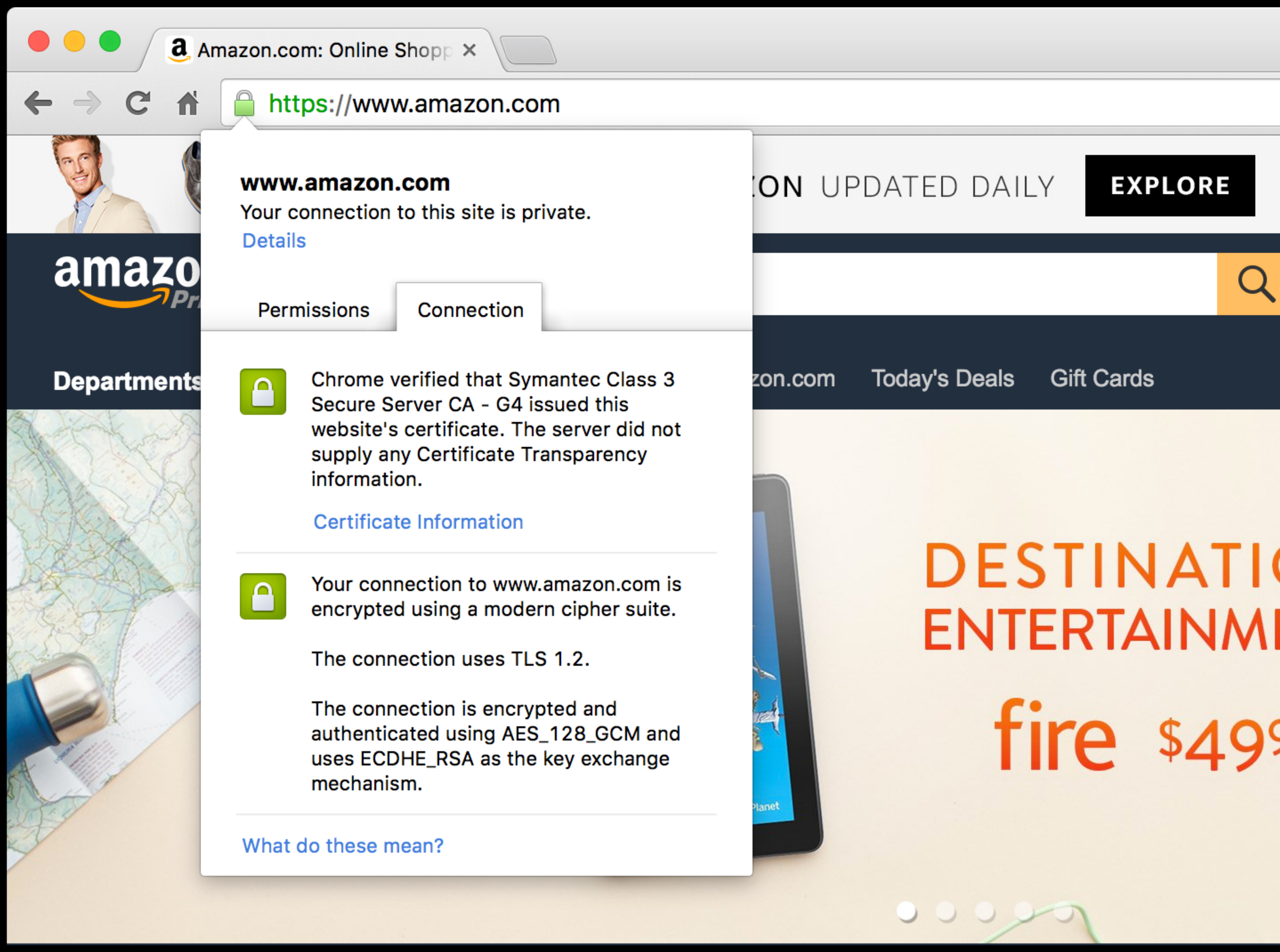
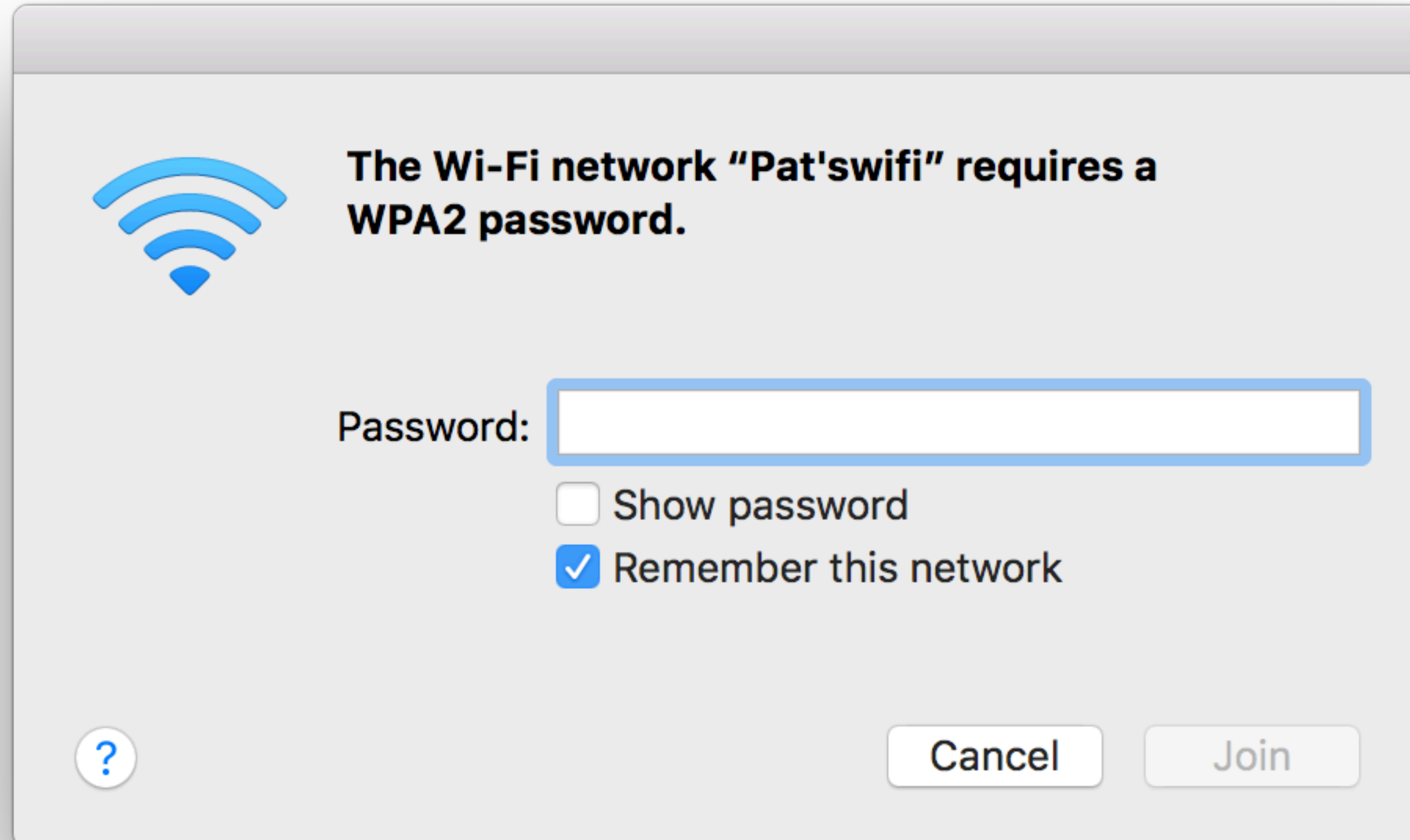
Cryptography Part 1

CMSC 23200/33250, Winter 2023, Lecture 9

David Cash & Blase Ur

University of Chicago


Amazon.com: Online Shopp
https://www.amazon.com
www.amazon.com
Your connection to this site is private.
Details
Permissions
Connection
Chrome verified that Symantec Class 3 Secure Server CA - G4 issued this website's certificate. The server did not supply any Certificate Transparency information.
Certificate Information
Your connection to www.amazon.com is encrypted using a modern cipher suite.
The connection uses TLS 1.2.
The connection is encrypted and authenticated using AES_128_GCM and uses ECDHE_RSA as the key exchange mechanism.
What do these mean?
UPDATED DAILY
EXPLORE
Departments
Today's Deals
Gift Cards
DESTINATI
ENTERTAINM
fire


The Wi-Fi network “Pat'swifi” requires a WPA2 password.
Password:
Show password
Remember this network
?
Cancel
Join

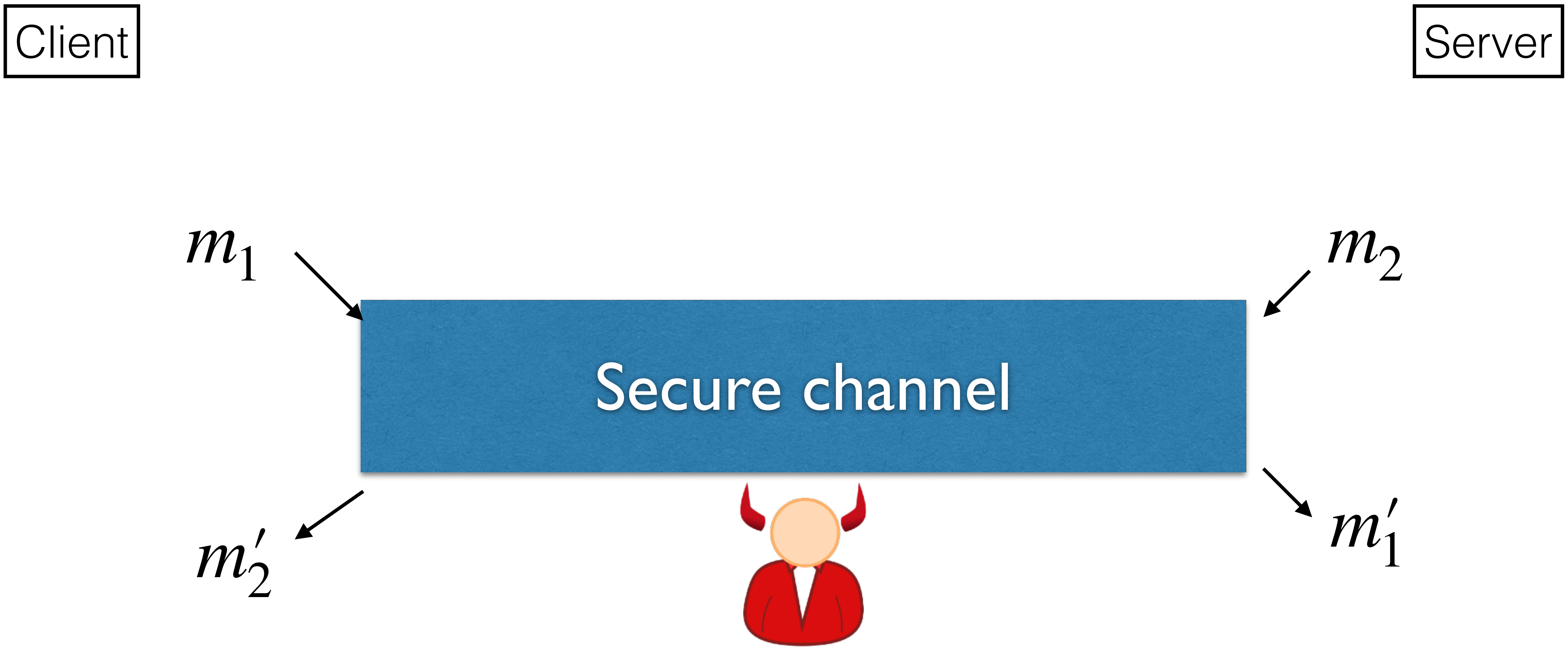

nothing
Today 11:11
Can you please come over asap to help me move the couch?
I need to be out of here by 3pm
I guess you forgot your phone at home or something
Delivered
Send
iMessage

What is Cryptography?

Cryptography involves algorithms with security goals.

Cryptography involves using math to stop adversaries.

Common Security Goal: Secure Channel



Confidentiality: Adversary does not learn anything about messages m_1, m_2

Authenticity: $m_1' = m_1$ and $m_2' = m_2$

Crypto in CS23200/33250

- A brief overview of major concepts and tools
- Cover (some of) big “gotchas” in crypto deployments
- Cover background for networking and authentication later

Not going to cover math, proofs, or many details.
Consider taking CS284 (Cryptography)!

Four settings for cryptography

Pre-shared key? \ Security Goal	Confidentiality	Authenticity/Integrity
Yes (“Symmetric”)	Symmetric Encryption (aka Secret-key Encryption)	Message Authentication Code (MAC)
No (“Asymmetric”)	Public-Key Encryption	Digital Signatures

Rest of this lecture

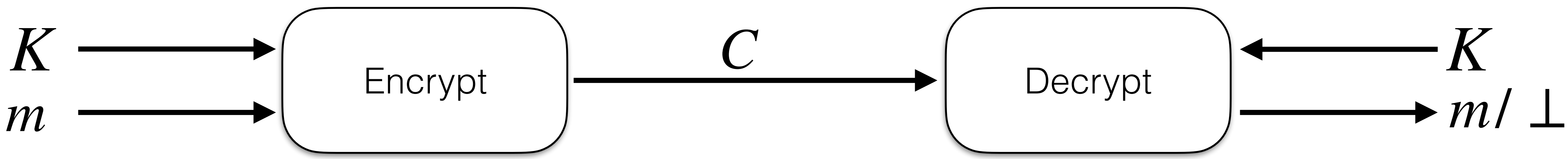
- Symmetric Encryption Basics
- Stream Ciphers
- Message Authentication Codes

Rest of this lecture

- **Symmetric Encryption Basics**
- Stream Ciphers
- Message Authentication Codes

Ciphers (a.k.a. Symmetric Encryption)

A <u>cipher</u> is a pair of algorithms Encrypt, Decrypt:



Require that decryption recovers the same message.

Historical Cipher: ROT13 (“Caesar cipher”)

Encrypt(K,m): shift each letter of plaintext forward by K positions in alphabet (wrap from Z to A).

Plaintext: **DEFGH**
Key (shift): 3
Ciphertext: FGHKL

Plaintext: **ATTACKATDAWN**
Key (shift): 13
Ciphertext: NGGNPXNGQNJA

Historical Cipher: Substitution Cipher

Encrypt(K,m): Parse key K as a permutation π on {A,… Z}. Apply π to each character of m.

P: **ATTACKATDAWN**

K: π →

C: **ZKKZAMZKYZGT**

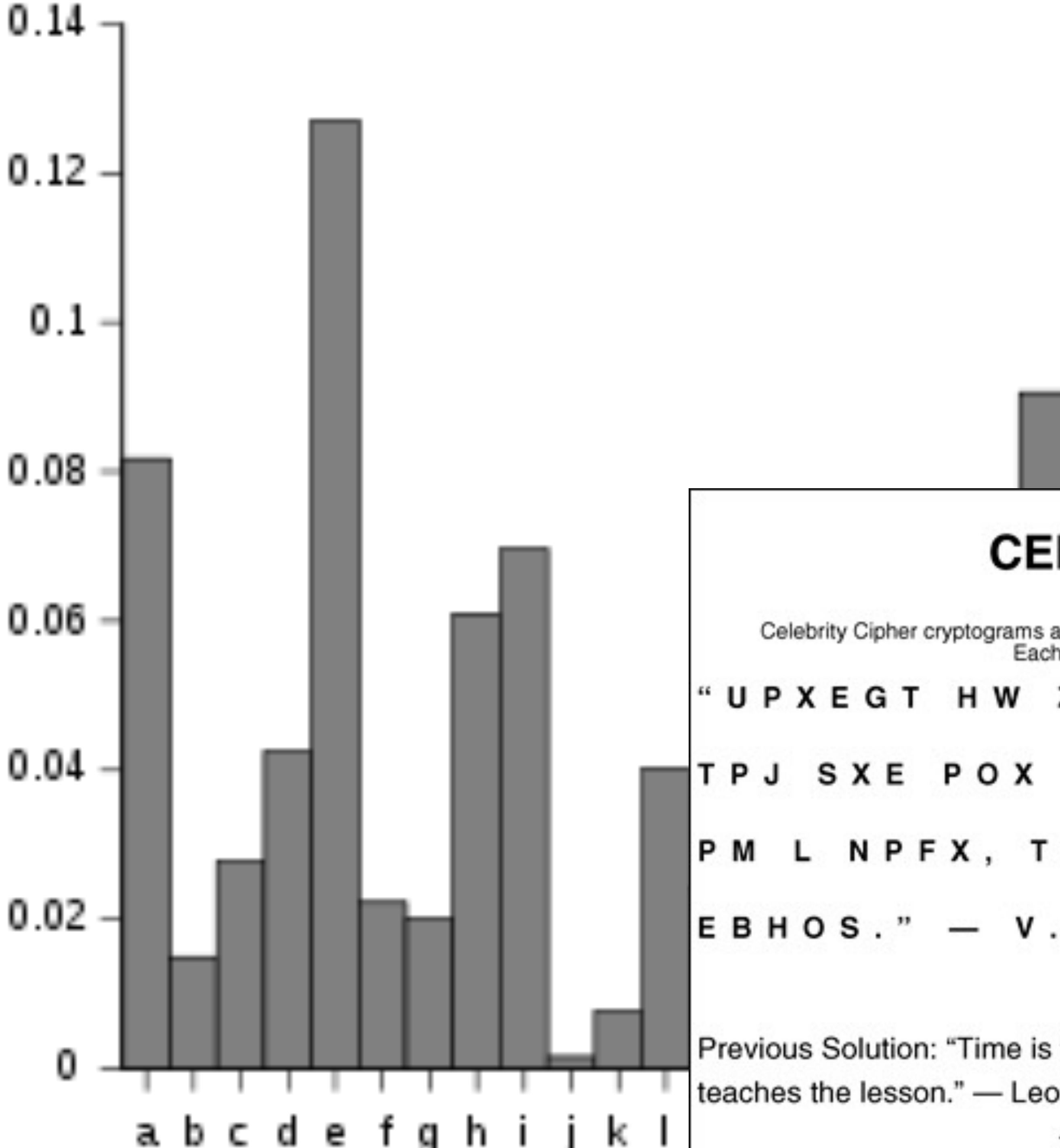
x	π(x)
A	Z
B	U
C	A
D	Y
E	R
F	E
G	X
H	B
I	D
J	C
K	M
L	Q
M	H
N	T
O	I
P	S
Q	V
R	N
S	P
T	K
U	O
V	F
W	G
X	W
Y	L
Z	J

How many keys?

$26! \approx 2^{88}$

9 million years to try all keys at rate of 1 trillion/sec

Cryptanalysis of Substitution Cipher



CELEBRITY CIPHER
by Luis Campos

Celebrity Cipher cryptograms are created from quotations by famous people, past and present. Each letter in the cipher stands for another.

"U P X E G T H W Z H F X Y L F H O S L N P F X . H M T P J S X E P O X V P G A V G P O S L E E B X X O A P M L N P F X , T P J ' C X Z P W E E B X V B P Z X E B H O S . " — V . W . Y X G V H O

Previous Solution: "Time is the cruelest teacher; first she gives the test, then teaches the lesson." — Leonard Bernstein

TODAY'S CLUE: N equals J

Quick recall: Bitwise-XOR operation

We will use bit-wise XOR:

$$\begin{array}{r} 0101 \\ \oplus\ 1100 \\ \hline 1001 \end{array}$$

<u>Some Properties:</u>

- $X \oplus Y = Y \oplus X$
- $X \oplus X = 000\ldots0$
- $X \oplus Y \oplus X = Y$

Cipher Example: One-Time Pad

Key K: Bitstring of length L

Plaintext M: Bitstring of length L

Encrypt(K,M): Output $K \oplus M$

Decrypt(K,C): Output $K \oplus C$

Example:

$$\begin{array}{r} 0101 \\ \oplus\ 1100 \\ \hline 1001 \end{array}$$

Correctly decrypts because

$$K \oplus C = K \oplus (K \oplus M) = (K \oplus K) \oplus M = M$$

Q: Is the one-time pad secure?

Bigger Q: What does "secure" even mean?

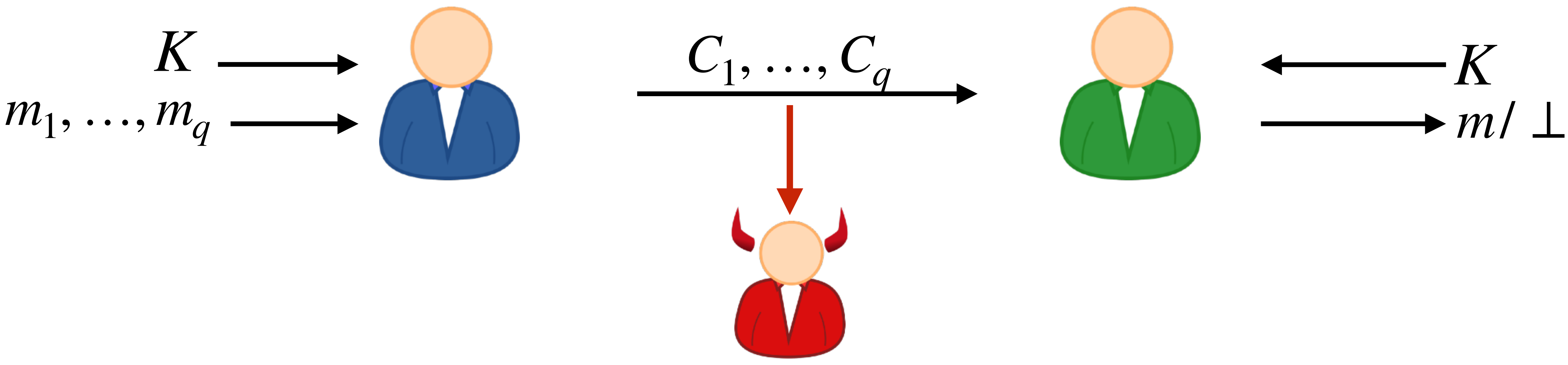
Evaluating Security of Crypto Algorithms

<u>Kerckhoff's Principle</u>: Assume adversary knows your algorithms and implementation. The only thing it doesn't know is the key.

- <u>Example</u>: Adversary knows you are running SSH, and it knows all of the ciphers that SSH allows by looking at the public standard (or downloading the open-source software itself)

- … but it does not know the keys involved

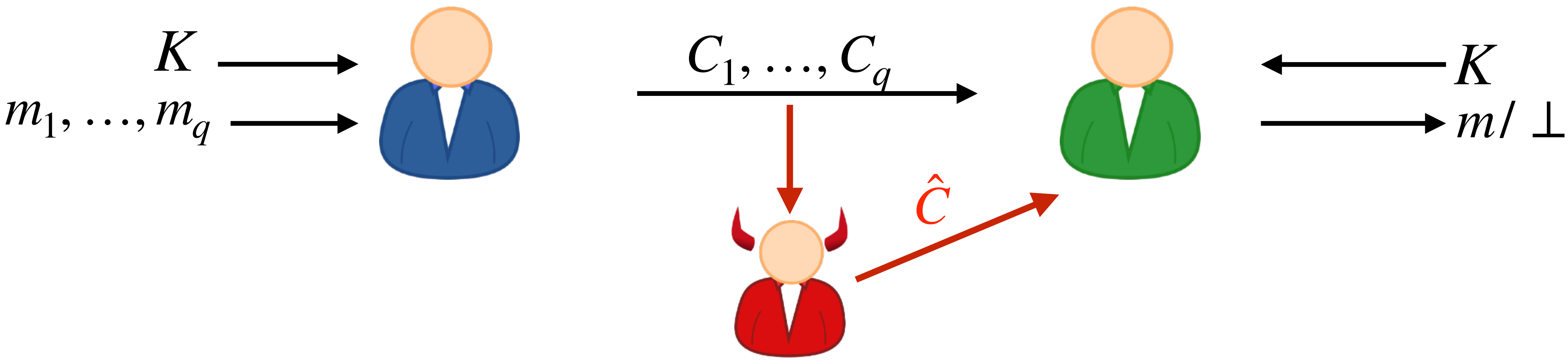
Adversary Goal #1: Break Confidentiality



The adversary sees ciphertexts and attempts to recover some "useful information" about plaintexts.

Other attack settings are important (e.g. adversary can ask for some encryptions, some decryptions…)

Adversary Goal #2: Break Authenticity

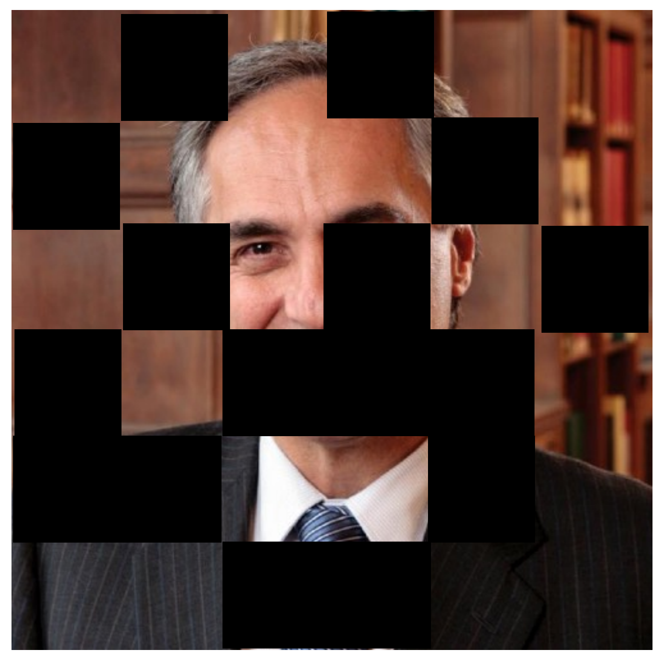


The adversary sees ciphertexts and attempts to create and inject a new ciphertext without being detected by receiver.

Other attack settings are important here too.

Recovering Partial Information; Partial Knowledge

- Recovering entire messages is useful
- But recovering **partial information** is also be useful



A lot of information is missing here.

But can we say who this is?

- Attacker may know large parts of plaintext already (e.g. formatting strings or application content). The attacker tries to obtain something it doesn't already know.

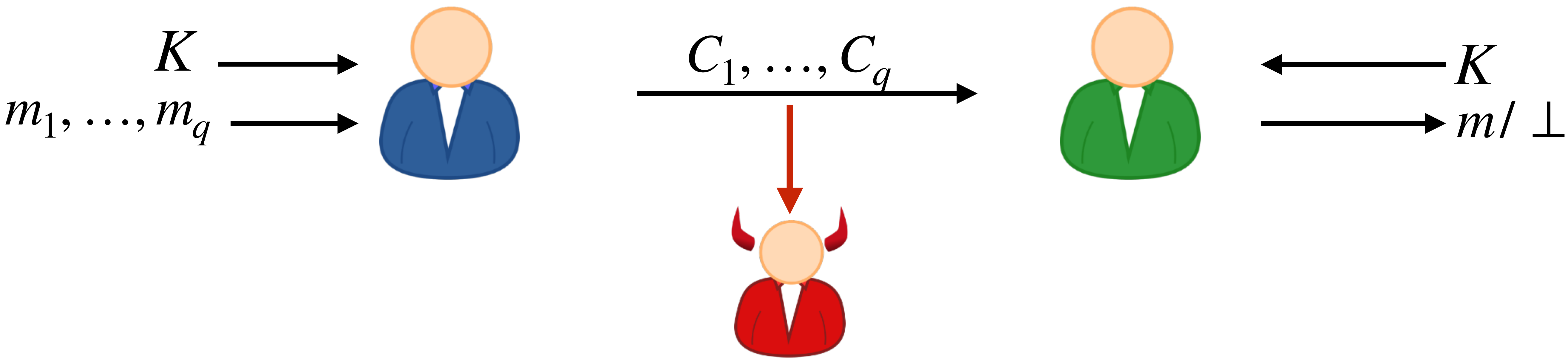
```
M = http://site.com?password=[redacted]
```

Confidentiality Goal for Encryption

An **attack** is successful as long as it recovers <u>some</u> info about plaintext that is useful to adversary.

Encryption should hide <u>all possible partial information</u> about plaintexts, since what is useful is situation-dependent.

Attacks can succeed without recovering the key



Full break: Adversary recovers K, decrypts all ciphertexts.

However: Clever attacker may compromise encryption without recovering the key.

Security of One-Time Pad

Claim: If adversary sees **only one** ciphertext under a random key, then any plaintext is equally likely, so it cannot recover any partial information besides plaintext length.

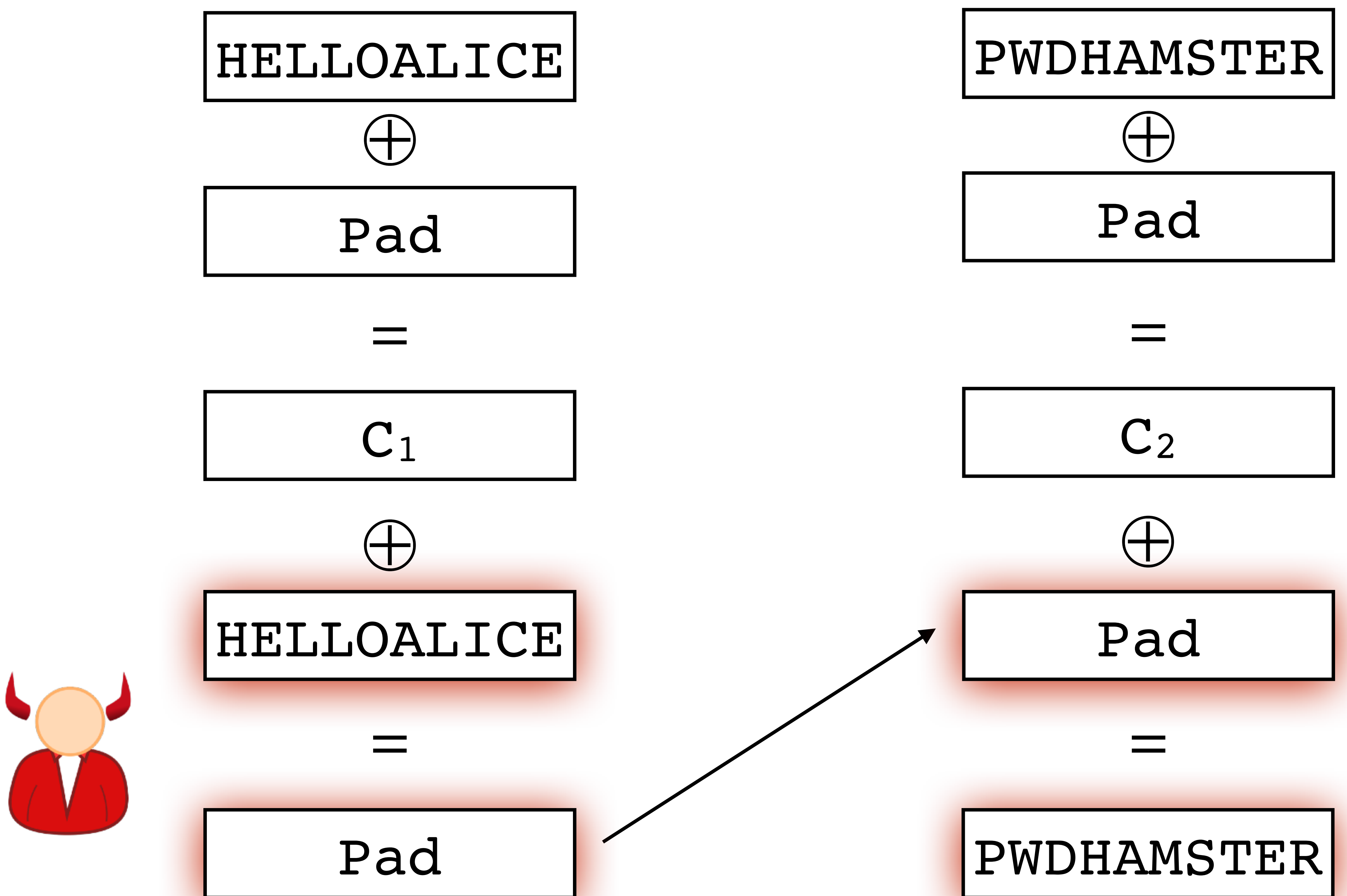
Ciphertext observed:	`10111`
Possible plaintext:	`00101`
$\Rightarrow$ Possible key:	`10010`

1. Adversary goal: Learn partial information from plaintext
2. Adversary capability: Observe a single ciphertext
3. Adversary compute resources: Unlimited time/memory (!)

Issues with One-Time Pad

1. Reusing a pad is insecure
2. One-Time Pad does not provide integrity/authenticity
3. One-Time Pad has a long key

Issue #1: Reusing a One-Time Pad is Insecure

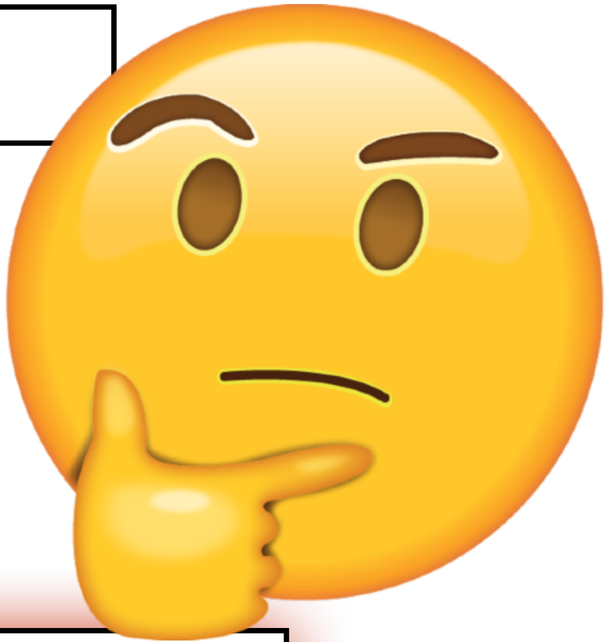


Issue #1: Reusing a One-Time Pad is Insecure

Has led to real attacks:
- Project Venona (1940s) attack by US on Soviet encryption
- MS Windows NT protocol PPTP
- WEP (old WiFi encryption protocol)
- Fortiguard routers! [link]

= c_1

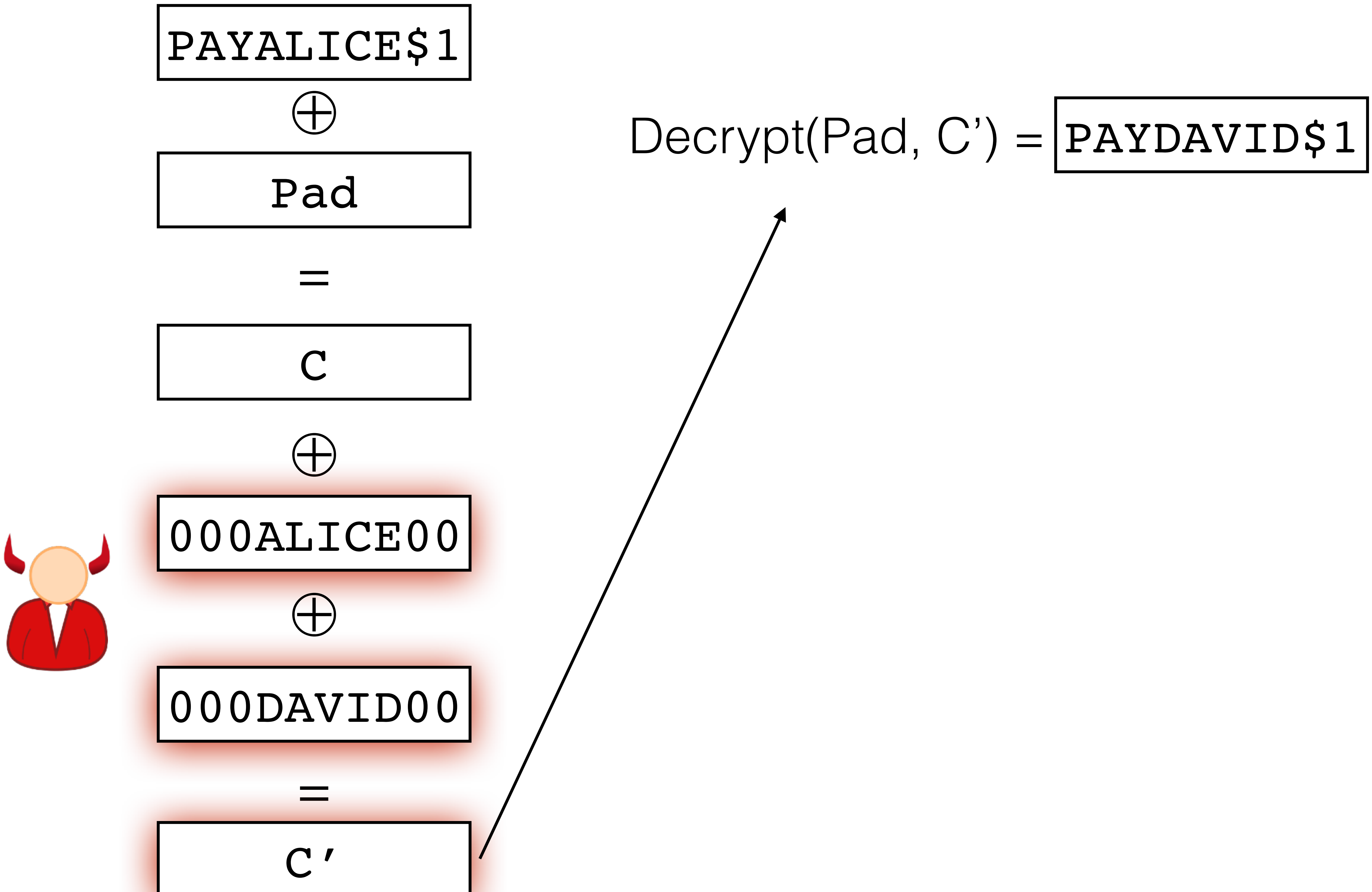
= c_2



$c_1 \oplus c_2$

= S3CR3T1234 $\oplus$ 3L33THXRRR

Issue #2: One-Time Pad Does Not Provide Integrity



Issue #3: One-Time Pad Needs a Long Key

Can prove: Any cipher as secure as the OTP must have:

Key-length ≥ Plaintext-length

In practice:

- Use *stream cipher*: Encrypt(K,m) = G(K)⊕m
- Add *authentication tag*
- Use *nonces* to encrypt multiple messages

Outline

- Symmetric Encryption Basics
- **Stream Ciphers**
- Block Ciphers

Tool to address key-length of OTP: Stream Ciphers

<u>Stream cipher syntax:</u> Algorithm G that takes one input and produces a very long bit-string as output.

Usually very, very large (petabytes if needed)

<u>Key/Seed k:</u> `1100..11` ← Typically 16 or 32 bytes.

G

<u>G(k):</u> `1111101000100011101010010100010110010011110 0…`

$\oplus$ `DONUTSDONUTSDONUTSDONUTSDONUTSDONUTSDONUTSDON`

Use G(seed) in place of pad.
Still malleable and still one-time, but key is shorter.

Stream Cipher Security Goal (Sketch)

Security goal: When `k` is random and unknown, `G(k)` should "look" random.

… even to an adversary spending a lot of computation.

Much stronger requirement that "passes statistical tests".

Brute force attack: Given `y=G(k)`, try all possible `k` and see if you get the string `y`.

Clarified goal: When `k` is random and unknown, `G(k)` should "look" random to anyone with less computational power needed for a brute force attack.

(keylength = 256 is considered strong now)

Aside: Fundamental Physical Property of the Universe*

There exist (1-to-1) functions (say on bitstrings) that are:
1) Very fast to evaluate
2) Computationally infeasible to reverse

The disparity can be almost arbitrarily large!

Evaluating y = f(x) may only take a few cycles….

… and finding x from y within the lifetime of the universe may not be possible, even with a computer made up of every particle in the universe.

**conjectured, but unproven property*

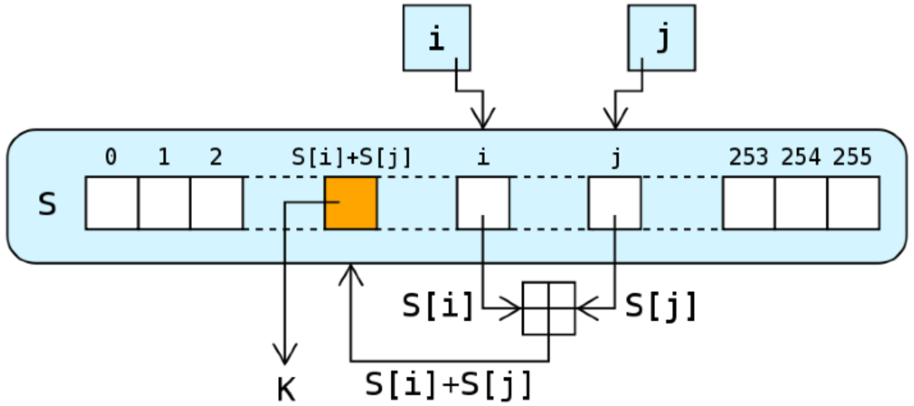
Computational Strength

# Steps	Who can do that many?
2^{56}	Strong computer with GPUs
2^{80}	All computers on Bitcoin network in 4.5 hours
2^{128}	Very large quantum computer? (Ask Diana,Fred,Bill,Robert)*
2^{192}	Nobody?
2^{256}	Nobody?

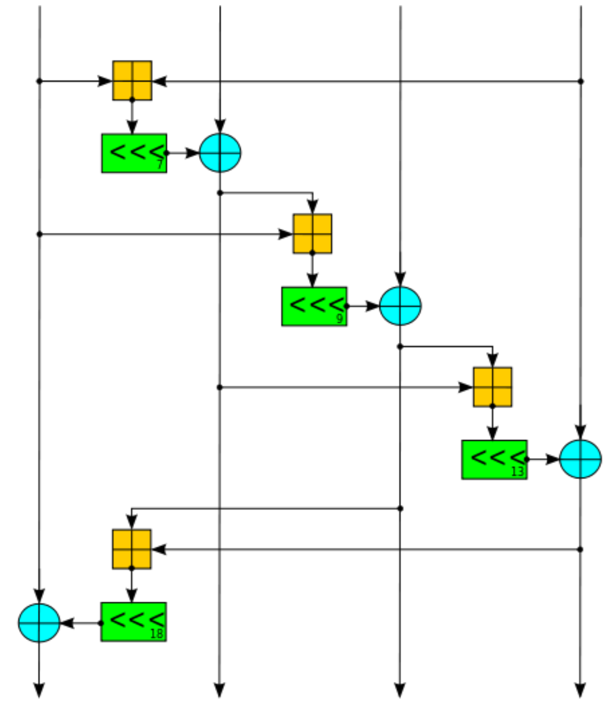
*Not directly comparable but this is an estimate of equivalent power. Quantum computers are most effective against public-key crypto, but they also speed up attacks on symmeric-key crypto. (More next time.)

Practical Stream Ciphers (Not covered in this class)

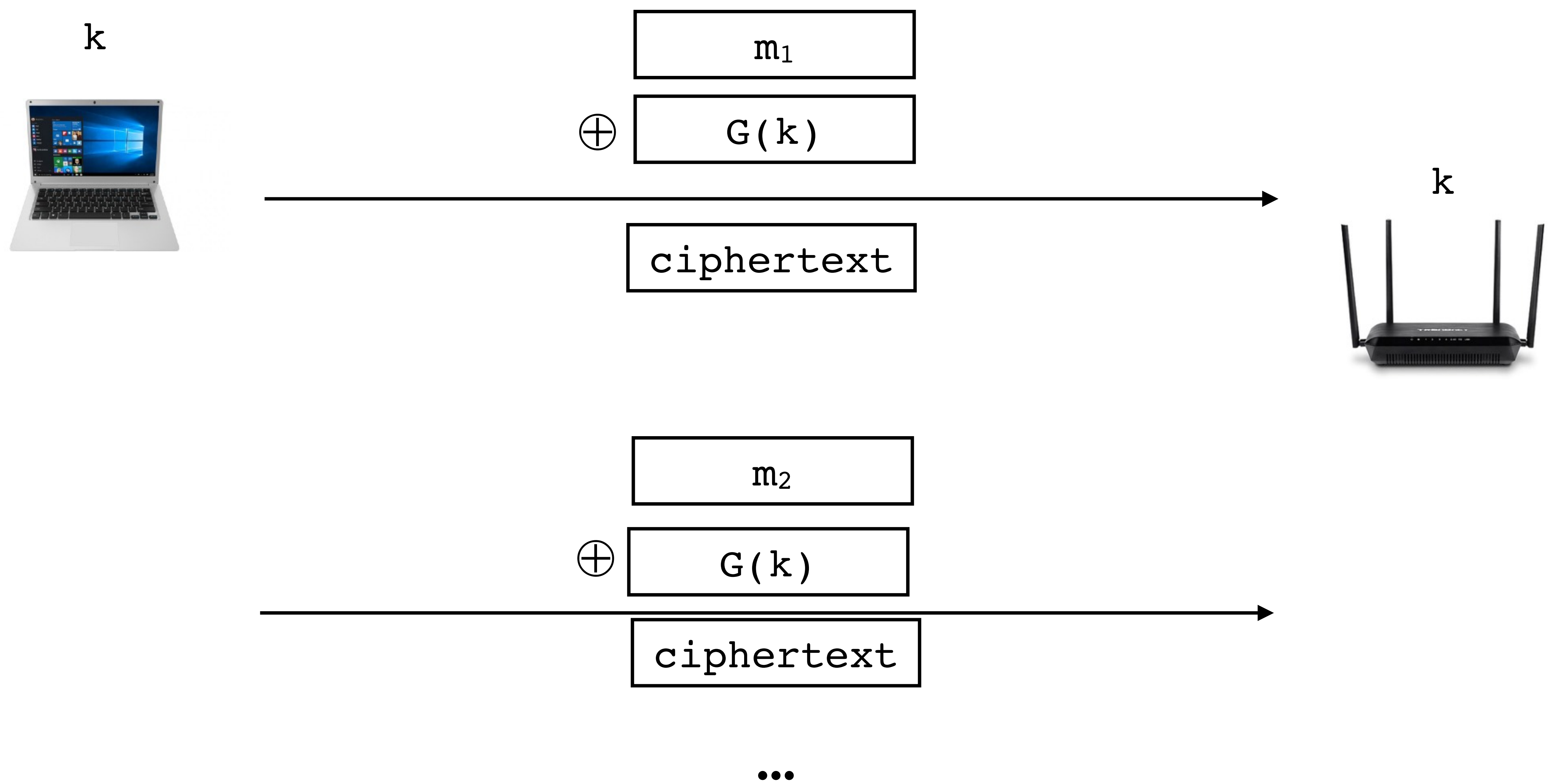
RC4 (1987)**:** "Ron's Cipher #4". Mostly retired by 2016.



ChaCha20 (2007)**:** Successfully deployed replacement. Supports *nonces*.



Pad reuse can still happen with stream ciphers



Addressing pad reuse: Stream cipher with a nonce

<u>Stream cipher with a nonce:</u> Algorithm G that takes **two inputs** and produces a very long bit-string as output.

<u>Nonce IV:</u> `1100..11`

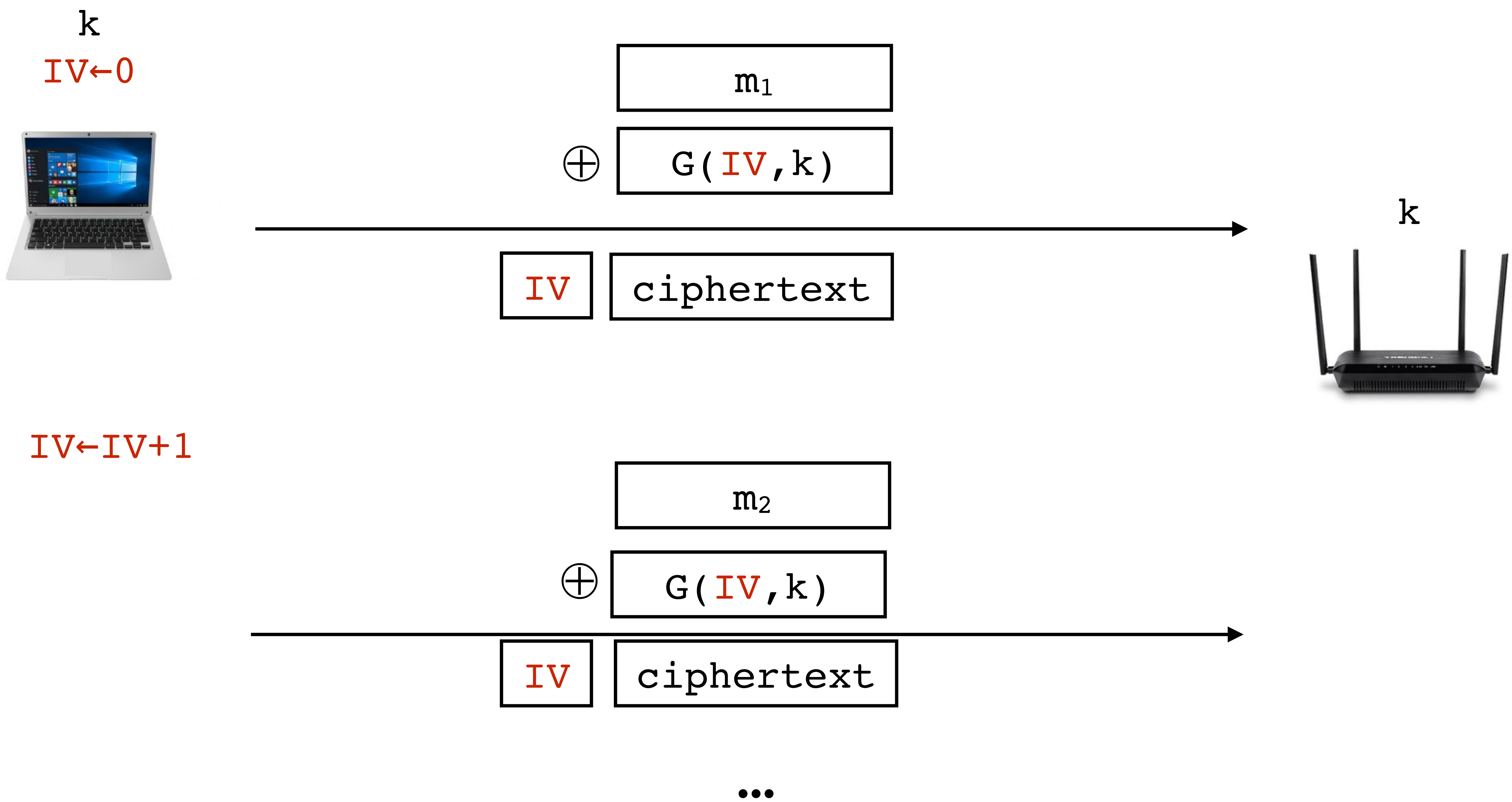
<u>Key/Seed k:</u> `1100..11`

<u>G(IV,k):</u> `11111010001000111010100101000101100100111100…`

- “nonce” = “number once”.
- Usually denoted IV = “initialization vector”

<u>Security goal:</u> When `k` is random and unknown, `G(IV,k)` should “look” random and independent for each value of `IV`.

Solution 1: Stream cipher with a nonce

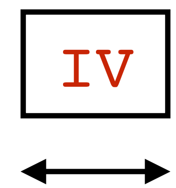


- If nonce repeats, then pad repeats

Example of Pad Re-use: WEP

Warning: Broken

<u>IEEE 802.11b WEP:</u> WiFi security standard '97-'03



IV is 24-bit wide counter

- Repeats after 2^{24} frames (≈16 million)
- IV is often set to zero on power cycle

<u>Solutions: (WPA2 replacement)</u>

- Larger IV space, or force rekeying more often
- Set IV to combination of packet number, address, etc

Example of Pad Re-use: WEP

Warning: Broken

IEEE 802.11b WEP: WiFi security standard '97-'03

BIZ & IT —

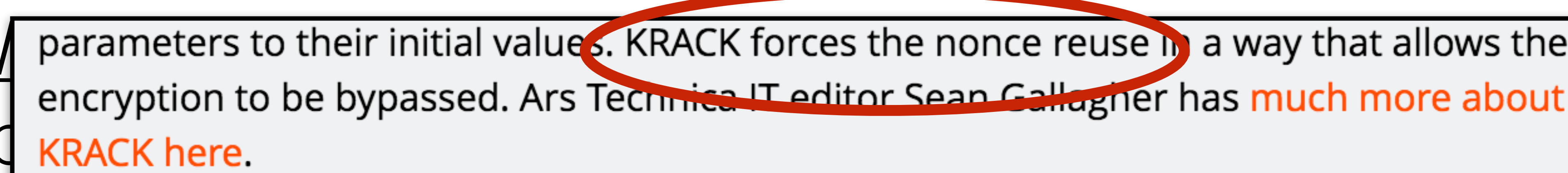
Serious flaw in WPA2 protocol lets attackers intercept passwords and much more

KRACK attack is especially bad news for Android and Linux users.

DAN GOODIN - 10/15/2017, 11:37 PM

- Re
- Of

Solutions: (W



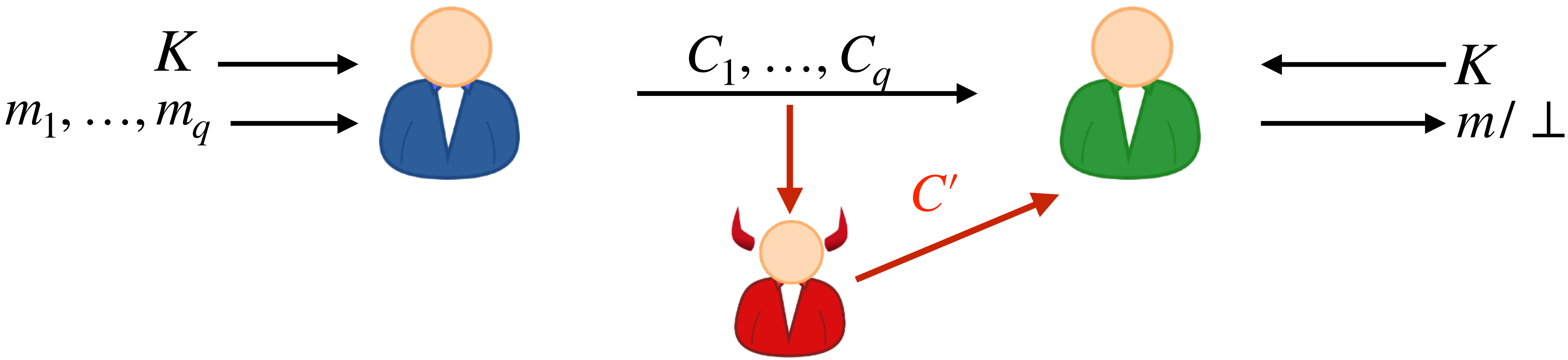
parameters to their initial values. KRACK forces the nonce reuse in a way that allows the encryption to be bypassed. Ars Technica IT editor Sean Gallagher has much more about KRACK here.

- Larger IV sp
- Set IV to combination of packet number, address, etc

Issues with One-Time Pad

1. Reusing a pad is insecure ✓ *Use unique nonces*
2. One-Time Pad is does not provide integrity/authenticity
3. One-Time Pad has a long key ✓ *Use stream cipher with short key*

Adversary Goal #2: Break Authenticity



The adversary sees ciphertexts and attempts to create and inject a new ciphertext without being detected by receiver.

Other attack settings are important here too.

Stream ciphers do not give integrity

```
M = please pay ben 20 bucks

C = b0595fafd05df4a7d8a04ced2d1ec800d2daed851ff509b3e446a782871c2d
```

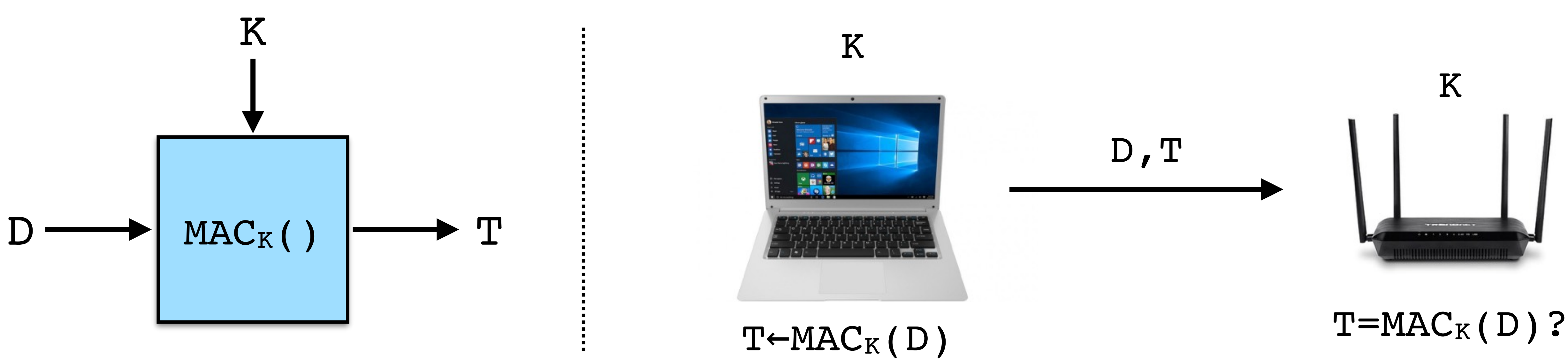
```
C'= b0595fafd05df4a7d8a04ced2d1ec800d2daed851ff509b3e546a782871c2d

M' = please pay ben 21 bucks
```

Inherent to stream-cipher approach to encryption.

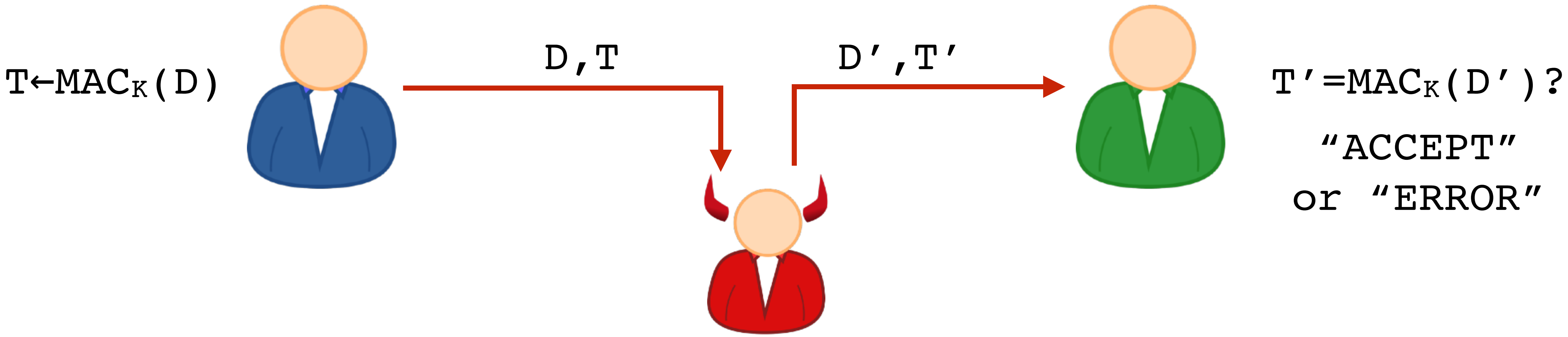
Message Authentication Code

A **message authentication code (MAC)** is an algorithm that takes as input a key and a message, and outputs an "unpredictable" **tag.**



D will usually be a ciphertext, but is often called a "message".

MAC Security Goal: Unforgeability



MAC satisfies **unforgeability** if it is infeasible for Adversary to fool Bob into accepting `D'` not previously sent by Alice.

MAC Security Goal: Unforgeability

Note: No encryption on this slide.

`D = please pay ben 20 bucks`

`T = 827851dc9cf0f92ddcdc552572ffd8bc`

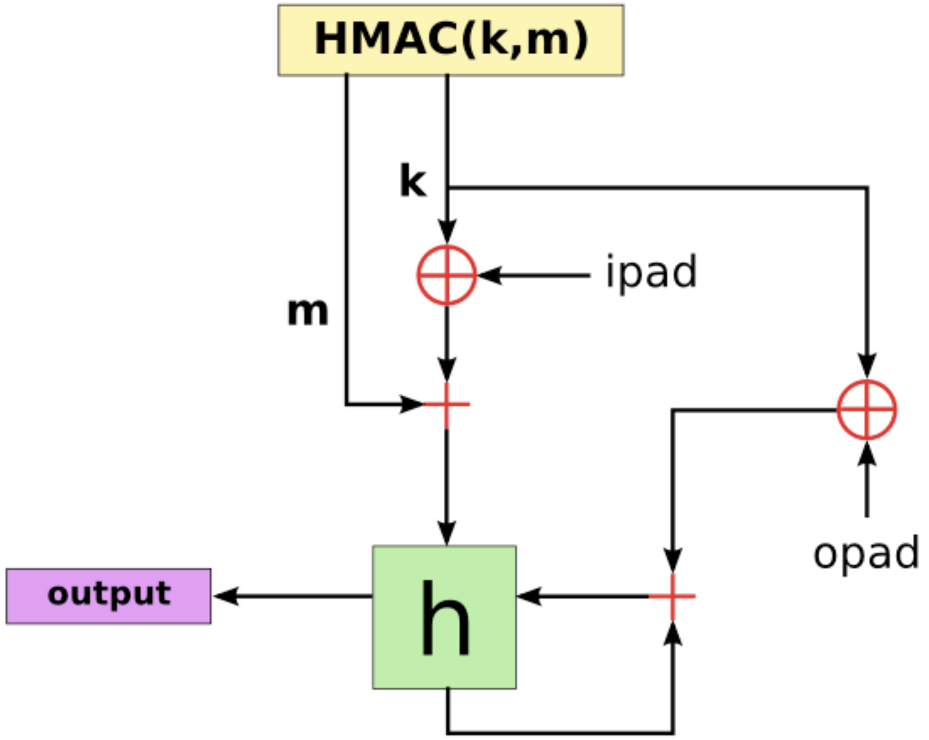
`D'= please pay ben 21 bucks`

`T'= baeaf48a891de588ce588f8535ef58b6`

Should be hard to predict **`T'`** for any new **`D'`**.

MACs In Practice: Use HMAC or Poly1305-AES

- More precisely: Use HMAC-SHA2. More on hashes and MACs in a moment.



- Other, less-good option: AES-CBC-MAC (bug-prone)

Authenticated Encryption

Encryption that provides **confidentiality** and **integrity** is called **Authenticated Encryption**.

- Built using a good stream cipher and a MAC.
  - Ex: Salsa20 with HMAC-SHA2
- Best solution: Use ready-made Authenticated Encryption
  - Ex: AES-GCM is the standard

The End