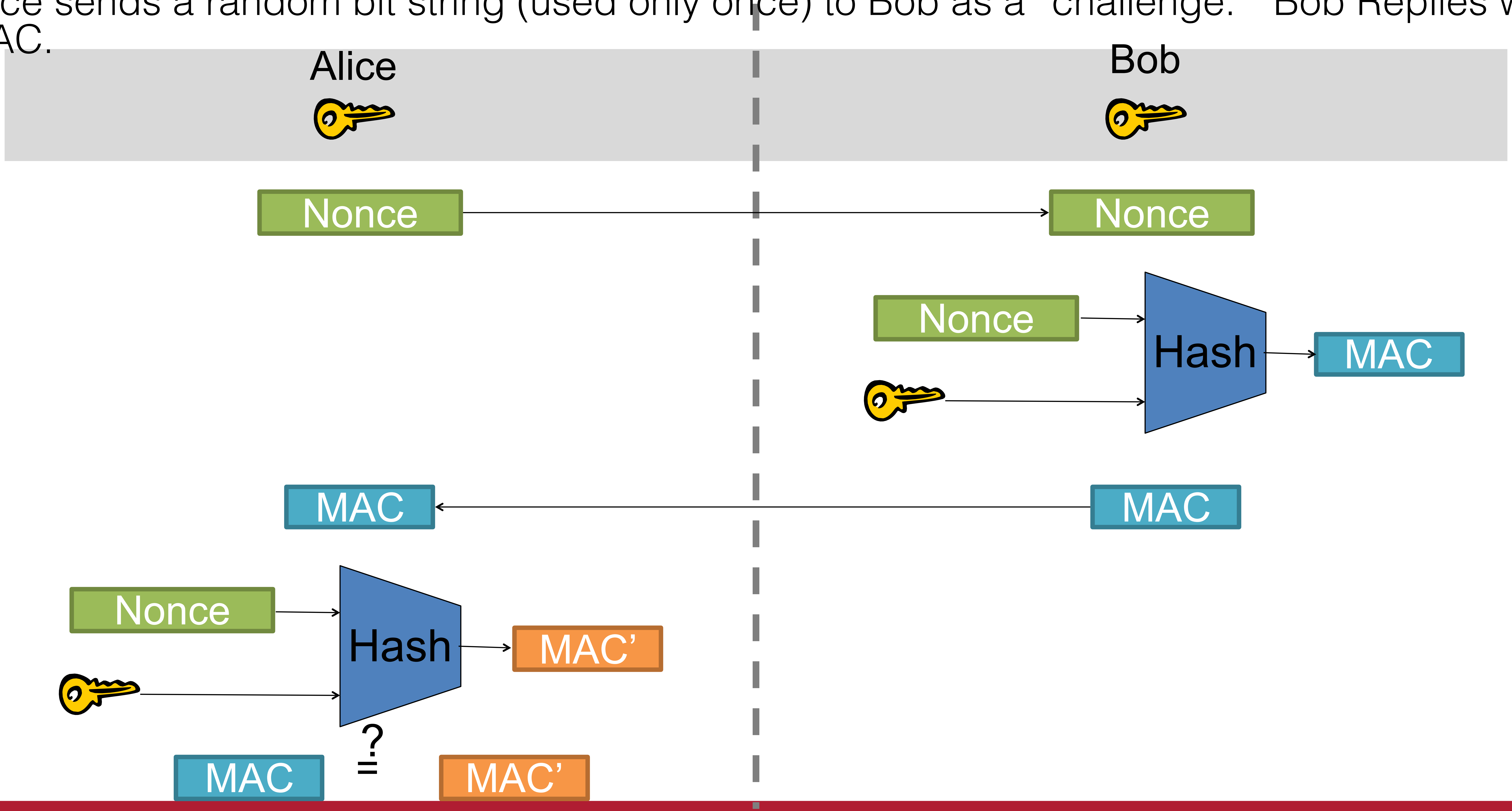
- Solution: Use a **nonce**
    - Alice sends a random bit string (used only once) to Bob as a "challenge." Bob Replies with "fresh" MAC.



Nonce
Alice
Bob
Nonce
Hash
B4FE64
K A-B
B4FE64
Performs same hash with $K_{A\text{-}B}$ and compares results

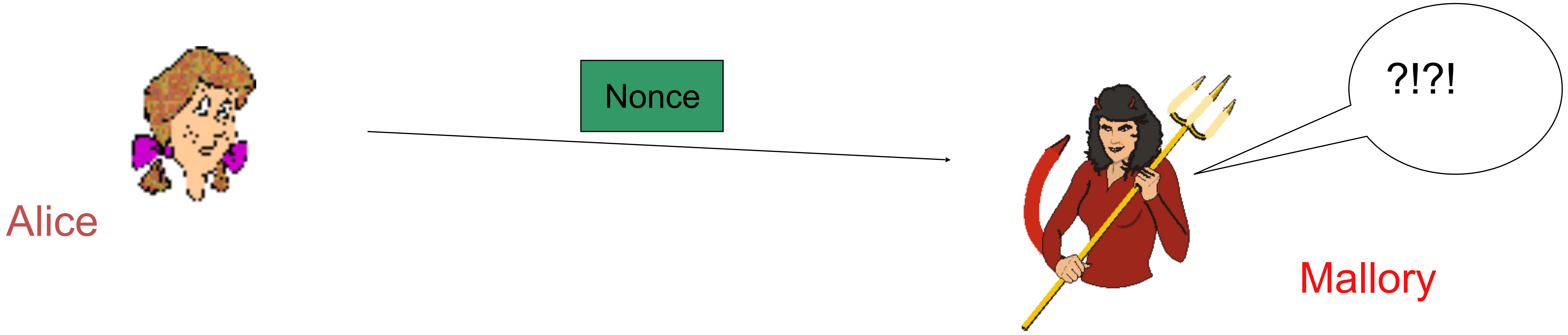
Symmetric Key: Authentication

- Solution: Use a **nonce**
  - Alice sends a random bit string (used only once) to Bob as a "challenge." Bob Replies with "fresh" MAC.



Symmetric Key: Authentication

SECURE?



If Alice sends Mallory a nonce, she cannot compute the corresponding MAC without $K_{A\text{-}B}$

Cryptography Overview

	Symmetric	Asymmetric
Confidentiality	One-Time Pad Stream Ciphers Block Ciphers	
Integrity	Message Authentication Code (e.g., HMAC, CBC-MAC)	
Authentication	MAC + Nonce	

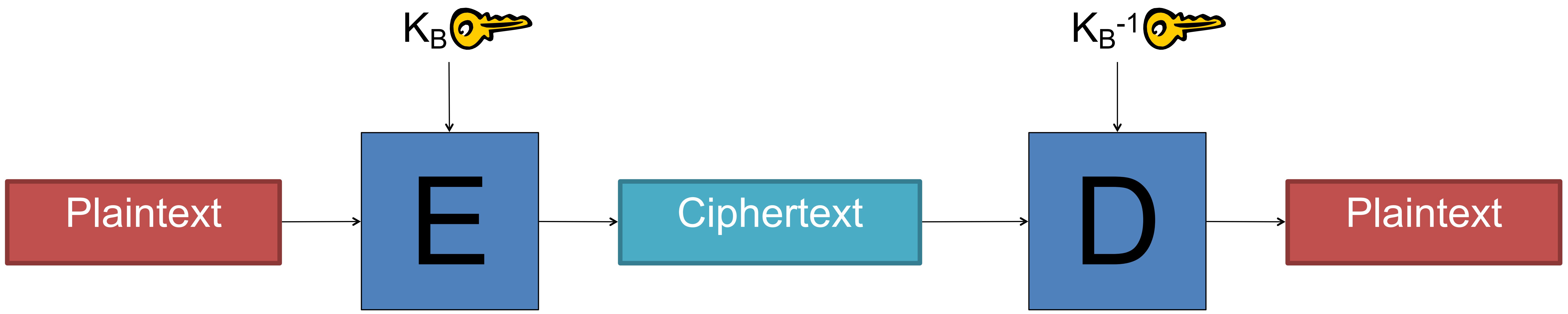
Asymmetric Key Cryptography

K_B Bob's <u>public</u> key

K_B^{-1} Bob's <u>private</u> key

- The keys are inverses, so: $K_B^{-1}\left(K_B(m)\right) = m$

- Instead of shared keys, each person has a "key pair"



Asymmetric Key Cryptography

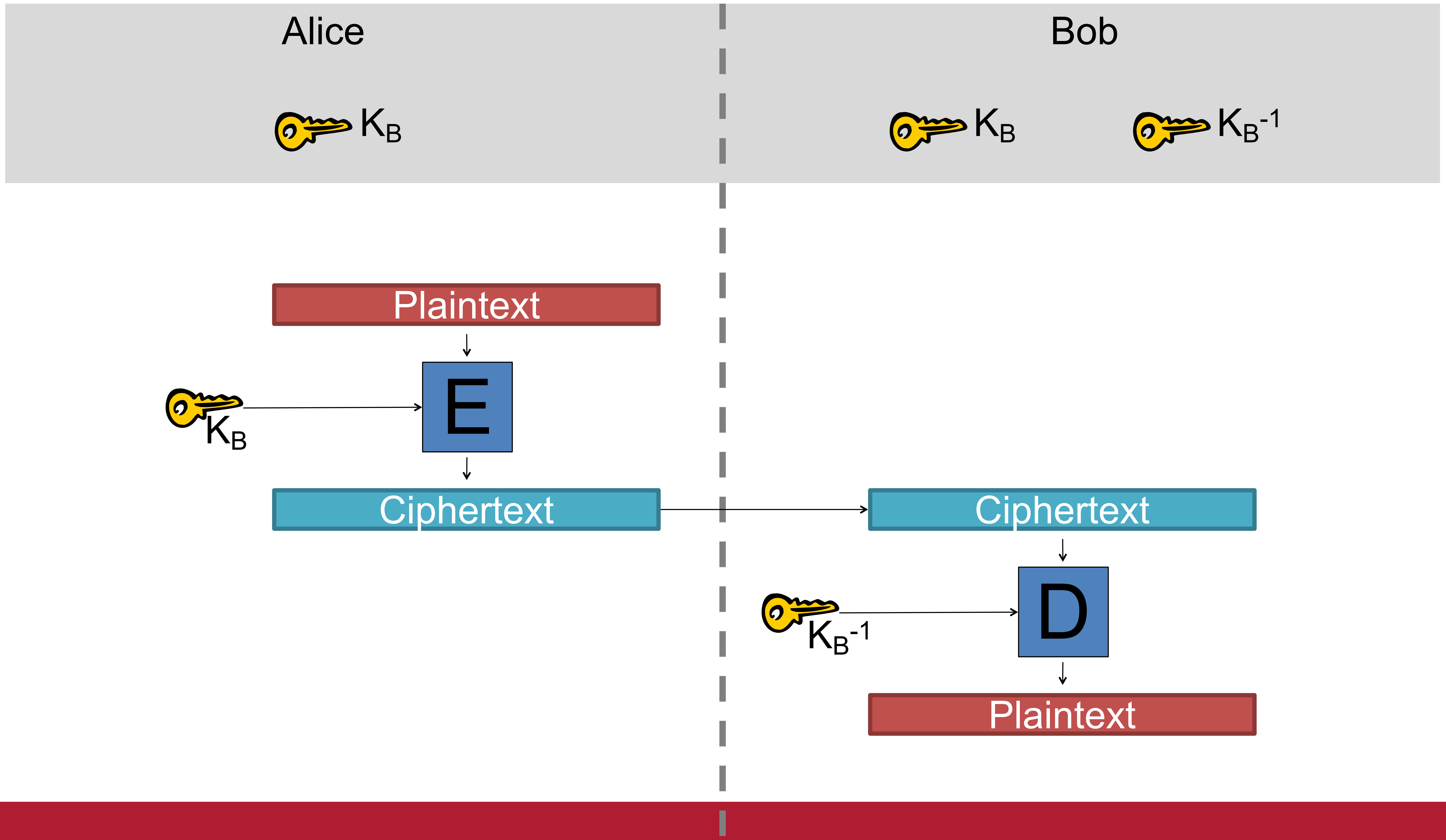
- It is believed to be computationally infeasible:
  - to derive K_B^{-1} from K_B
  - to get M from $K_B(M)$ other than using K_B^{-1}

=> K_B can safely be made public.

Note: We will not explain the computation that $K_B(m)$ entails, but rather treat these functions as black boxes with the desired properties.

Asymmetric Key: Confidentiality

Public Key Encryption



Asymmetric Key: Integrity & Authentication

- What can we conclude given
  - message M
  - value S s.t. $K_B(S) = M$

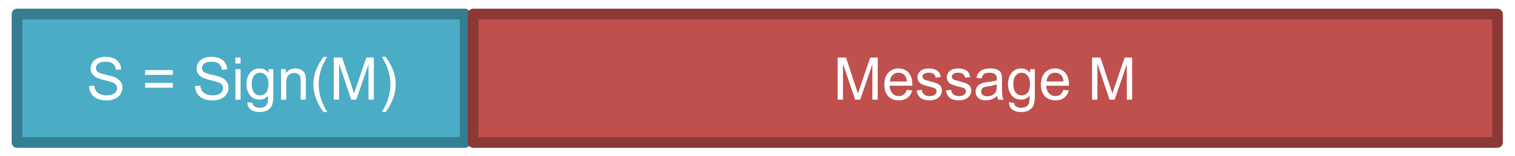
- M must be from Bob, because it must be that $S = K_B^{-1}(M)$ and only Bob has K_B^{-1}!

- This gives us two primitives:
  - $\mathrm{Sign}(M) = K_B^{-1}(M)$
  - $\mathrm{Verify}(S, M) = \mathrm{test}(\ K_B(S) == M\ )$

Asymmetric Key: Integrity & Authentication

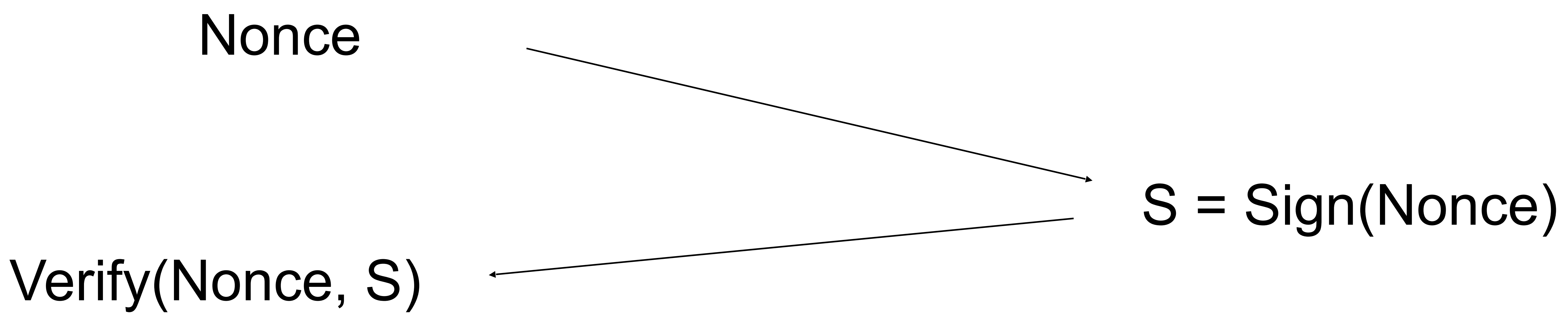
- We can use Sign() and Verify() in a similar manner as our HMAC symmetric scheme.

Integrity



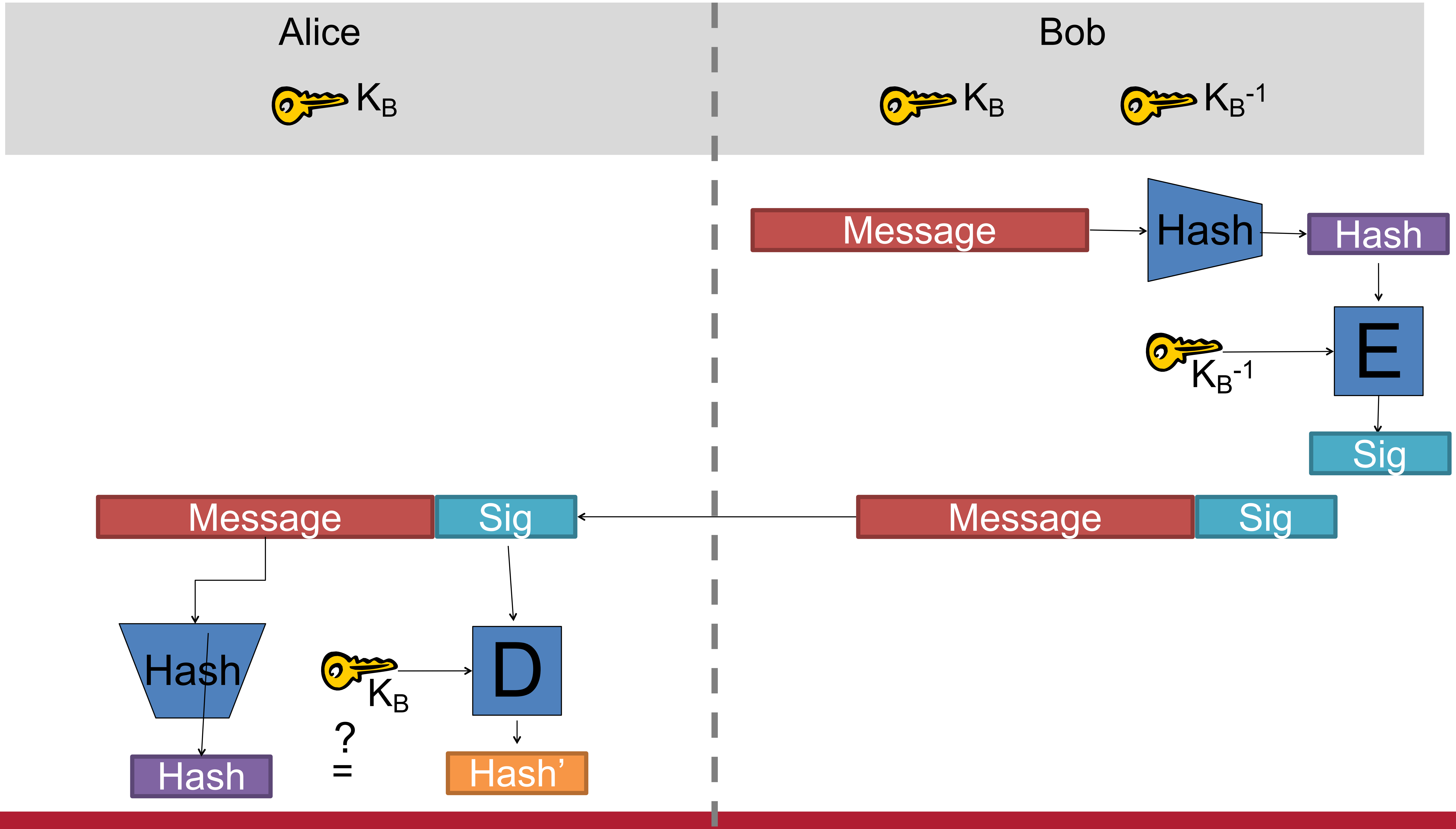
Receiver must only check Verify(M, S)

Authentication



Asymmetric Key: Integrity

Sign & Verify



Cryptography Overview

	Symmetric	**Asymmetric**
Confidentiality	One-Time Pad Stream Ciphers Block Ciphers	Encrypt w/ Public Key
Integrity	Message Authentication Code (e.g., HMAC, CBC-MAC)	Digital Signature
Authentication	MAC + Nonce	Digital Signature + Nonce

Symmetric vs. Asymmetric

Symmetric

- Shared secret
- 80 bit key for high security (in 2010)
- ~1,000,000 ops/s on 1GHz proc
- 10x speedup in HW

Asymmetric

- Public/private key pairs
- 2048 bit key for high security (in 2010)
- ~100 signs/s & ~1,000 verifies/s (RSA, 1GHz)
- Limited speedup in HW

A Note on Notation

$\{ M \}_K \rightarrow$ Encryption

K is symmetric

$\{ M \}_K \rightarrow$ Encryption

K is public

$\{ M \}_K \rightarrow$ Digital Signature

K is private

One last “little detail”…

How do I get these keys in the first place??
Remember:

- Symmetric key primitives assumed Alice and Bob had already shared a key.
- Asymmetric key primitives assumed Alice knew Bob’s public key.

This may work with friends, but when was the last time you saw Amazon.com walking down the street?

“Key Signing Party”

Key Setup

- We’ll briefly look at 2 mechanisms:
    - Diffie Hellman Key Exchange
    - Certificate Authorities

Diffie-Hellman key exchange

- An early (1976) way to create a shared secret.
- Everyone knows a prime, p, and a generator, g.
- Alice and Bob want to share a secret, but only have internet to communicate over.

THESE ARE A FEW OF MY
FAVORITE THINGS
quickmeme.com

An activity: agree on a secret word while the whole classroom can hear you.

Why is this hard?

DH key exchange

Everyone: large prime p and generator g

Create secret: a

Alice

Send Bob: $g^a \bmod p$ →

Create secret: b

Bob

← Send Alice: $g^b \bmod p$

Compute: $(g^b \bmod p)^a$

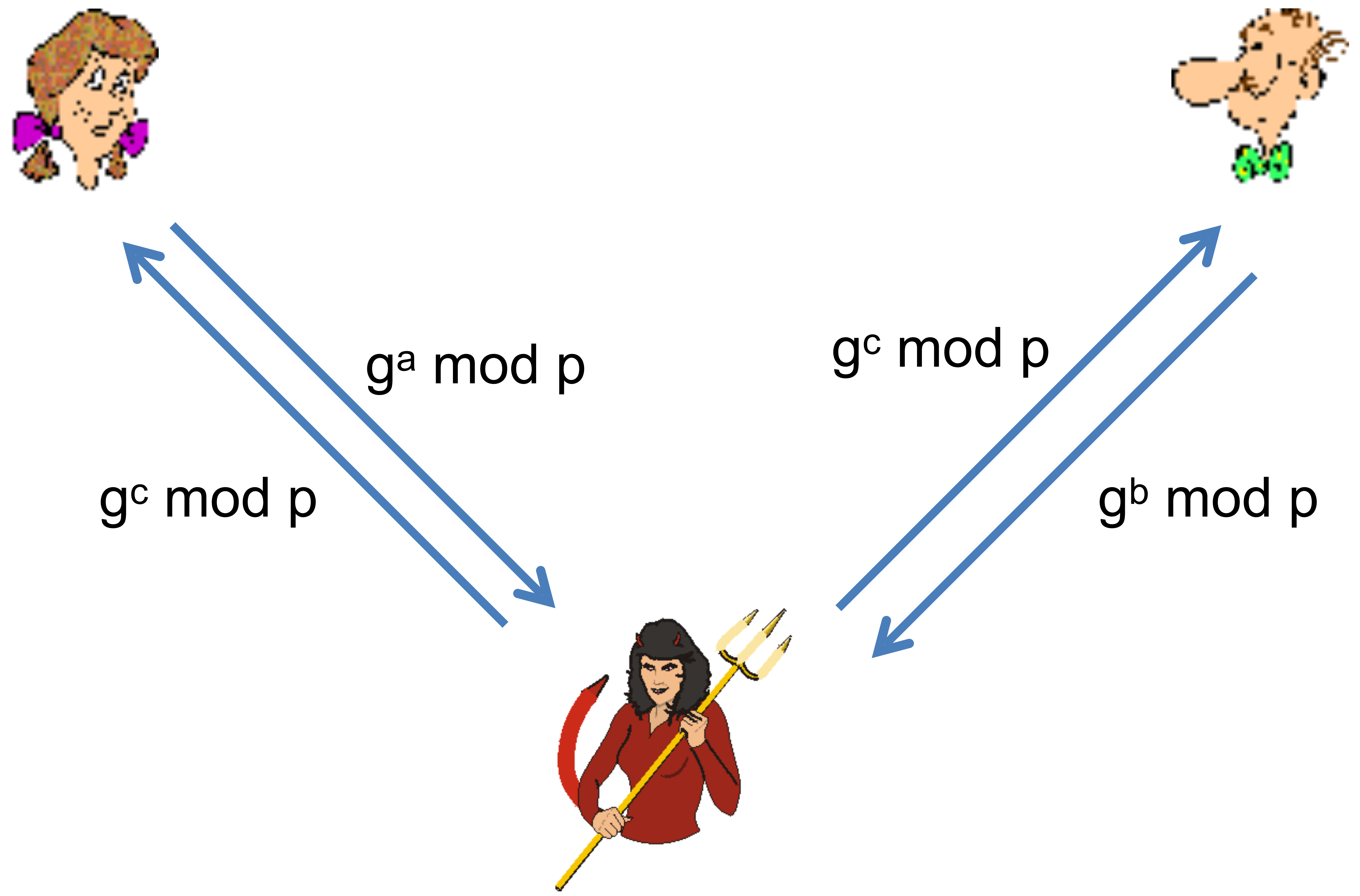
Compute: $(g^a \bmod p)^b$

Voila: They both know g^{ab} which is secret!

Math says: No attacker can compute g^{ab} mod p just by listening to their communication! (It’s computationally intractable)

Security mindset: are we good to go?

DH key exchange & Man-In-The-Middle

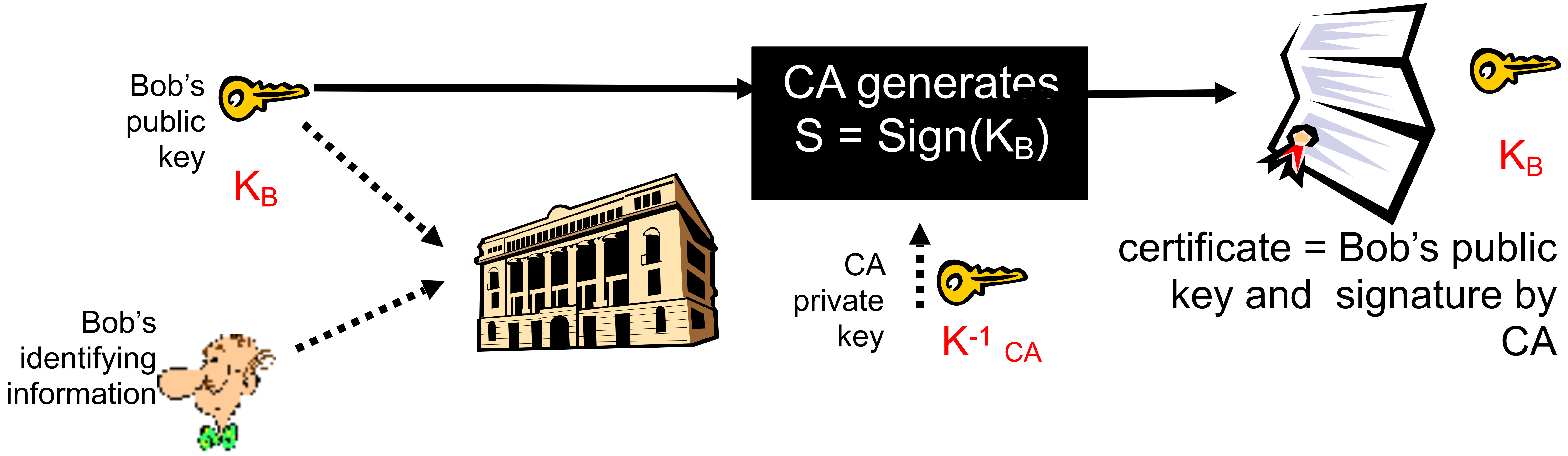


Threat Model

- Always important to be clear about what you think your attacker is capable of!
- If you think your attacker is capable of *modifying* traffic, can't use DH!
- But if attacker is just an eavesdropper — you're good to go!

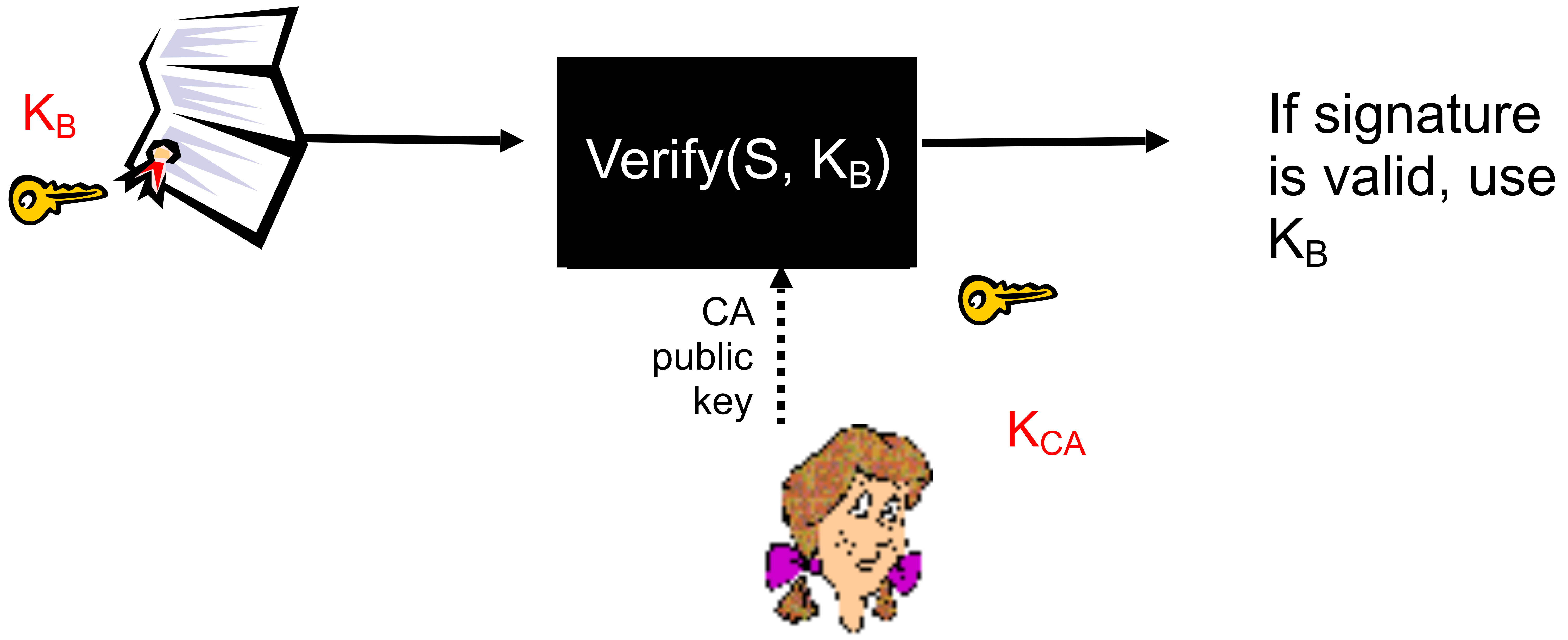
Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- An entity E registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - Certificate contains E's public key AND the CA's signature of E's public key.

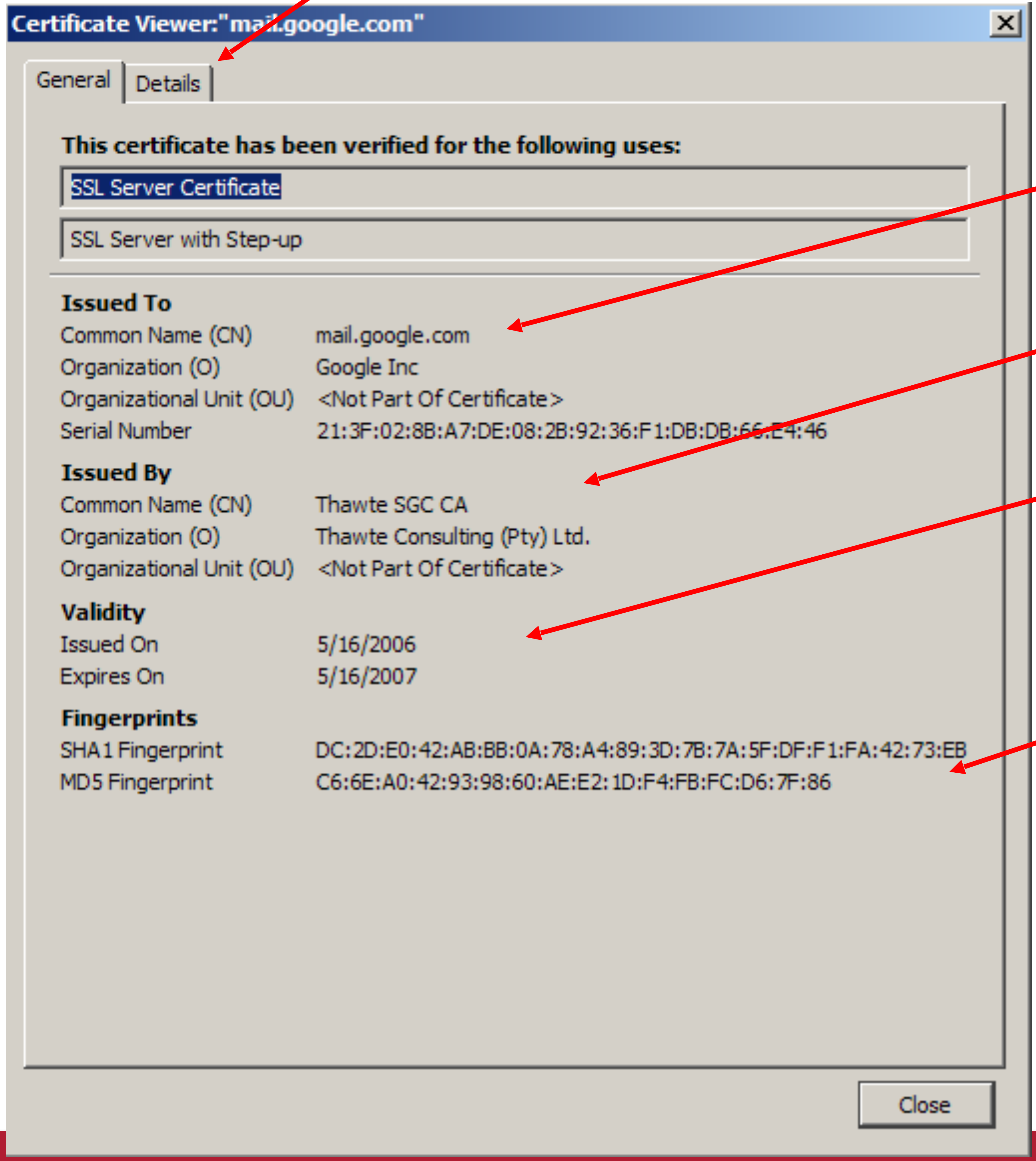

Bob's public key
K_B
Bob's identifying information
CA generates $S = \text{Sign}(K_B)$
CA private key
K^{-1}_{CA}
K_B
certificate = Bob's public key and signature by CA

Certification Authorities

- When Alice wants Bob's public key:
  - Gets Bob's certificate (Bob or elsewhere).
  - Use CA's public key to verify the signature within Bob's certificate, then accepts public key



Certificate Contents



This certificate has been verified for the following uses:

SSL Server Certificate

SSL Server with Step-up

Issued To

Common Name (CN)	mail.google.com
Organization (O)	Google Inc
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	21:3F:02:8B:A7:DE:08:2B:92:36:F1:DB:DB:66:E4:46

Issued By

Common Name (CN)	Thawte SGC CA
Organization (O)	Thawte Consulting (Pty) Ltd.
Organizational Unit (OU)	<Not Part Of Certificate>

Validity

Issued On	5/16/2006
Expires On	5/16/2007

Fingerprints

SHA1 Fingerprint	DC:2D:E0:42:AB:BB:0A:78:A4:89:3D:7B:7A:5F:DF:F1:FA:42:73:EB
MD5 Fingerprint	C6:6E:A0:42:93:98:60:AE:E2:1D:F4:FB:FC:D6:7F:86

- Cert owner
- Cert issuer
- Valid dates
- Fingerprint of signature

Which Authority Should You Trust?

- Today: many authorities
- What about a shared Public Key Infrastructure (PKI)?
  - A system in which “roots of trust” authoritatively bind public keys to real-world identities
  - So far it has not been very successful

Let’s put it all together!

Transport Layer Security (TLS)
aka Secure Socket Layer (SSL)

Uses **certificate authority** to provide public key

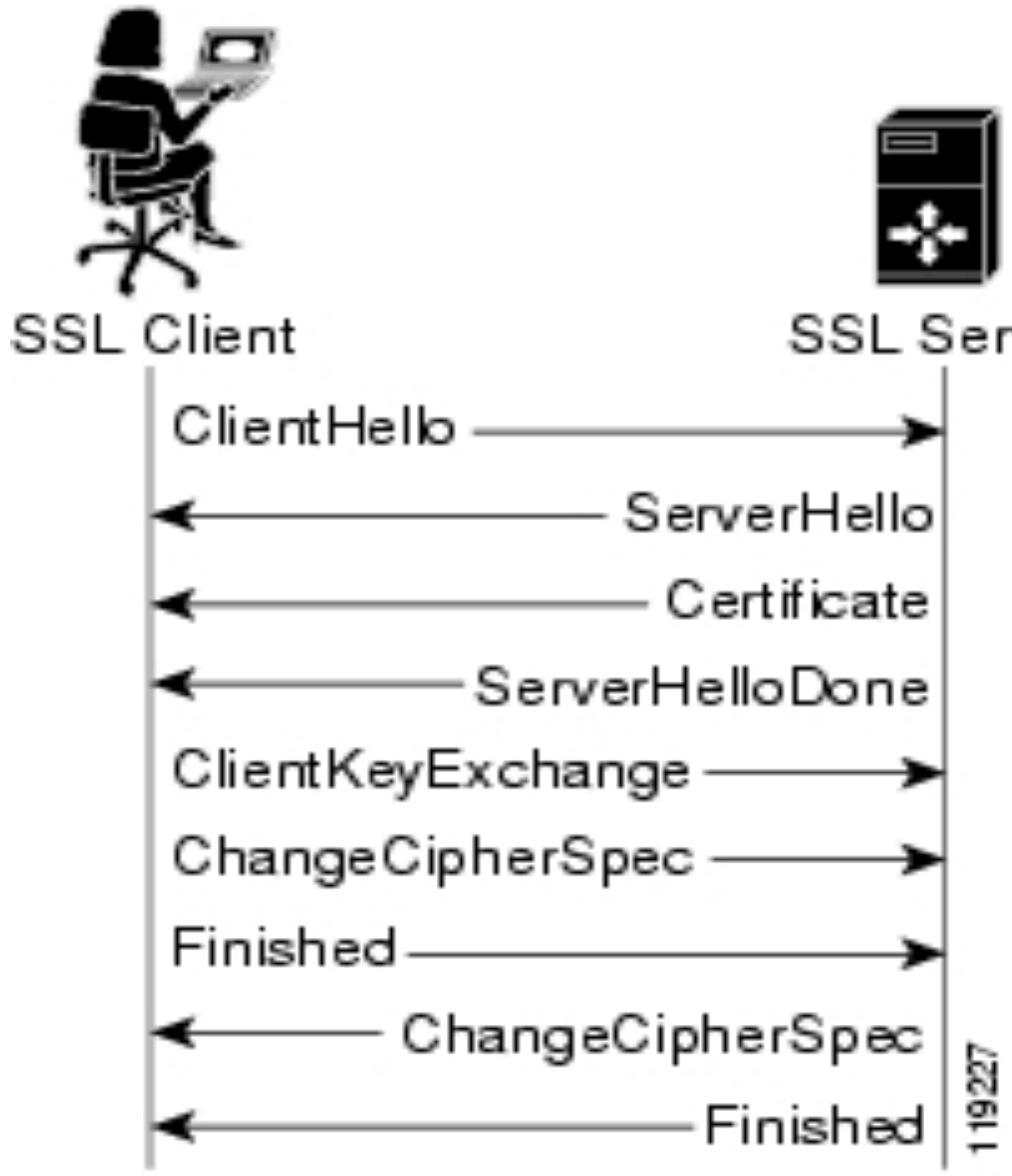
↓

Uses **asymmetric crypto** to establish symmetric key

↓

Uses **symmetric crypto** for data encryption

Setup Channel with TLS “Handshake”



Handshake Steps:

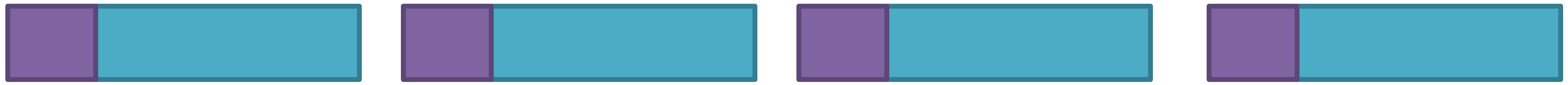
1) Client and server negotiate exact cryptographic protocols
2) Client validates public key certificate with CA public key.
3) Client encrypts secret random value with server’s key, and sends it as a challenge.
4) Server decrypts, proving it has the corresponding private key.
5) This value is used to derive symmetric session keys for encryption & MACs.

How TLS Handles Data

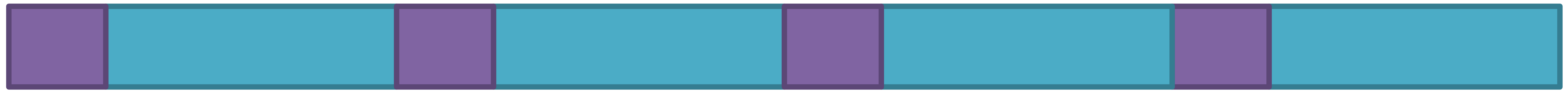
1) Data arrives as a stream from the application via the TLS Socket

2) The data is segmented by TLS into chunks

3) A session key is used to encrypt and MAC each chunk to form a TLS “record”, which includes a short header and data that is encrypted, as well as a MAC.



4) Records form a byte stream that is fed to a TCP socket for transmission.



Summary – Part I

- Internet design and growth => security challenges
- Symmetric (pre-shared key, fast) and asymmetric (key pairs, slow) primitives provide:
  - Confidentiality
  - Integrity
  - Authentication
- “Hybrid Encryption” leverages strengths of both.
- Great complexity exists in securely acquiring keys.
- Crypto is hard to get right, so use tools from others, don’t design your own (e.g. TLS).

Resources

- Textbook: 8.1 – 8.3
- Wikipedia for overview of Symmetric/Asymmetric primitives and Hash functions.
- OpenSSL (www.openssl.org): top-rate open source code for SSL and primitive functions.
- "Handbook of Applied Cryptography" available free online: www.cacr.math.uwaterloo.ca/hac/