

CSE 484 / CSE M 584: Cryptography

Winter 2022

Tadayoshi (Yoshi) Kohno
yoshi@cs

Announcements

- Feb 14: Alex Gantman (Qualcomm) – entirely virtual lecture
- Feb 16: Lucy Simko (UW) – entirely virtual lecture
- Feb 17: Final Project Part #1 deadline
 - https://courses.cs.washington.edu/courses/cse484/22wi/assignments/final_project.html
 - Groups strongly encouraged; groups of up to 3 people
 - 12–15-minute video, at non-accelerated or non-slowed timing (this is a security class, after all :P)
 - Use guest lectures, homework 1, **homework 2**, reading of news, personal interests, and any other resource you wish for inspiration

Announcements (continued)

- HW2: Online now (or soon)
- HW3: Cryptography (also online soon)
- Structurally, I moved the crypto homework (usually HW2) to HW3 because more compatible with where we are at
 - Usually, we announce HW2 before we are finished with the crypto discussions
 - Given timing for this quarter, better for the crypto homework to be due toward the end of the quarter
 - For HW3, I will tell you as we go which parts you should already be able to do now, in case you would like to get a head start

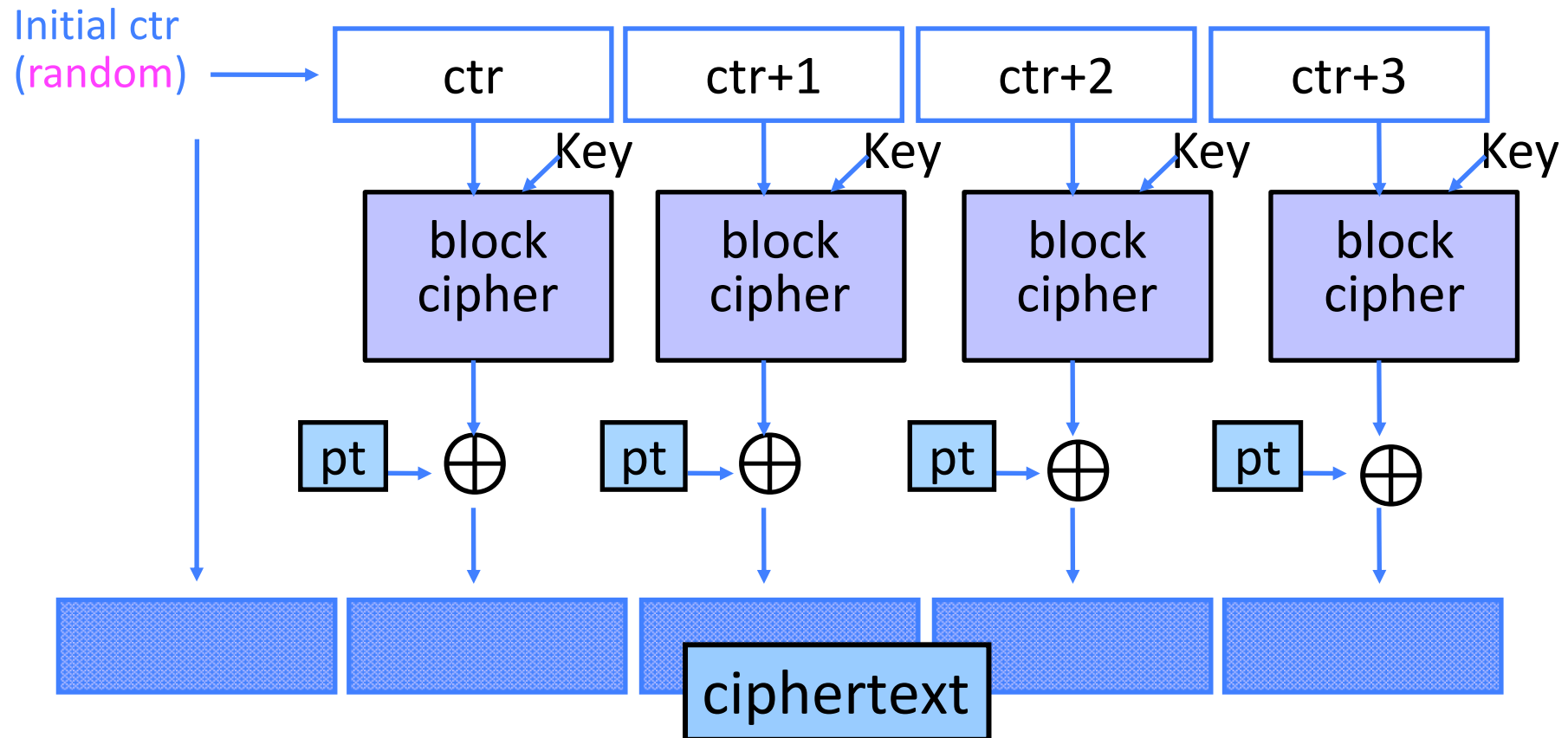
Announcements (continued)

- Homework 2
 - More on security mindset
 - Relationship between people+society and technology
- From HW2 preamble:
 - Technologies do not exist in isolation. Technologies exist in the context of people and society. Societies have policies and laws and norms. The people might be the direct users of a technology, or they might be impacted by a technology even if they are not the direct users.
 - Computer security offers one vantage point to thinking about the relationship between technologies and people + society. The reason: in computer security and privacy, we consider adversaries. There are other vantages points, like considering algorithmic bias, but for the purposes of this course we will focus on computer security and privacy.
 - Homework 2, like Homework 1, is (1) an opportunity to revisit the security mindset, this time with the context of the first portion of the course and with a renewed focus on the relationship between technology and people+society. Additionally, Homework 2 (2) provides an opportunity to spend some time thinking about a technology that might be relevant to your final project (if you wish).

Announcements (continued)

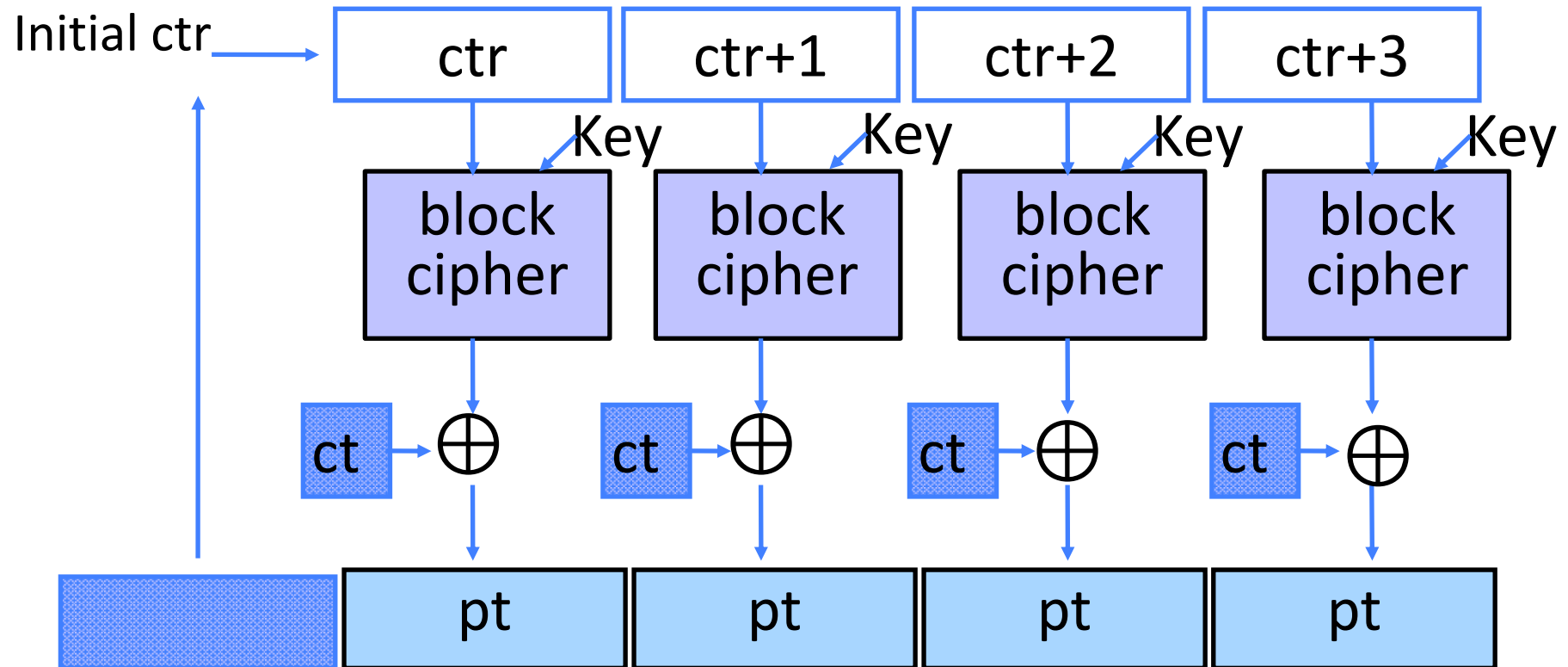
- Physical security lecture

Counter Mode (CTR): Encryption



- Identical blocks of plaintext encrypted differently
- **Still does not guarantee integrity; Fragile if ctr repeats**

Counter Mode (CTR): Decryption



Ok, so what mode do I use?

- Don't choose a mode, use established libraries 😊
- Good modes:
 - GCM - Galois/Counter Mode
 - CTR (sometimes)
 - Even ECB is fine in “the right circumstance” ← so much packed into “the right circumstances” however, so best to avoid

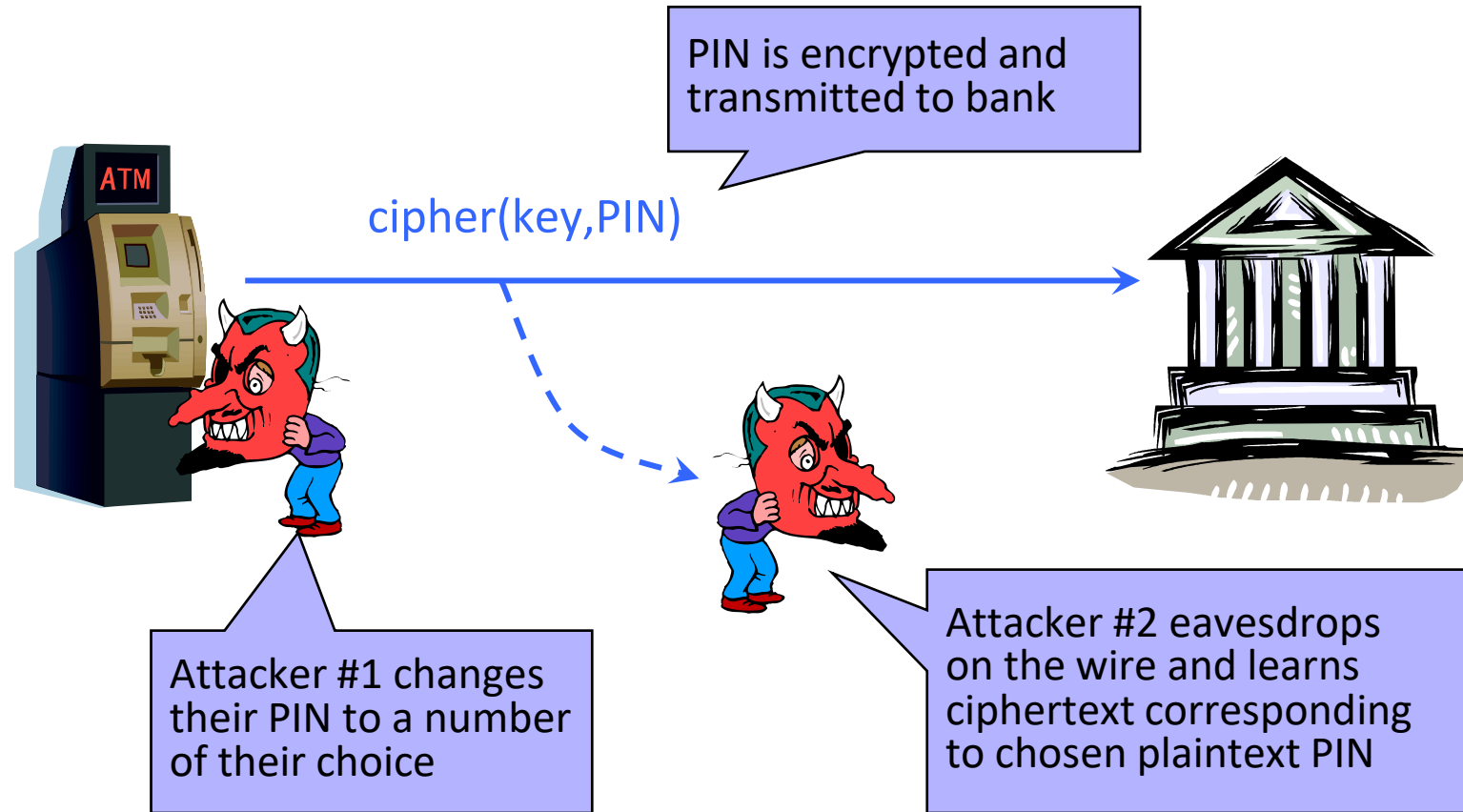
When is an Encryption Scheme “Secure”?

- Hard to recover the key?
 - What if attacker can learn plaintext without learning the key?
- Hard to recover plaintext from ciphertext?
 - What if attacker learns some bits or some function of bits?

How Can a Cipher Be Attacked?

- Attackers knows ciphertext and encryption algorithm
 - **What else does the attacker know?** Depends on the application in which the cipher is used!
- **Ciphertext-only attack**
- **KPA: Known-plaintext attack** (stronger)
 - Knows some plaintext-ciphertext pairs
- **CPA: Chosen-plaintext attack** (even stronger)
 - Can obtain ciphertext for any plaintext of choice
- **CCA: Chosen-ciphertext attack** (very strong)
 - Can decrypt any ciphertext except the target

Chosen Plaintext Attack



... repeat for any PIN value

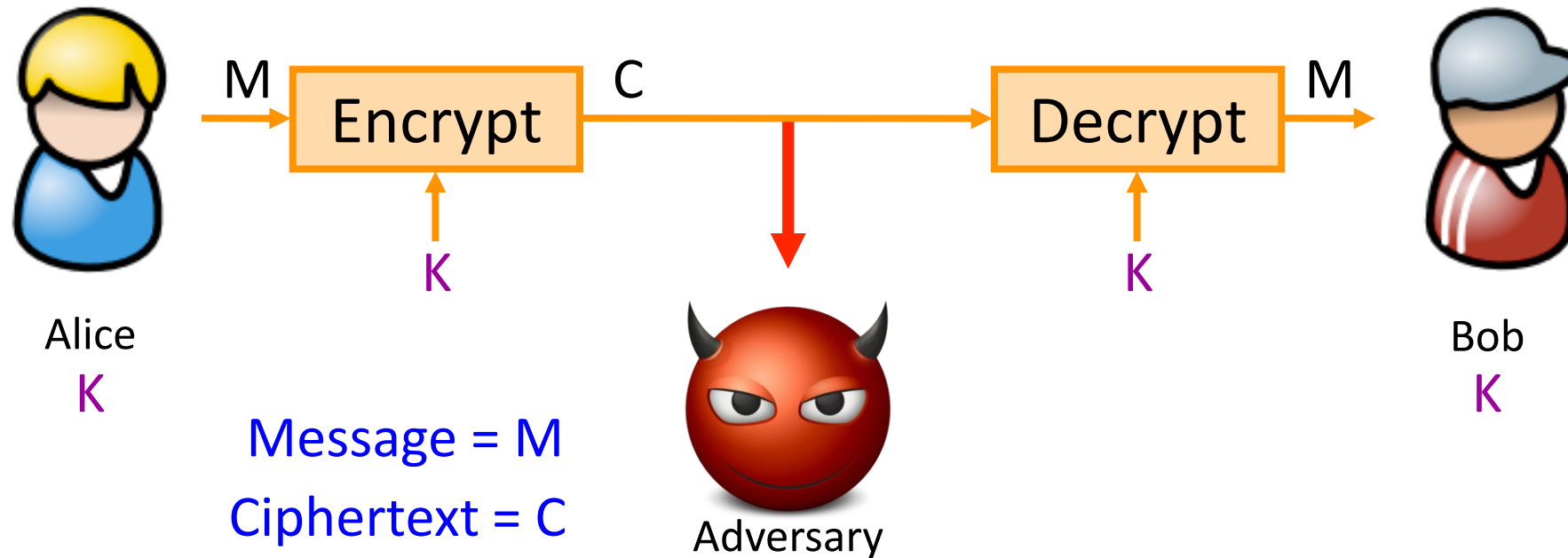
Very Informal Intuition

Minimum security requirement for a modern encryption scheme

- Security against chosen-plaintext attack (CPA)
 - Ciphertext leaks no information about the plaintext
 - Even if the attacker correctly guesses the plaintext, they cannot verify their guess
 - Every ciphertext is unique, encrypting same message twice produces completely different ciphertexts
 - Implication: encryption must be randomized or stateful
- Security against chosen-ciphertext attack (CCA)
 - Integrity protection – it is not possible to change the plaintext by modifying the ciphertext

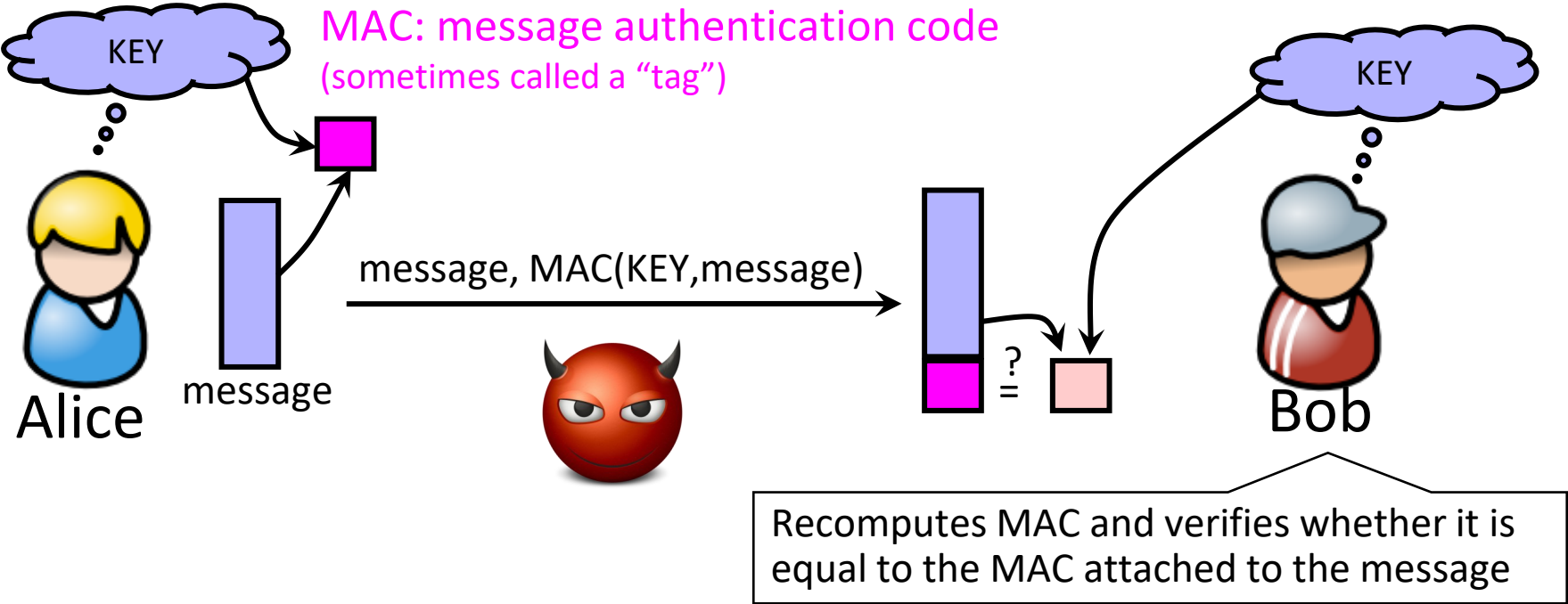
So Far: Achieving Privacy

Encryption schemes: A tool for protecting **privacy**.



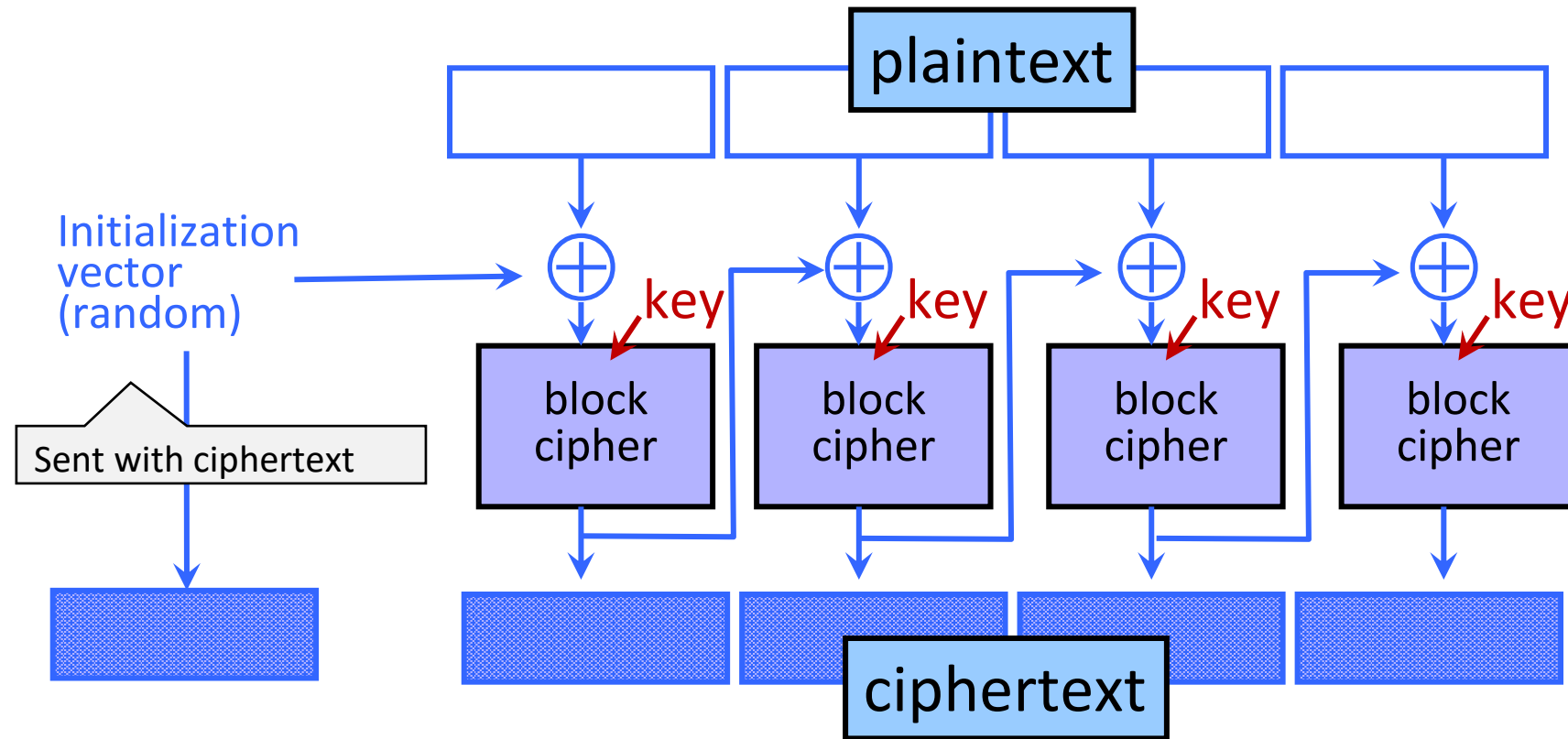
Now: Achieving Integrity

Message authentication schemes: A tool for protecting integrity.



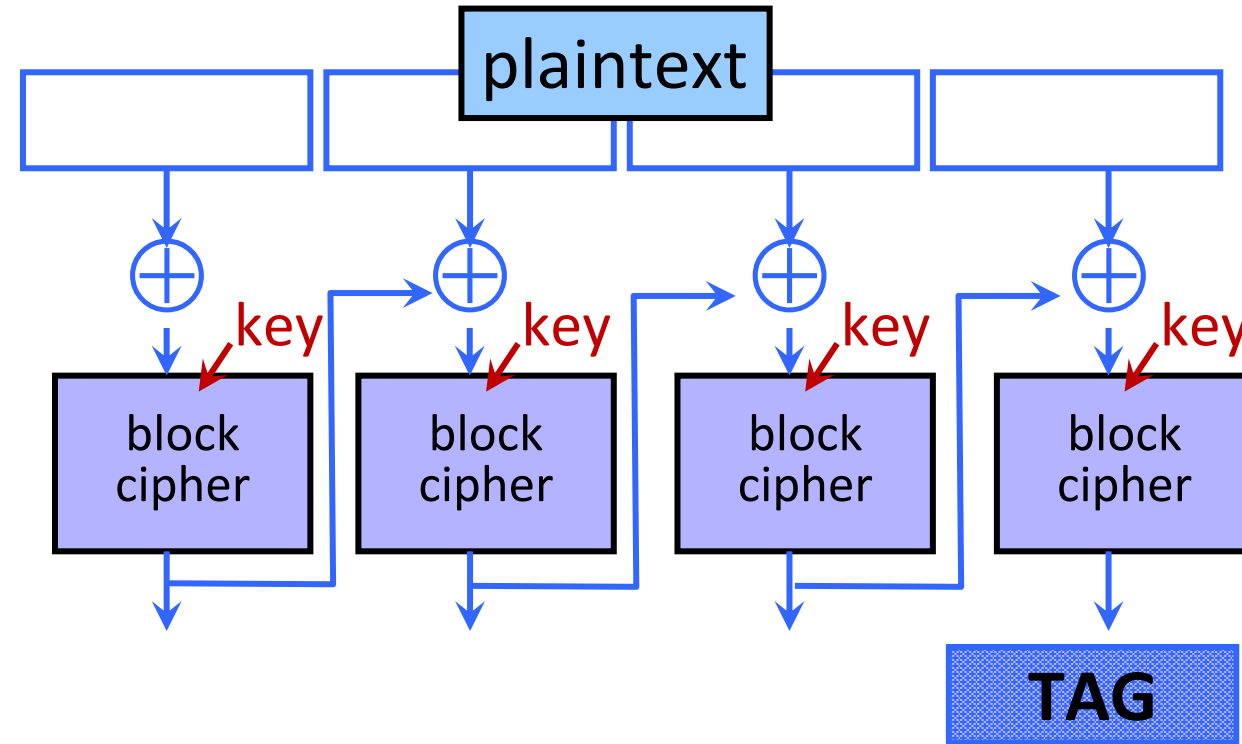
Integrity and authentication: only someone who knows KEY can compute correct MAC for a given message.

Reminder: CBC Mode Encryption



- Identical blocks of plaintext encrypted differently
- Last cipherblock depends on entire plaintext
 - Still does not guarantee integrity

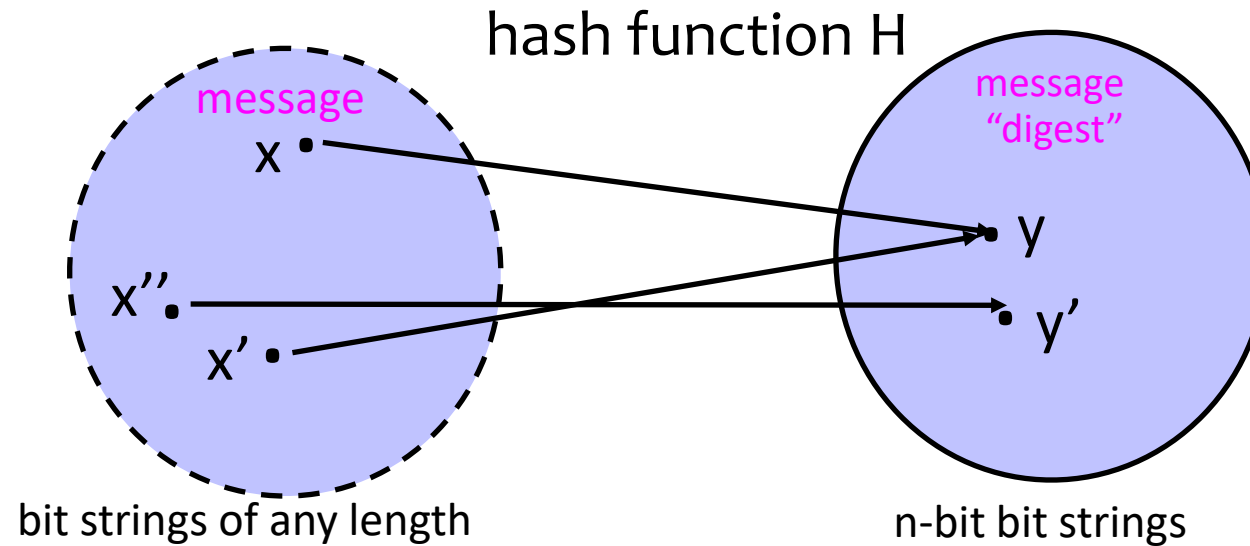
CBC-MAC



- Not secure when system may MAC messages of different lengths (*more in section!*).
- Use a different key – not encryption key
- NIST recommends a derivative called CMAC [FYI only]

Another Tool: Hash Functions

Hash Functions: Main Idea



- Hash function H is a lossy compression function
 - Collision: $h(x)=h(x')$ for distinct inputs x, x'
- $H(x)$ should look "random"
 - Every bit (almost) equally likely to be 0 or 1
- Cryptographic hash function needs a few properties...

Property 1: One-Way

- Intuition: hash should be hard to invert
 - “Preimage resistance”
 - Let $h(x') = y$ in $\{0,1\}^n$ for a random x'
 - Given y , it should be hard to find any x such that $h(x)=y$
- How hard?
 - Brute-force: try every possible x , see if $h(x)=y$
 - SHA-1 (common hash function) has 160-bit output
 - Expect to try 2^{159} inputs before finding one that hashes to y .

Property 2: Collision Resistance

- Should be hard to find $x \neq x'$ such that $h(x) = h(x')$

Birthday Paradox

- Are there two people in the ~first page of people on Zoom (depending on the size of your window) or first few rows of class that have the same birthday?
 - 365 days in a year (366 some years)
 - Pick one person. To find another person with same birthday would take on the order of $365/2 = 182.5$ people
 - **Expect birthday “collision” with a room of only 23 people.**
 - For simplicity, approximate when we expect a collision as **$\text{sqrt}(365)$** .
- Why is this important for cryptography?
 - 2^{128} different 128-bit values
 - Pick one value at random. To exhaustively search for this value requires trying on average 2^{127} values.
 - **Expect “collision” after selecting approximately 2^{64} random values.**
 - **64 bits** of security against collision attacks, not 128 bits.

Property 2: Collision Resistance

- Should be hard to find $x \neq x'$ such that $h(x) = h(x')$
- Birthday paradox means that brute-force collision search is **only** $O(2^{n/2})$, *not* $O(2^n)$
 - For SHA-1, this means $O(2^{80})$ vs. $O(2^{160})$

One-Way vs. Collision Resistance

One-wayness does not imply collision resistance.

Collision resistance does not imply one-wayness.

You can prove this by constructing a function that has one property but not the other.

One-Way vs. Collision Resistance

(Details here mainly FYI)

- One-wayness does not imply collision resistance
 - Suppose g is one-way
 - Define $h(x)$ as $g(x')$ where x' is x except the last bit
 - h is one-way (to invert h , must invert g)
 - Collisions for h are easy to find: for any x , $h(x0)=h(x1)$
- Collision resistance does not imply one-wayness
 - Suppose g is collision-resistant
 - Define $y=h(x)$ to be $0x$ if x is n -bit long, $1g(x)$ otherwise
 - Collisions for h are hard to find: if y starts with 0 , then there are no collisions, if y starts with 1 , then must find collisions in g
 - h is not one way: half of all y 's (those whose first bit is 0) are easy to invert ([how?](#)); random y is invertible with probab. $\frac{1}{2}$

Property 3: Weak Collision Resistance

- Given randomly chosen x , hard to find x' such that $h(x)=h(x')$
 - Attacker must find collision for a specific x . By contrast, to break collision resistance it is enough to find any collision.
 - Brute-force attack requires $O(2^n)$ time
- Weak collision resistance does not imply collision resistance.

Hashing vs. Encryption

- Hashing is one-way. There is no “un-hashing”
 - A ciphertext can be decrypted with a decryption key... hashes have no equivalent of “decryption”
- Hash(x) looks “random” but can be compared for equality with Hash(x’)
 - Hash the same input twice → same hash value
 - Encrypt the same input twice → different ciphertexts
- Cryptographic hashes are also known as “cryptographic checksums” or “message digests”

Application: Password Hashing

- Instead of user password, store `hash(password)`
- When user enters a password, compute its hash and compare with the entry in the password file
- Why is hashing better than encryption here?
 - Breakout

Application: Password Hashing

- Instead of user password, store `hash(password)`
- When user enters a password, compute its hash and compare with the entry in the password file
- Why is hashing better than encryption here?
- System does not store actual passwords
- Don't need to worry about where to store the key
- Cannot go from hash to password

Application: Password Hashing

- Which property do we need?
 - One-wayness?
 - (At least weak) Collision resistance?
 - Both?

Application: Password Hashing + Salting

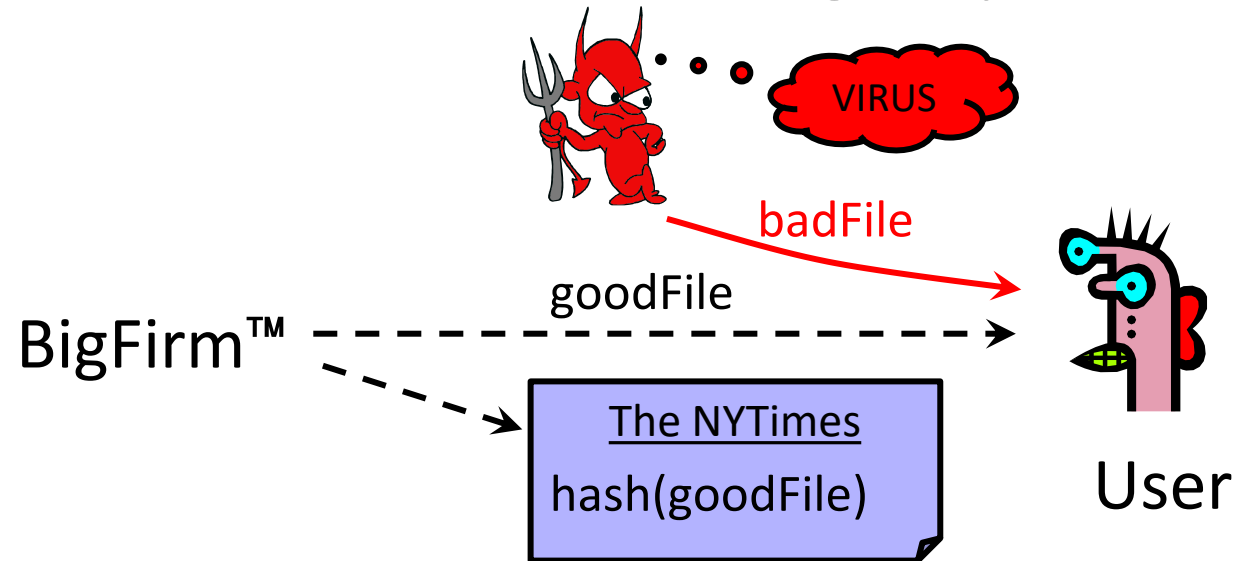
- **Salting**

- We 'salt' hashes for password by adding a randomized suffix to the password
 - E.g. Hash("coolpassword"+"35B67C2A")
 - We then store the salt with the hashed password!
 - Server generates the salt
-
- The goal is to prevent *precomputation attacks*
 - If the adversary doesn't know the salt, they can't *precompute* common passwords

Hash Functions Review

- Map large domain to small range (e.g., range of all 160- or 256-bit values)
- Properties:
 - Collision Resistance: Hard to find two distinct inputs that map to same output
 - One-wayness: Given a point in the range (that was computed as the hash of a random domain element), hard to find a preimage
 - Weak Collision Resistance: Given a point in the domain and its hash in the range, hard to find a new domain element that maps to the same range element

Application: Software Integrity



Goal: Software manufacturer wants to ensure file is received by users without modification.

Idea: given goodFile and hash(goodFile), very hard to find badFile such that hash(goodFile)=hash(badFile)

Application: Software Integrity

- Which property do we need?
 - One-wayness?
 - (At least weak) Collision resistance?
 - Both?

Which Property Do We Need?

One-wayness, Collision Resistance, Weak CR?

- UNIX passwords stored as hash(password)
 - **One-wayness:** hard to recover the/a valid password
- Integrity of software distribution
 - **Weak collision resistance**
 - But software images are not really random... may need **full collision resistance** if considering malicious developers

Common Hash Functions

- **SHA-2: SHA-256, SHA-512, SHA-224, SHA-384**
- **SHA-3: standard released by NIST in August 2015**
- MD5 – **Don't Use!**
 - 128-bit output
 - Designed by Ron Rivest, used very widely
 - Collision-resistance broken (summer of 2004)
- RIPEMD
 - 160-bit version is OK
 - 128-bit version is *not* good
- SHA-1 (Secure Hash Algorithm) – **Don't Use!**
 - 160-bit output
 - US government (NIST) standard as of 1993-95
 - Theoretically broken 2005; practical attack 2017!

SHA-1 Broken in Practice (2017)

Google just cracked one of the building blocks of web encryption (but don't worry)

It's all over for SHA-1

by [Russell Brandom](#) | [@russellbrandom](#) | Feb 23, 2017, 11:49am EST

<https://shattered.io>

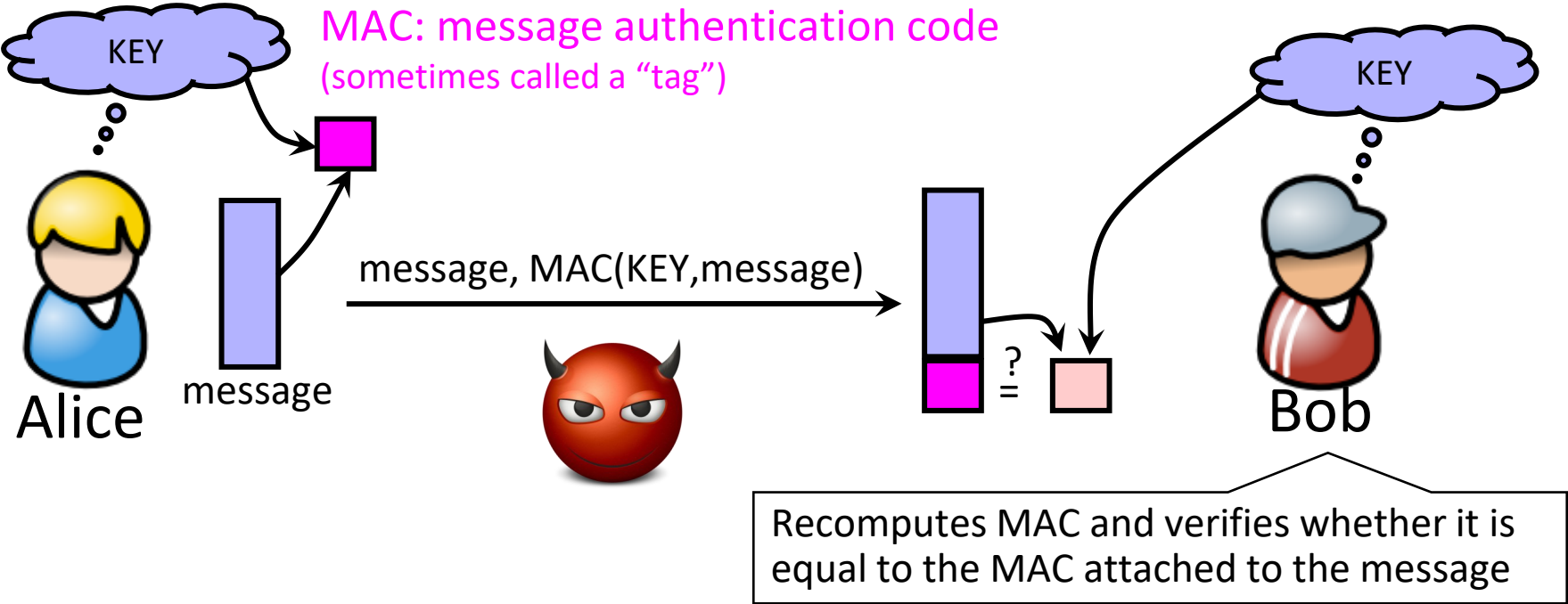


Aside: How we evaluate hash functions

- Speed
 - Is it amenable to hardware implementations?
- Diffusion
 - Does changing 1 bit in the input affect all output bits?
- Resistance to attack approaches
 - Collisions?
 - Length extensions?
 - etc

Recall: Achieving Integrity

Message authentication schemes: A tool for protecting integrity.



Integrity and authentication: only someone who knows KEY can compute correct MAC for a given message.

HMAC

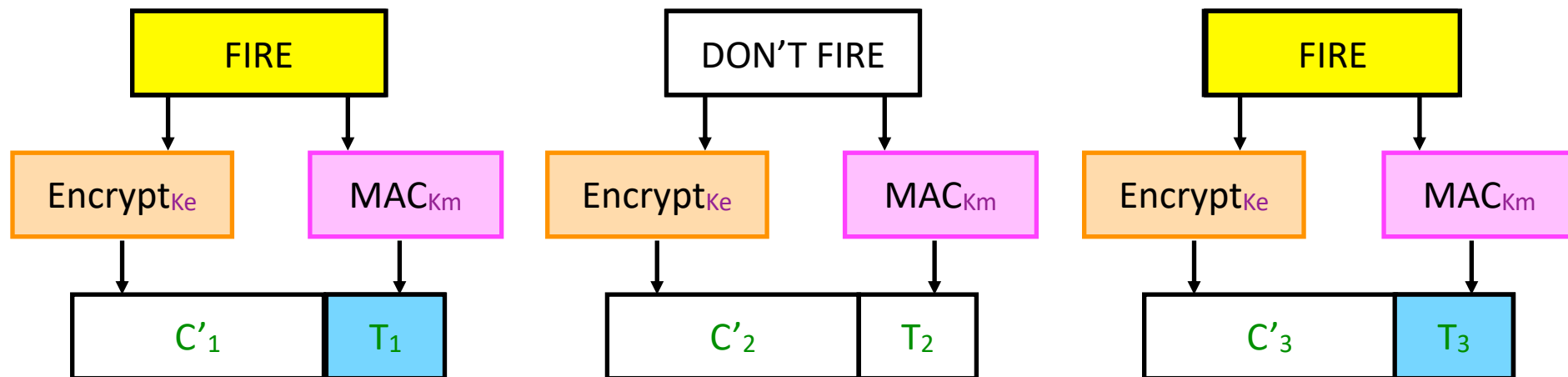
- Construct MAC from a cryptographic hash function
 - Invented by Bellare, Canetti, and Krawczyk (1996)
 - Used in SSL/TLS, mandatory for IPsec
- Why not encryption? (Historical reasons)
 - Hashing is faster than block ciphers in software
 - Can easily replace one hash function with another
 - There used to be US export restrictions on encryption

MAC with SHA3

- $\text{SHA3}(\text{Key} || \text{Message})$
- SHA3 has some nice features that prevent the class of attacks HMAC prevents

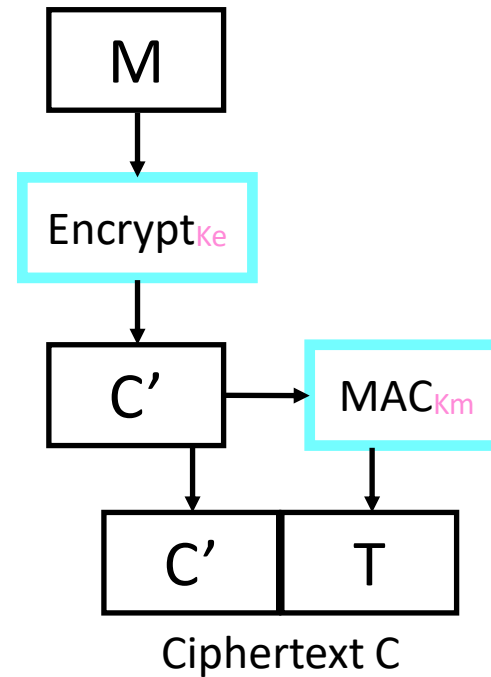
Authenticated Encryption

- What if we want both privacy and integrity?
- Natural approach: combine **encryption scheme** and a **MAC**.
- **But be careful!**
 - Obvious approach: Encrypt-and-MAC
 - Problem: MAC is deterministic! same plaintext \rightarrow same MAC



Authenticated Encryption

- Instead:
Encrypt then MAC.
- (Not as good:
MAC-then-Encrypt)



Encrypt-then-MAC

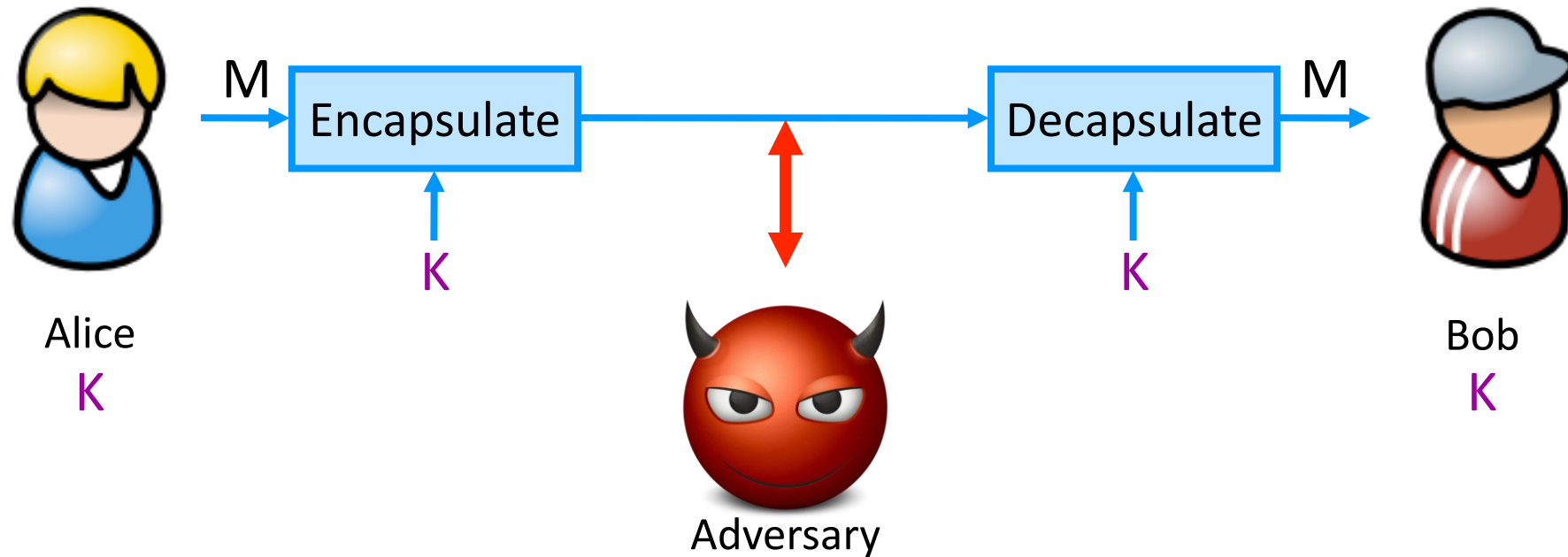
Back to cryptography land

Stepping Back: Flavors of Cryptography

- Symmetric cryptography
 - Both communicating parties have access to a **shared random string K** , called the **key**.
- Asymmetric cryptography
 - Each party creates a public key **pk** and a secret key **sk** .

Symmetric Setting

Both communicating parties have access to a **shared random string K** , called the **key**.



Asymmetric Setting

Each party creates a public key pk and a secret key sk .

