Lecture 2: Crypto Overview

CPSC 444/544, 2026 Spring

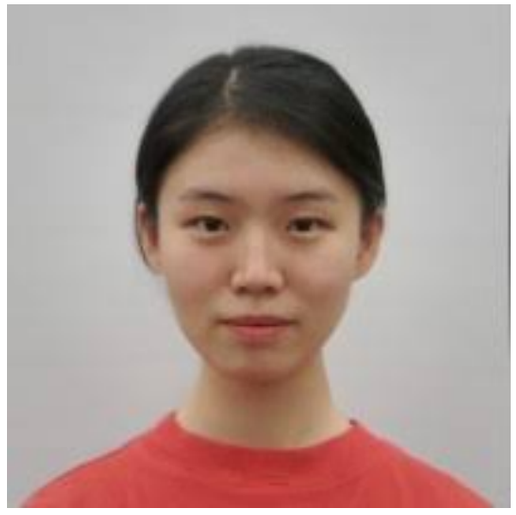
Instructor: Fan Zhang

Yale

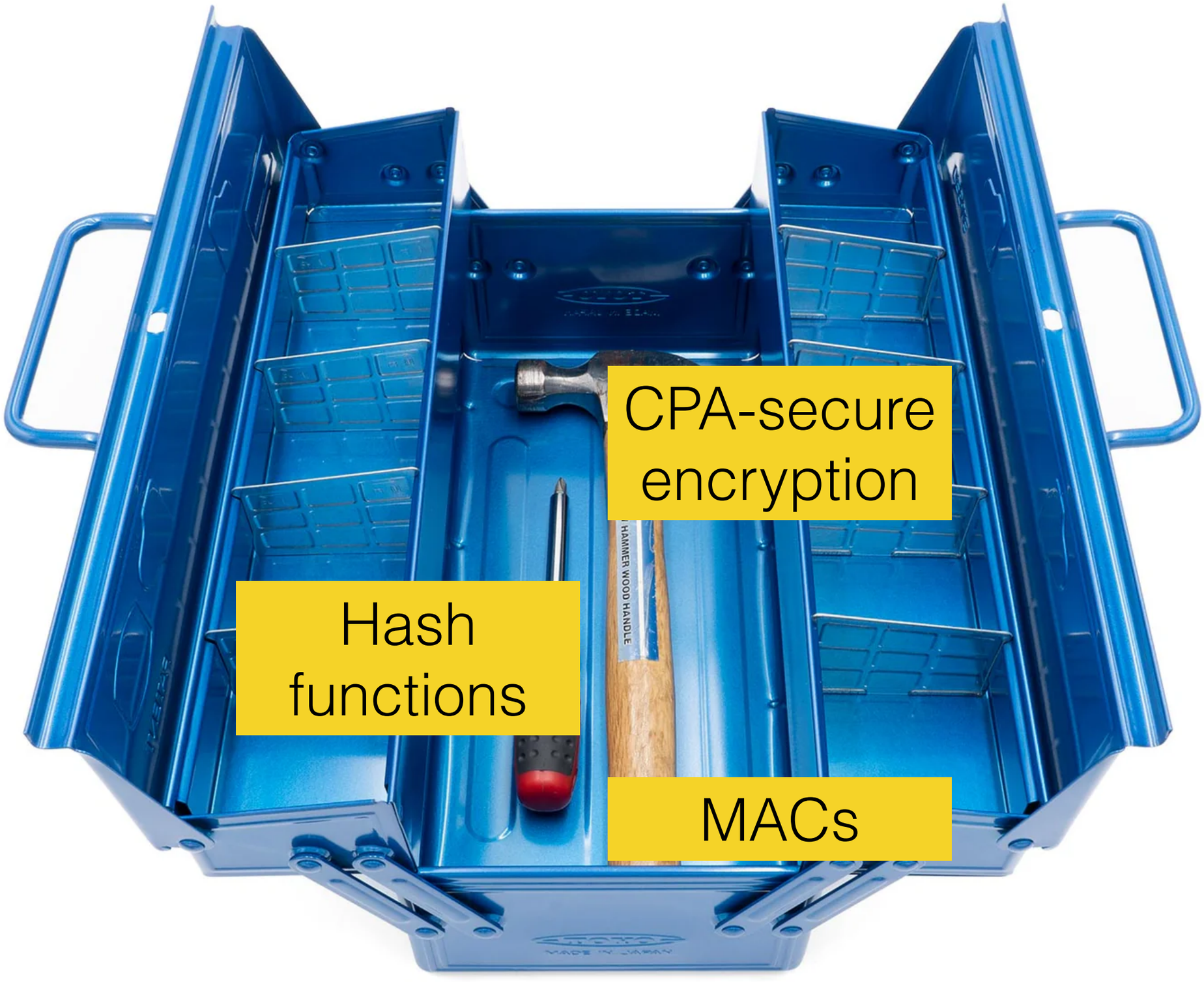
Computer Science

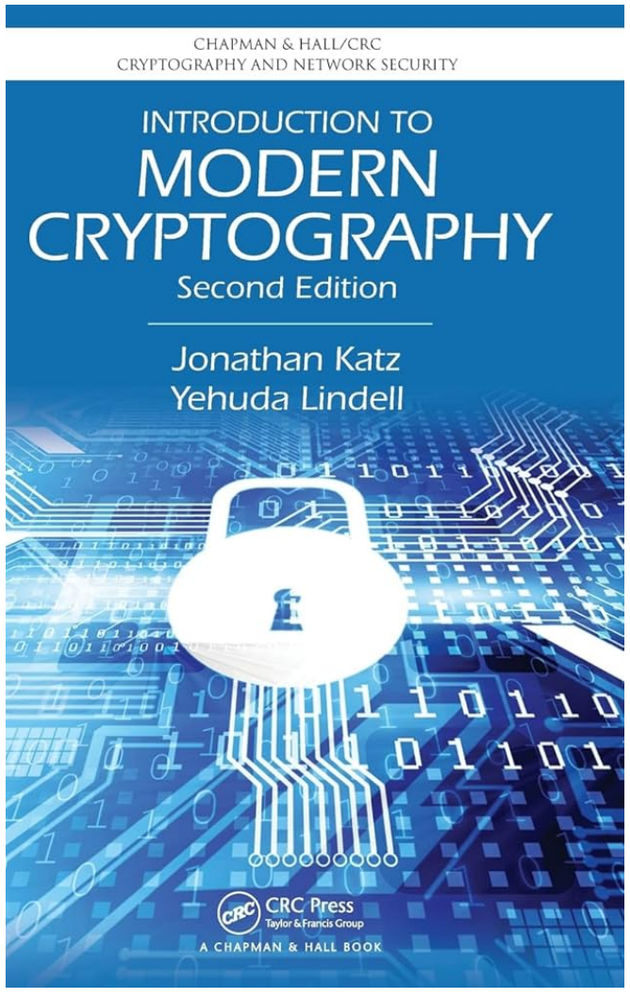
Oh

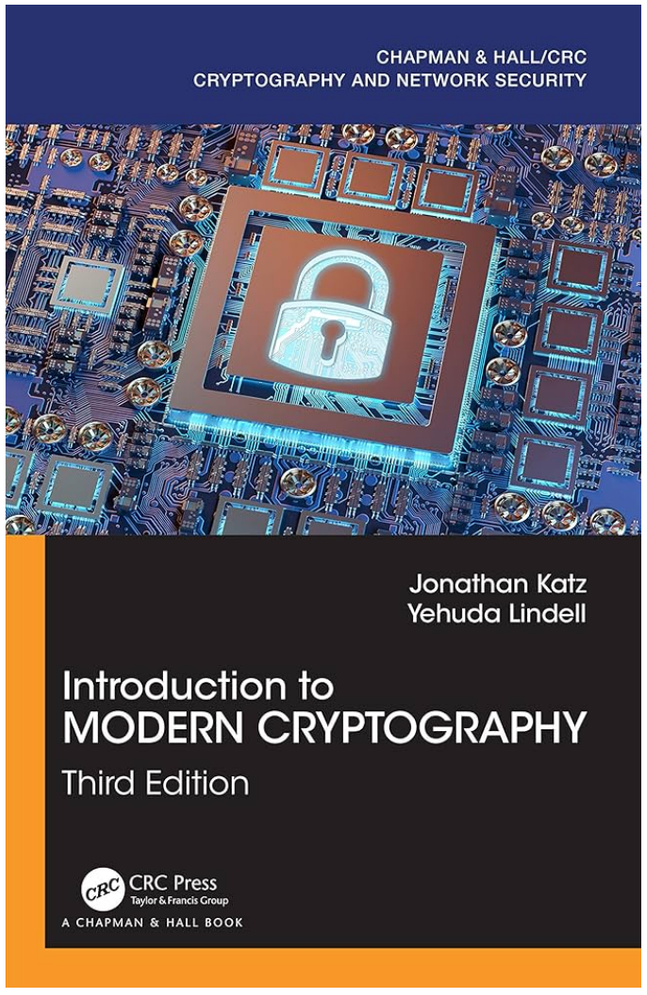
- 4 pm Wed or by appointment (starting next week)
- AKW 503



- TA: Yunhao Wang
- OH: 4-5 pm Thursday
- AKW 508a


CPA-secure encryption
Hash functions
MACs





Introduction to modern cryptography

Author	Katz, Jonathan, 1974-
Published	Boca Raton, FL : CRC Press, 2021.
Online	✔ Online book
Location	Yale Internet Resource >> None
Format	Books / Online

- I use [KL Page XYZ] to refer to **3rd Edition**.
  - Digital version freely available through Yale Library
  - All content we refer to should also be in 2nd Edition with possibly different page numbers.

Ask this three questions

1. **Security goal:** What policy or good state is meant to be enforced?
2. **Adversarial model:** Who is the adversary? What is the adversary's space of possible actions?
3. **Mechanisms:** Are the right security mechanisms in place to achieve the security goal given the adversarial model?

Symmetric-key encryption

Syntax of symmetric-key encryption

- Gen(1^n): outputs a key K.
  - We assume K is securely distributed to Alice and Bob.
- Enc(K, m): outputs a ciphertext encrypting m
- Dec(K, c): recovers the original message from c

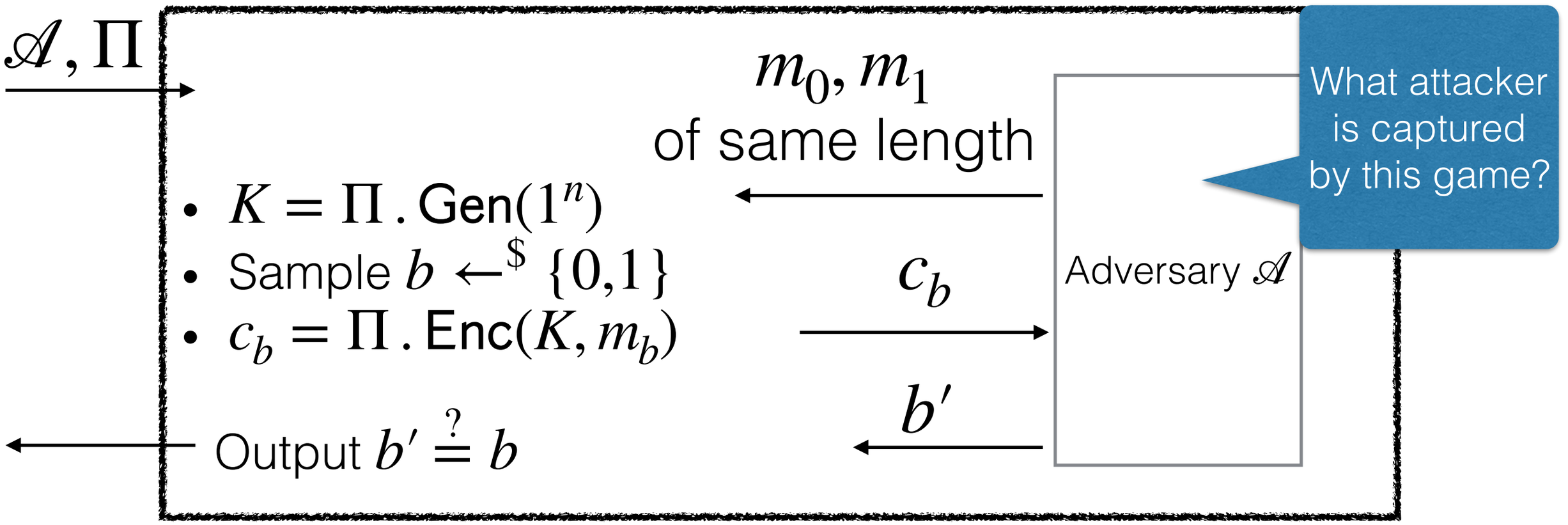
Security goals

- How about “the attacker can’t recover m”?
- Not good enough!
- The attacker also shouldn’t learn
  - last few bits of m
  - f(m) for some function f
- Took cryptographers until 80’s - 90’s to figure out
  - Intuition: “anything about m that she doesn’t already know”

Security games

Game $\mathsf{G}^{\text{IND-EAV}}(\mathscr{A}, \Pi)$

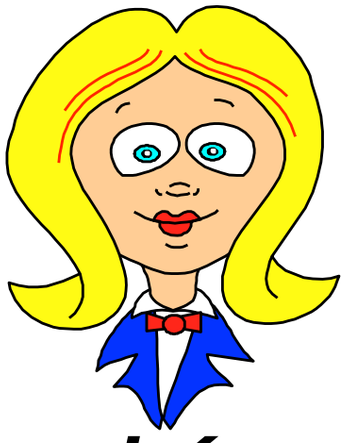
(indistinguishability against eavesdropper)



Π satisfies indistinguishability against eavesdropper if

$\forall$ PPT $\mathscr{A}$, $|Pr[\mathsf{G}^{\text{IND-EAV}}(\mathscr{A}, \Pi) = 1] - 1/2| < \mathsf{neg}(n)$.

Adversary in this game: an **Eavesdropper**



Alice

K

(secret)

$C = \mathrm{enc}_K[M]$

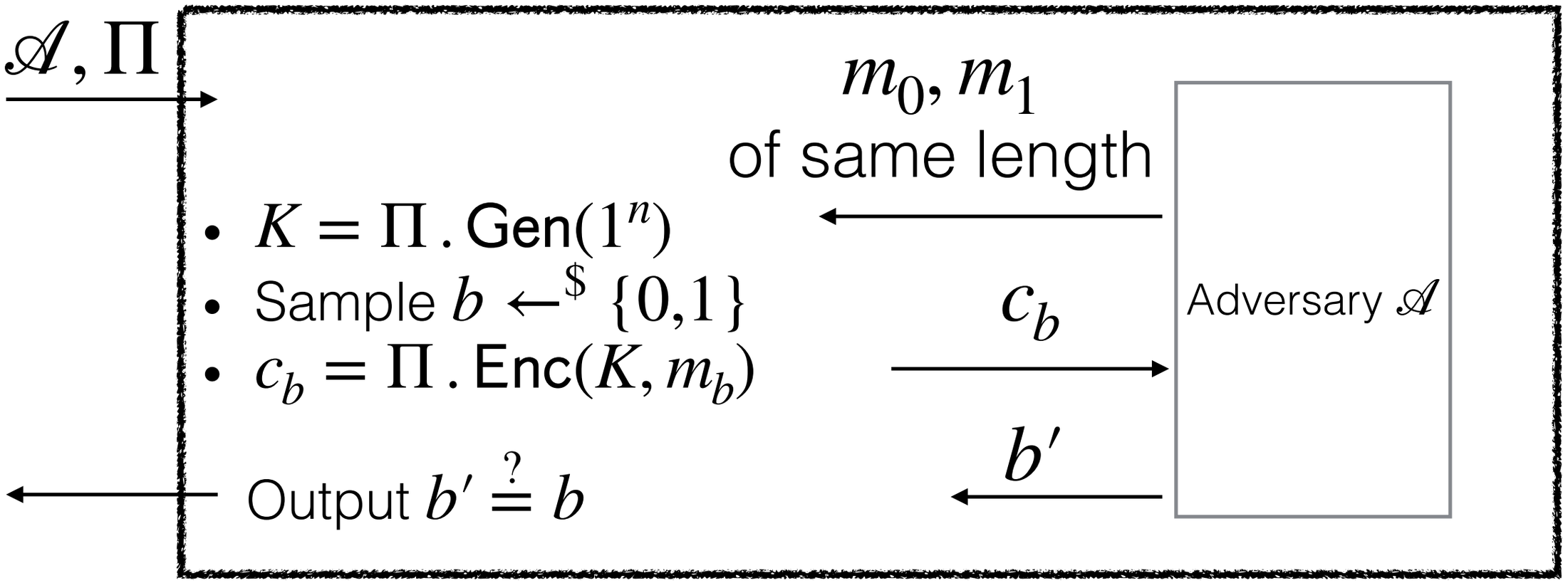
Bob

K

What's *M*?

Eve

Game $\mathbf{G}^{\text{IND-EAV}}(\mathcal{A}, \Pi)$



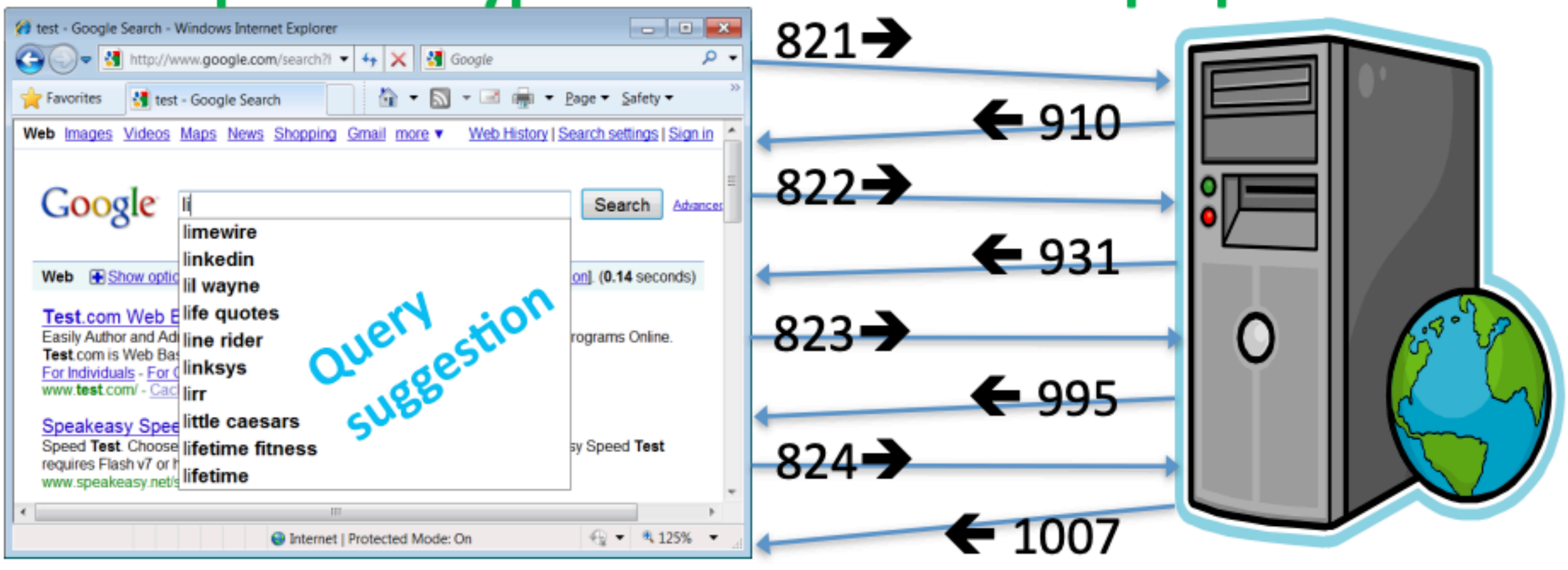
- Model v.s. Reality: Why same length?
  - Answer: o.w. too hard to hide
- What's the implication?

Shuo Chen; Rui Wang; XiaoFeng Wang; Kehuan Zhang. *Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow.* IEEE S&P 2010.

Google bing YAHOO!

Scenario: search using encrypted Wi-Fi WPA/WPA2.

Example: user types "*list*" on a WPA2 laptop.



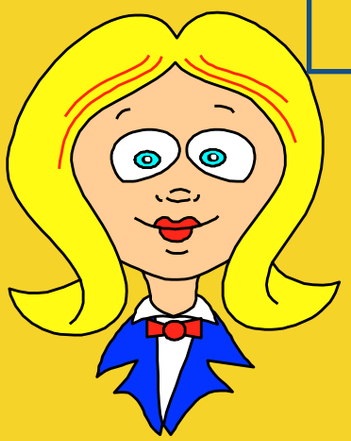
Attacker's effort: linear, not exponential.

Consequence: Anybody on the street knows our search queries.

"Unbreakable" cipher: One-time pad

Key K random bit string; same length as message

Decrypt by XORing out K; $M = C \oplus K$



$C = \text{enc}_K[M]$

$K = 1001010$
$M = 0101101$

$C = 1100111$

$K = 1001010$
$C = 1100111$

$M = 0101101$

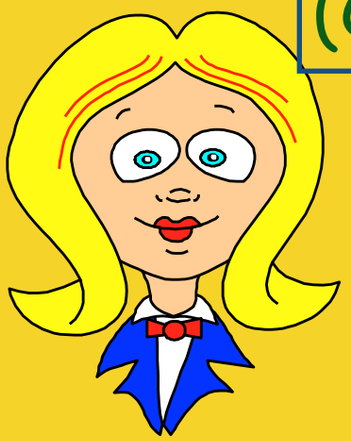
Ciphertext C is bitwise XOR of K and C

One-time pad

Perfect secrecy if every K equally likely… because:

* For any M, every possible C equally likely!
* So C reveals no information about M!

(C. Shannon, 1949)



$C = \text{enc}_K[M]$

$K = 1001010$
$M = 0101101$

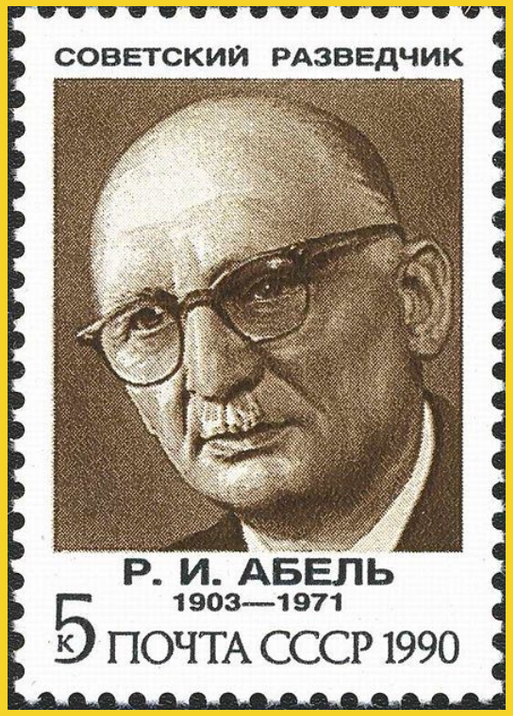
$C = 1100111$

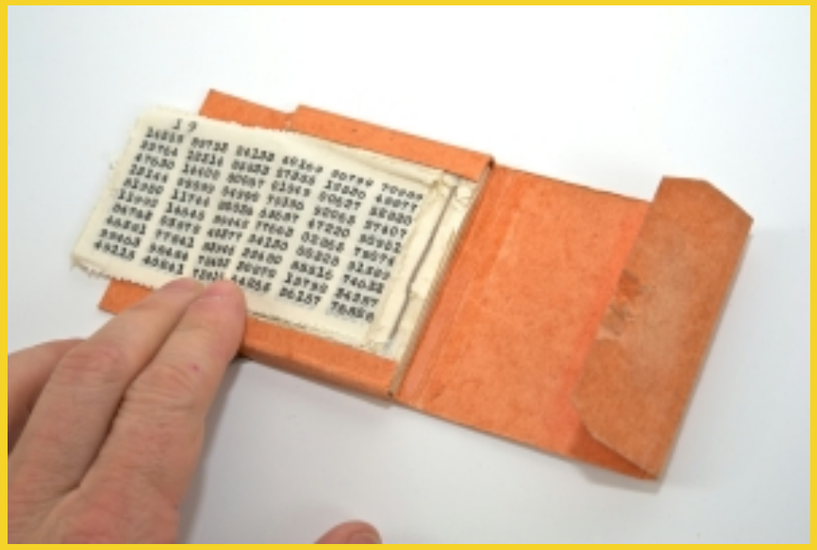
$K = 1001010$
$C = 1100111$

$M = 0101101$

"Unbreakable" cipher: One-time pad

- We will soon see why it's "one-time"
- Some high profile uses
  - KGB agents and controllers
    - E.g., Colonel Rudolf Abel, active in NYC, 1950s
  - Hotlines between Moscow and Washington D.C., Canberra and Moscow, etc.
    - U.S.-Moscow line created in1963 after Cuban missile crisis
    - Teleprinters with one-time tape system
    - Keying tapes delivered via embassies





Perfect secrecy, but...

- One-time pad is one-time
  - Breakable if used twice
  - It happened: Canberra-Moscow Line broken because Soviets reused Moscow-D.C. pad!
- Key length = message length very cumbersome!
  - E.g., how can Alice encrypt her laptop hard drive? Alice carries around another hard drive containing the key?
- Operationally impractical

Relaxed goal: *Computational* Security

- A) Only defend against *efficient* attackers who finishes in $poly(n)$ time:
  - n is security parameter (e.g., key length)
  - I.e., we give limited time to the adversary
- B) We allow attackers to succeed with *negligible probability* (e.g., 2^{-n})
- Security theorems typically look like

> A scheme is *secure* if for every *probabilistic polynomial-time* adversary $\mathcal{A}$ carrying out an attack (of some formally specified type), the probability that $\mathcal{A}$ succeeds in the attack (where success is also formally specified) is *negligible*.

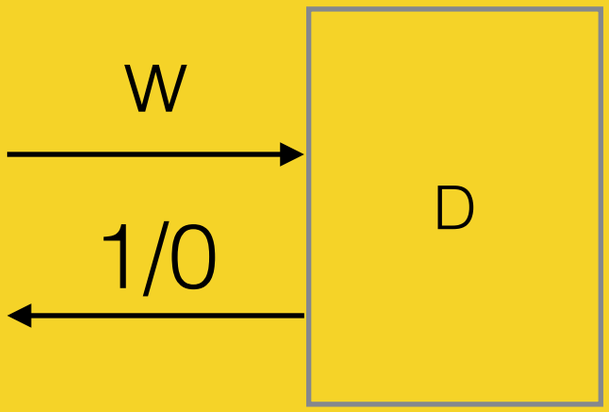
[KL, page 49]

“Negligible”

- Attacker’s winning probability should decrease very fast with security parameter n; much faster than user's slowdown (which is at most $\mathsf{poly}(n)$).
- Thus, a function $f : \mathbb{N} \to \mathbb{R}$ is *negligible* if it decays faster than any inverse polynomial function.
- Formally: f is negligible if for every polynomial $p(n)$, there is an N s.t. for all $n > N$, it holds that $f(n) < \frac{1}{p(n)}$.
- Examples: 2^{-n}, $2^{-\sqrt{n}}$, $n^{-\log n}$
- Convenient property: $\mathsf{negl}_1(n) \cdot p(n) = \mathsf{negl}_2(n)$. I.e., can tolerate polynomial blow-up.

IND-EAV secure scheme

- An EVE-secure encryption scheme can be constructed from **Pseudorandom Generator (PRG)**
- A PRG G expands a short (e.g., 16 bytes) seed s to a long (e.g., GBs) random-looking string $G(s)$ [KL, Def 3.14]
  - Let n be G's seed length, and $\ell(n)$ be its output length
    - e.g., $n = 128, \ell(n) = 1024$



$P_1 := Pr[D(w) = 1]$ for $s \leftarrow^{\$} \{0,1\}^n, w = G(s)$

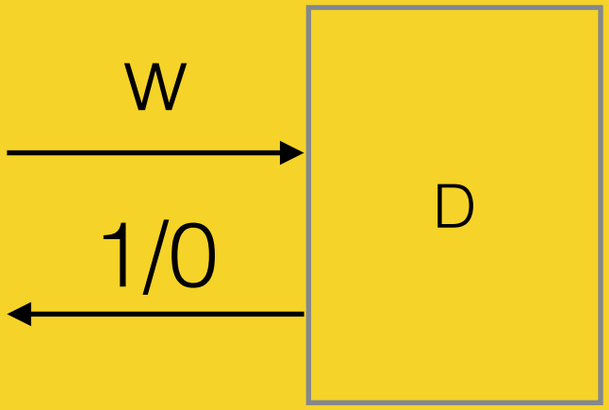
$P_2 := Pr[D(w) = 1]$ where $w \leftarrow^{\$} \{0,1\}^{\ell(n)}$

PRG security $\Rightarrow |P_1 - P_2|$ is neg for all PPT D

IND-EAV secure scheme

- Suppose G is a Pseudorandom Generator
- Consider this encryption scheme:
  - Gen: output $K \leftarrow^{\$} \{0,1\}^n$ (i.e., a random seed)
  - Enc(K, m): output $m \oplus G(K)$
  - Dec(K, m): output $c \oplus G(K)$
- Difference from OTP: a short key can encrypt a long message now.
- Claim: This scheme is EAV secure assuming G is a secure PRG.

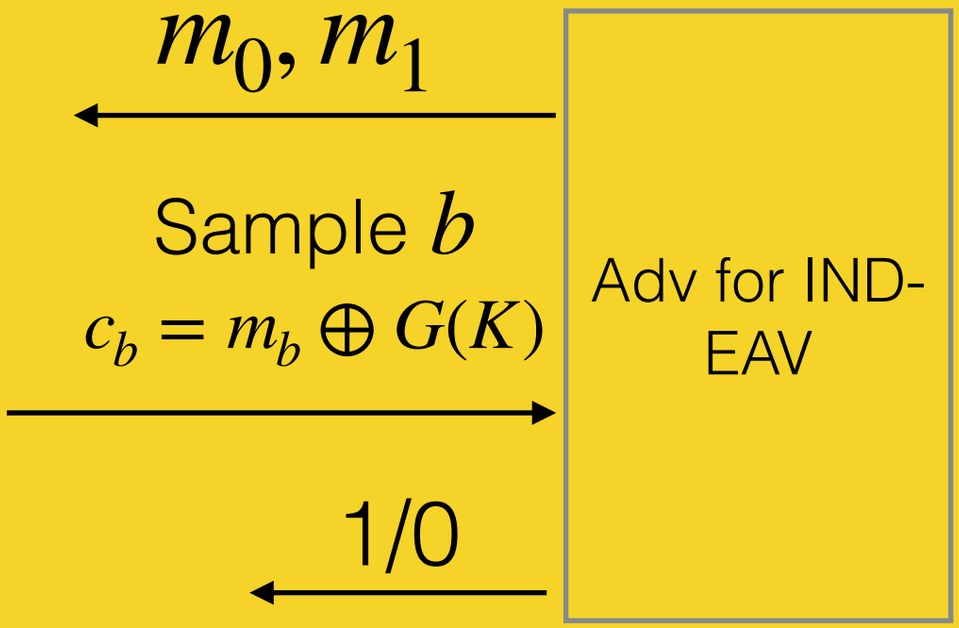
Proof of Reduction



$P_1 := Pr[D(w) = 1]$ for $s \leftarrow^{\$}, w = G(s)$

$P_2 := Pr[D(w) = 1]$ where $w \leftarrow^{\$}$

PRG security $\Rightarrow |P_1 - P_2|$ is neg



Proof by reduction: If there exists an adversary $\mathcal{A}$ who can break IND-EAV, it can be used to distinguish PRG G

$\Rightarrow$ such adversary $\mathcal{A}$ can't exists.

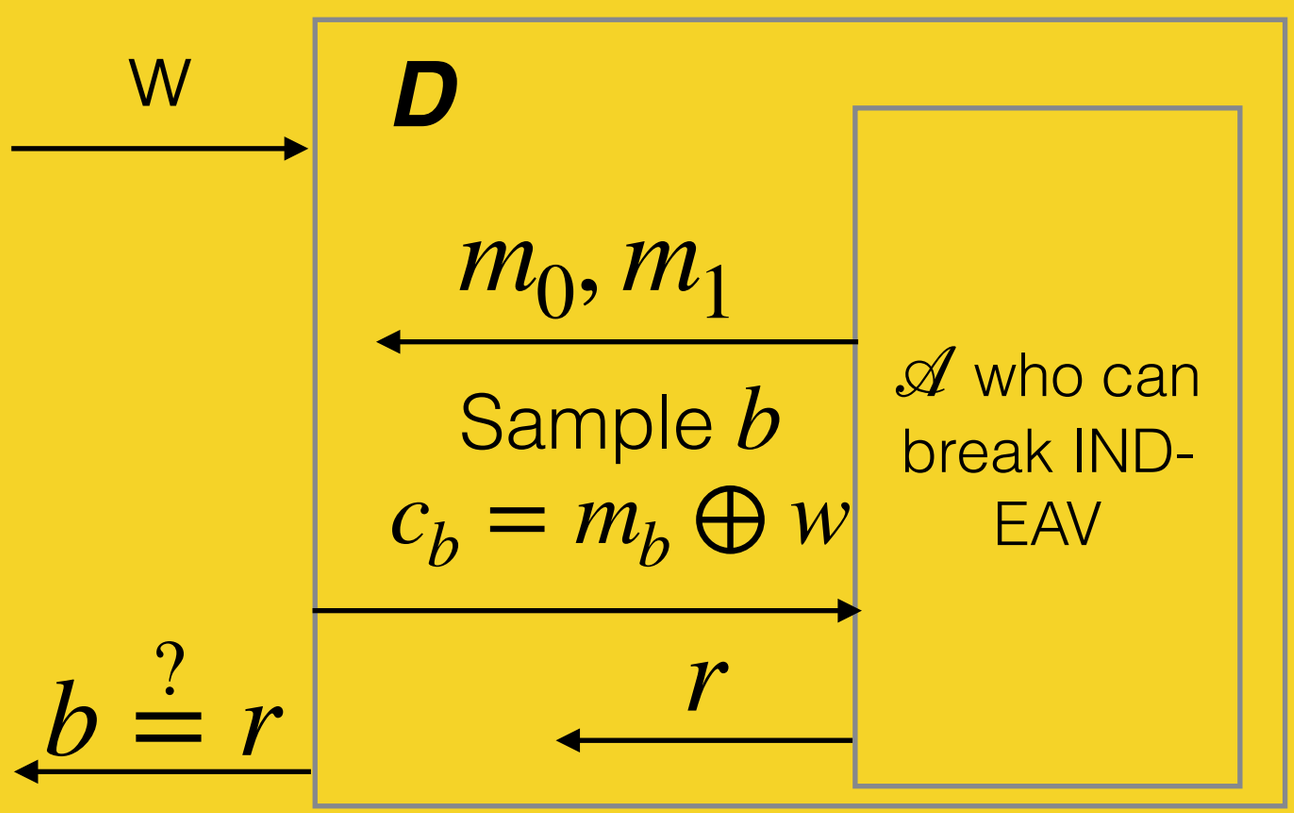
Proof of Reduction

$P_1 := Pr[D(w) = 1]$ for $s \leftarrow^{\$}, w = G(s)$

$P_2 := Pr[D(w) = 1]$ where $w \leftarrow^{\$}$

We claim: $|P_1 - P_2|$ is not neg for this ***D***

D outputs 1 iff $\mathscr{A}$ wins

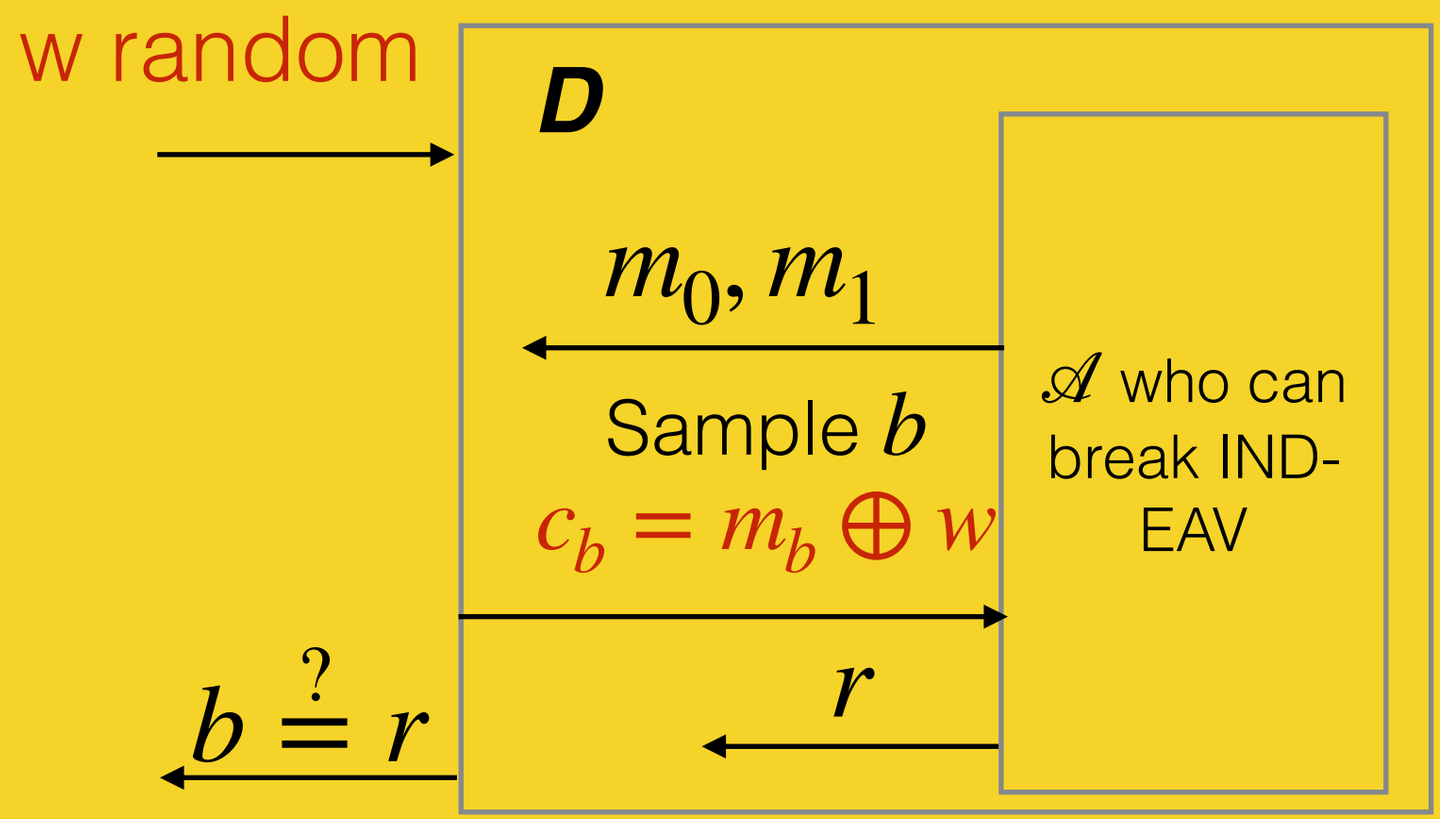


Proof of Reduction

$P_1 := Pr[D(w) = 1]$ for $s \leftarrow^{\$}, w = G(s)$

$P_2 := Pr[D(w) = 1]$ where $w \leftarrow^{\$}$

$= 1/2$ b/c of OTP security

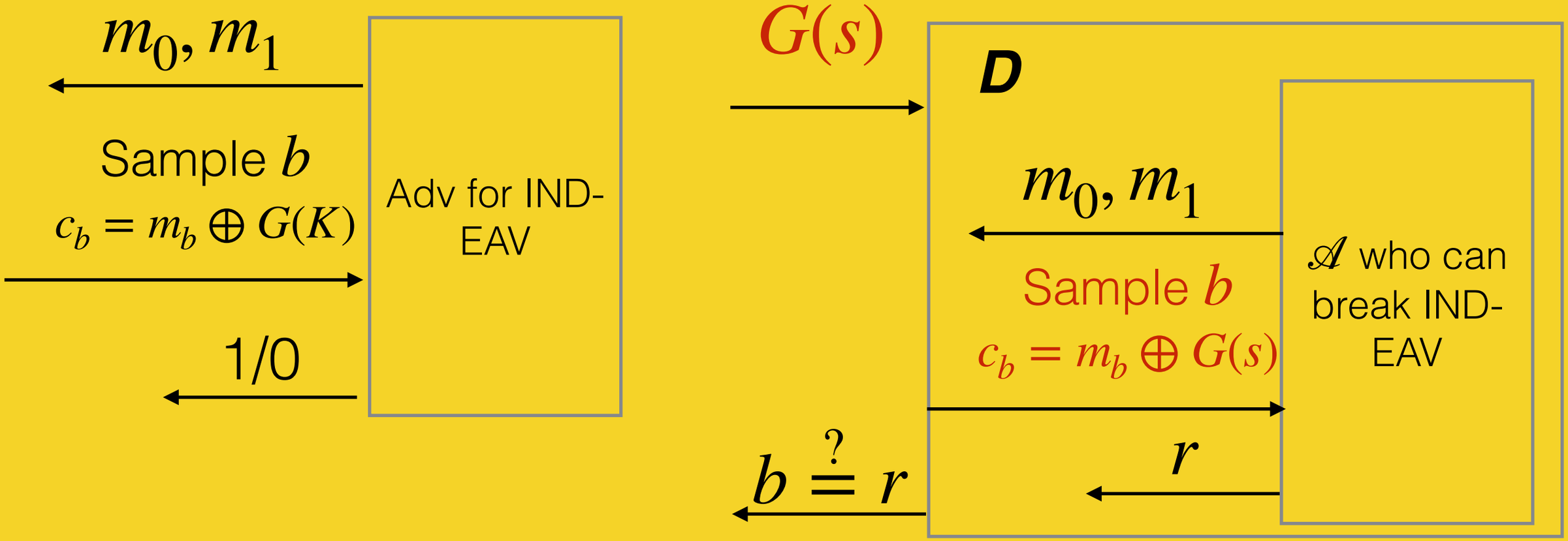


Proof of Reduction

$P_1 := Pr[D(w) = 1]$ for $s \leftarrow^{\$}, w = G(s)$

$\mathscr{A}$ breaks INA-EAV means $P_1 > 1/2 + \mathbf{negl(n)}$

Q.E.D.



Notes on proof of reduction

- Scheme is EAV-secure if G is a secure PRG
  - Not saying that PRG exists
  - Not say G is easier to build than encryption
  - Not saying that PRG and IND-EAV is equivalent
- Can use short key to encrypt long messages (not possible for OTP): power of computational security
- We won’t do a lot of proofs of reduction in class (which is the subject of intro to crypto courses)
  - Consult KL page 68 for details of the previous proof

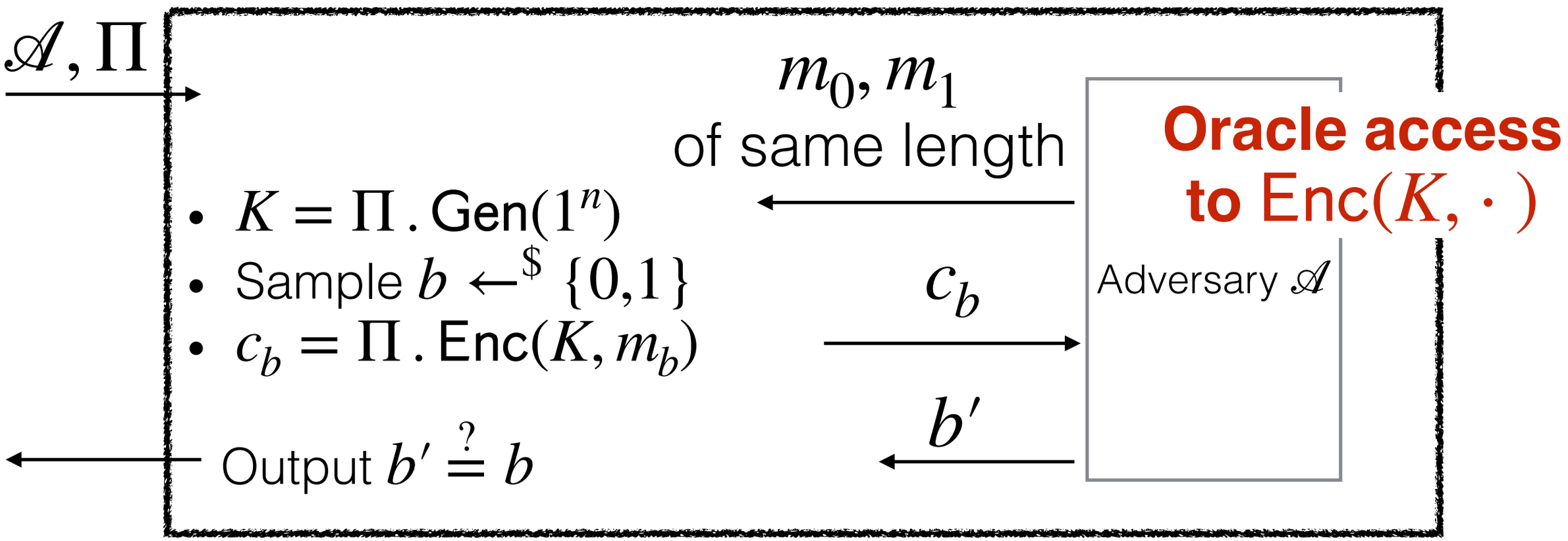
CPA Security

- Eavesdropping is passive (weak)
- A (stronger) attacker may trick Alice or Bob to encrypt messages of her choice
  - Chosen-Plaintext Attack (CPA)

Security games for CPA

Game $\mathsf{G}^{\text{IND-CPA}}(\mathscr{A}, \Pi)$

(indistinguishability against CPA attacker)



CPA security

- Why is CPA more powerful?
- A separating example:

Deterministic encryption schemes can’t be CPA secure!

Real-world scenario from KL

- May 1942, before the Battle of Midway Island, US Navy intercepted a Japanese ciphertext they were only able to partially decrypt: “Japan to attack XYZ”
  - XYZ was a ciphertext fragment that US can’t decrypt.
- They believed that XYZ encrypts “Midway Island”, but planners in DC can’t be convinced.
- US Forces at Midway sent a fake message “our fresh water supply is low”
- Later, another a Japanese message intercepted: “XYZ is low on water”
- Had Japanese used CPA-secure encryption, the history would look very different!

We have *extremely* practical CPA encryption from Block Ciphers

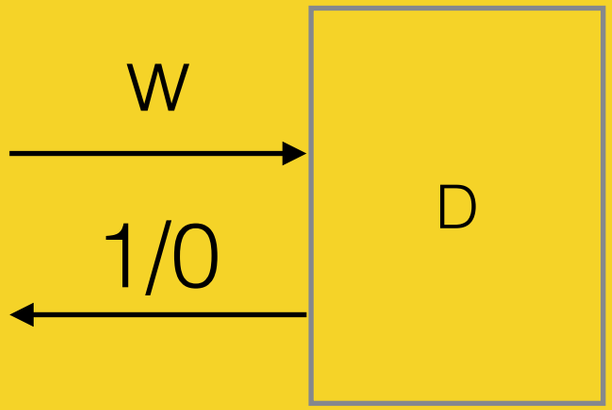
- Block cipher: E(K, m) and D(K, m)
  - Fixed length m, e.g., $m \in \{0,1\}^{128}$
  - Efficiently reversible: $\mathsf{D}(K, \mathsf{E}(K, m)) = m$
  - Given a random key K, function $\mathsf{E}(K, \cdot)$ is a (pseudo) random permutation over $\{0,1\}^{128}$
- Mathematical model: Pseudo-random Permutation (PRP).
  - Review slides in yellow for math definition of PRF and PRP and come to OH for questions.
- Primary example: AES and AES^{-1}

[skipped in class]

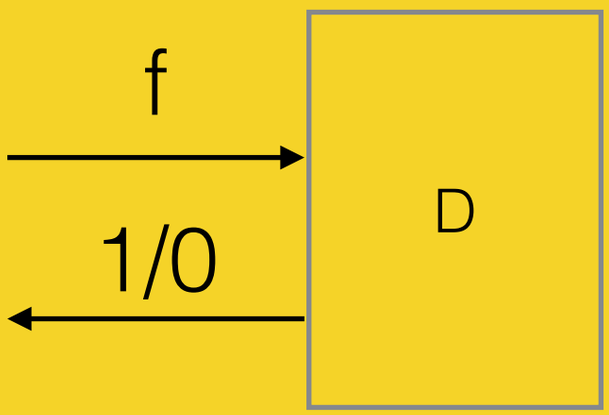
PseudoRandom Function (PRF)

- Let Func_n be the set of all mappings from n-bit inputs to n-bit outputs
  - Each function in the set is a look up table of 2^n rows
    - How many bits does it take to write down Func_n?
      - $(2^n)^{2^n}$ (Each function corresponds to a $2^n \cdot n$ bit string)
- A PRF is a family of function $\{F_1, F_2, \cdots, F_{2^n}\}$ such that for a uniform $k \in \{0,1\}^n$, $F_K(x)$ is indistinguishable from a random sample from Func_n
- This *should* be surprising, because many functions in Func_n will never be selected by K

PRF vs PRG



- Define two probabilities where D outputs 1 when
  - $s \leftarrow^{\$}, w = G(s)$
  - $w \leftarrow^{\$}$
- as P_1, P_2 respectively
- **PRG** is secure if $|P_2 - P1|$ is neg. for any PPT D



- Define two probabilities where D output 1 when
  - $f \leftarrow^{\$} \mathbf{Func}_n$
  - $f := F_k$ for $k \leftarrow^{\$} \{0,1\}^n$
- as P_1, P_2 respectively
- **PRF** is secure if $|P_1 - P_2|$ is neg. for any PPT D

PRG and PRF are closely related, and can build one from the other (not saying it's practically efficient to do so)

PRP

- PseudoRandom Permutation (PRP): exactly the same as PRF, but only considers permutation (i.e., bijections)
  - PRP is invertible so we can decrypt
- **Block cipher = practical PRP**

Security intuition of block ciphers (PRP)

- Let’s gain some intuition with n=2
- First, write down all permutations as lookup tables

00	00	00	00		11
01	01	01	11	…	10
10	10	11	01		01
11	11	10	10		00

- $2^n!$ of them in total

Security intuition of block ciphers (PRP)

- Second, assign **keys** to each permutation

	$k=0$	$k=1$	$k=2$	…
00	00	00	00	11
01	01	01	11	10
10	10	11	01	01
11	11	10	10	00

Ideal Cipher Model

	$k=0$	$k=1$	$k=2$	....	
00	00	00	00		11
01	01	01	11	...	10
10	10	11	01		01
11	11	10	10		00

- Someone wrote down these tables on the sky
- Ideal cipher $\mathsf{E}(k, \cdot)$ uses the k-th table
- **Security intuition**: not knowing K, every table is possible.

Power of cryptography

Random Permutation

	$k=2$	$k=10$	$k=1$	…	$k=17$
00	00	00	00		11
01	01	01	11	…	10
10	10	11	01		01
11	11	10	10		00

≈

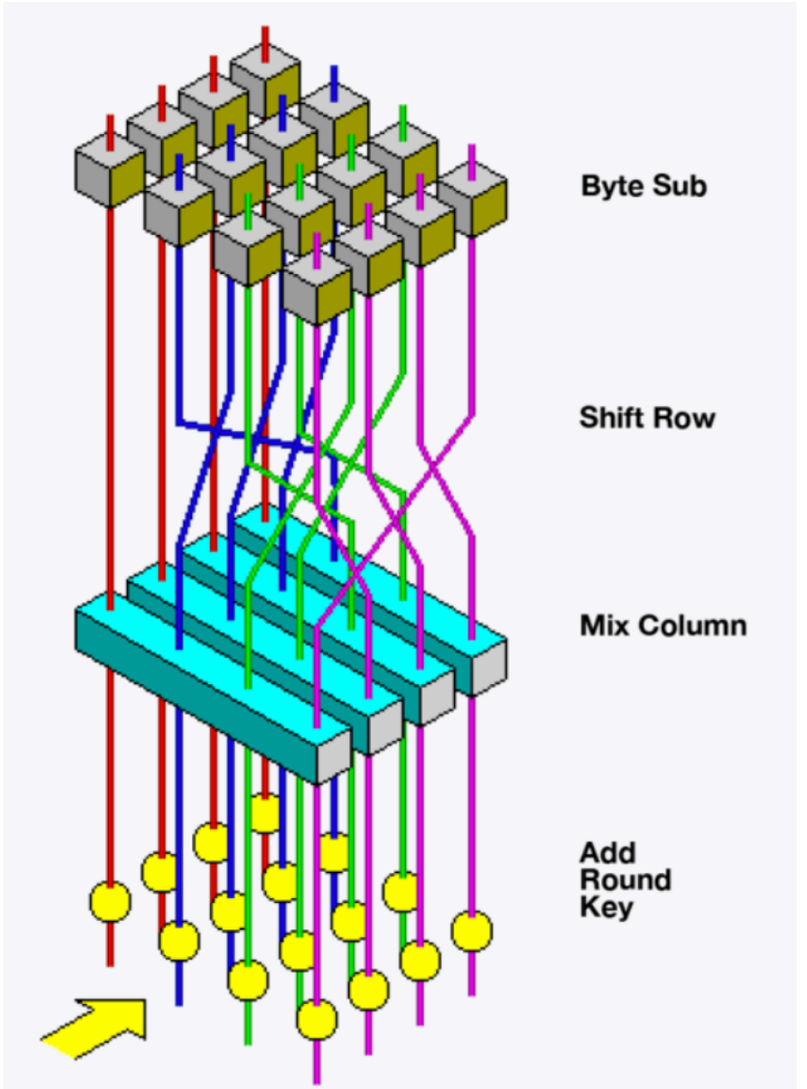
```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])

   for round = 1 step 1 to Nr-1
      SubBytes(state)
      ShiftRows(state)
      MixColumns(state)
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Huge table (how big?) can't fit in the Universe.

Hundreds lines of code (AES)

Advanced Encryption Standard

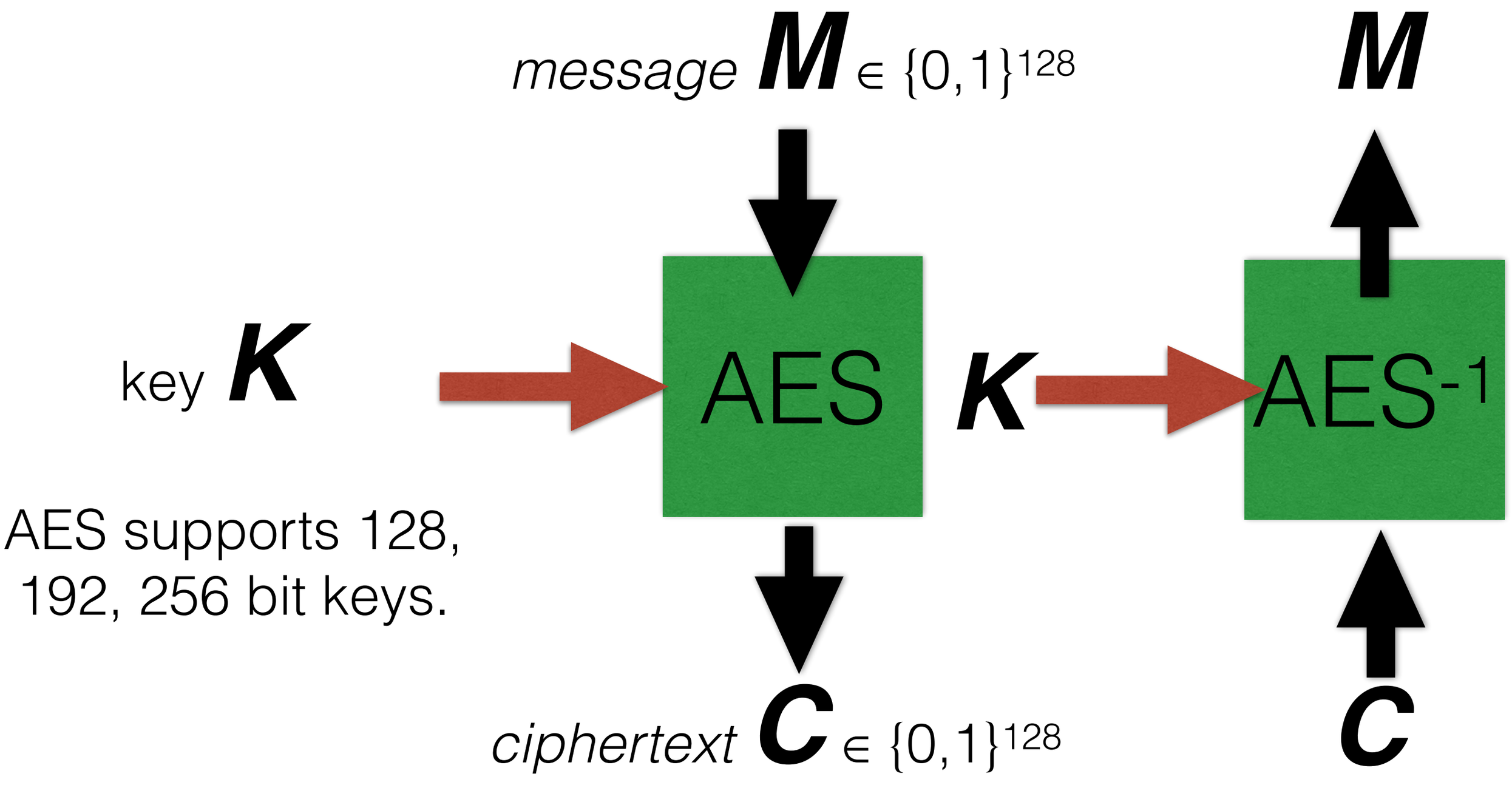
- Winner of the U.S. National Institute of Standards and Technology (NIST) competition for AES in 1997
  - Designed by Joan Daemen and Vincent Rijmen
- Standardized in FIPS-197 and is the main cipher used today.
- *Really* Fast: a few CPU cycles per byte with modern CPUs.
- Read Wikipedia if you are interested in its internal working.



Visualization of the AES round function

AES

For a **uniform** key, $\mathsf{AES}(K, \cdot)$ is a PRP for 128-bit strings.



AES supports 128, 192, 256 bit keys.

From AES to CPA-secure encryption

Quiz

- Is $\mathsf{Enc}(K, m) := \mathsf{AES}(K, m)$ a CPA-secure encryption scheme?

- No, because it’s deterministic

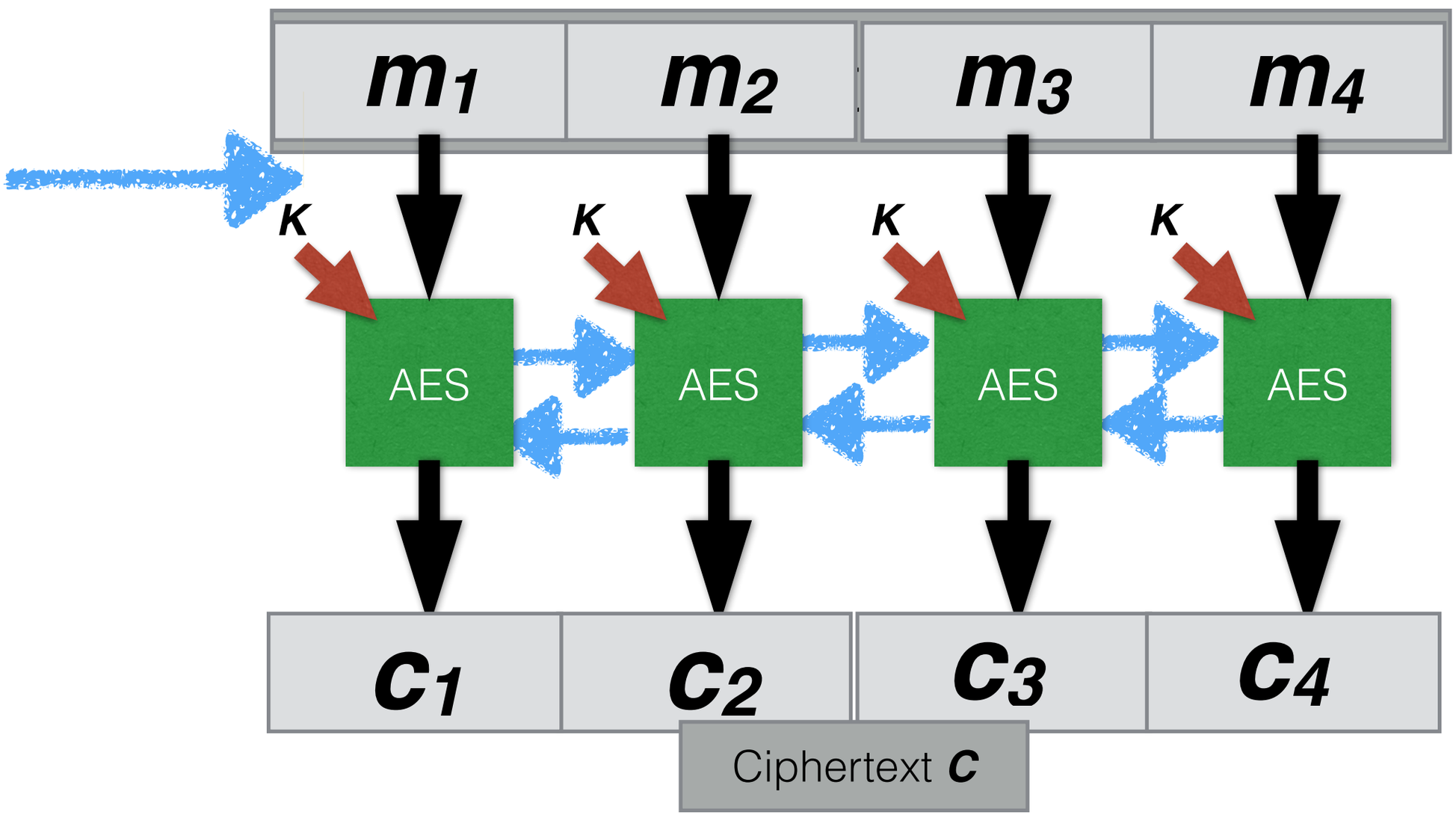
From AES to CPA-secure encryption

- A CPA-secure scheme for 128-bit messages
  - Sample $r \leftarrow^{\$} \{0,1\}^{128}$
  - Output $(r, \underbrace{\mathsf{AES}(K, r)}_{\text{One-time pad}} \oplus m)$

- What about longer inputs?

Mode of operation

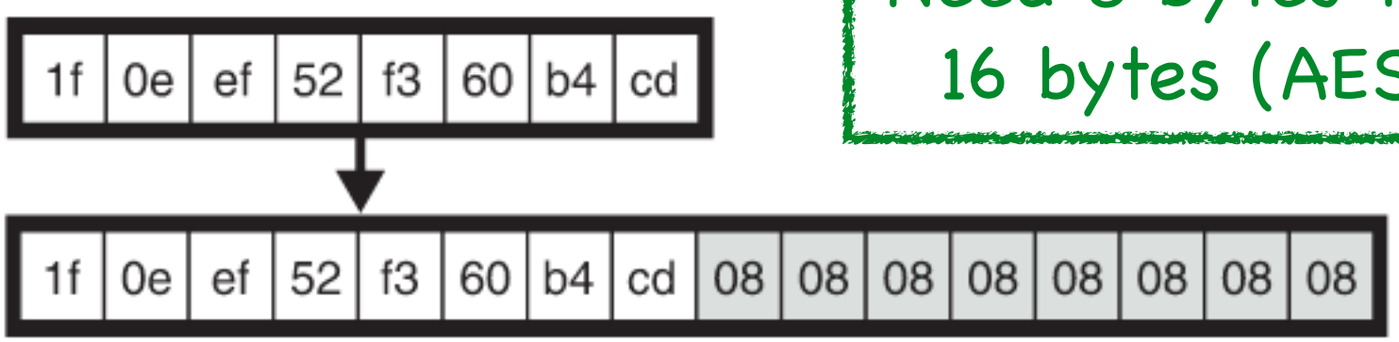
Split to 128 bits and padding if needed.



Various possible additions / interconnections:

Chopping & Padding

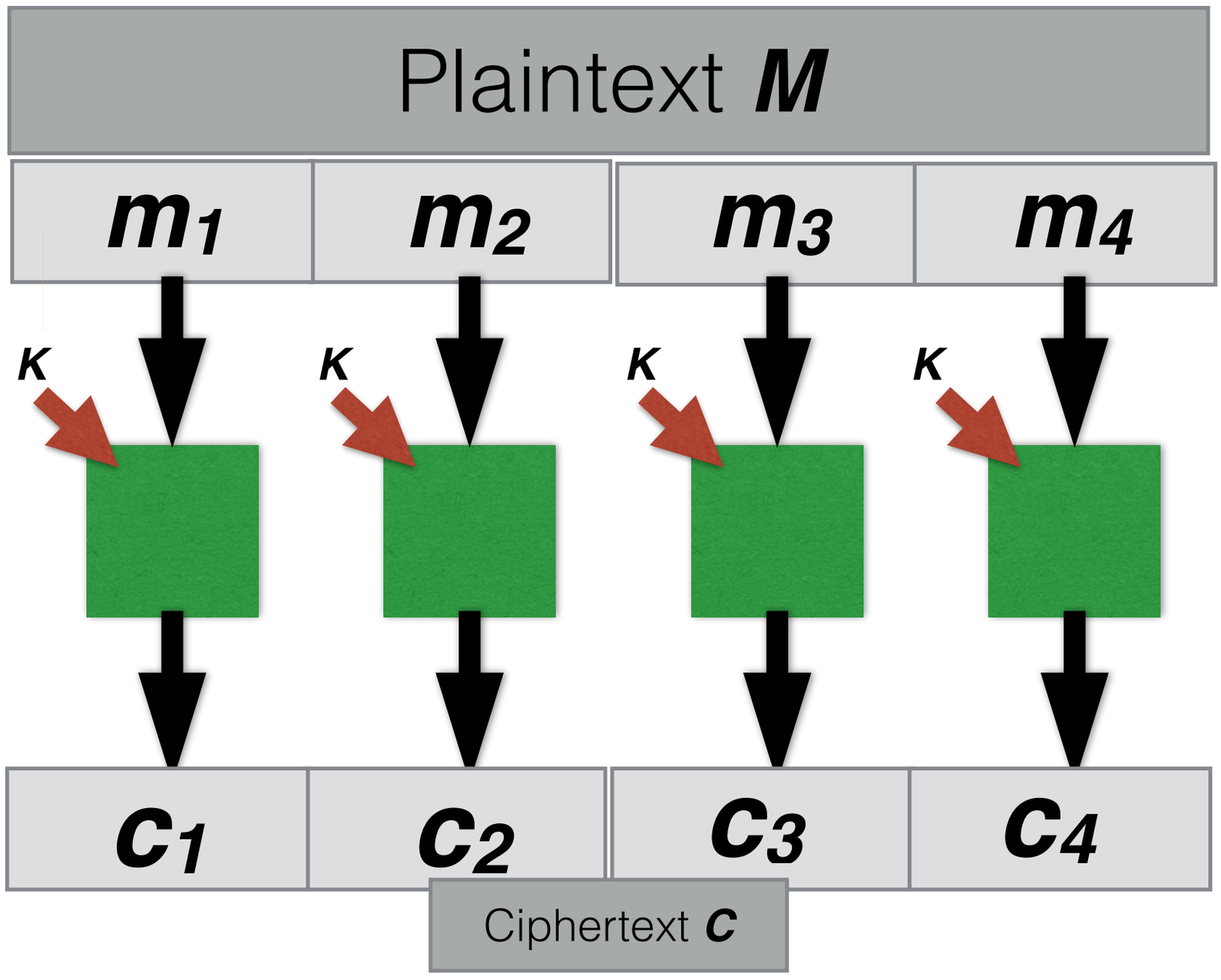
- First step is to chop inputs into 128-bit blocks (for AES)
- The recipient needs to know how to remove padding
  - Adding random bytes won't work
- Most popular scheme PKCS#7: pad using the length of required padding
- Will come back to this in a few lectures.



Need 8 bytes to pad to 16 bytes (AES input)

Receiver reads last byte (say n), removes last n bytes.

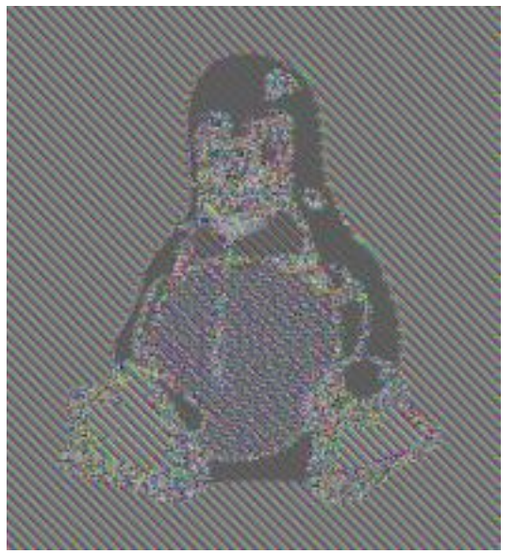
Electronic Code Book (ECB) mode



Identical message blocks ➜ identical ciphertext blocks!

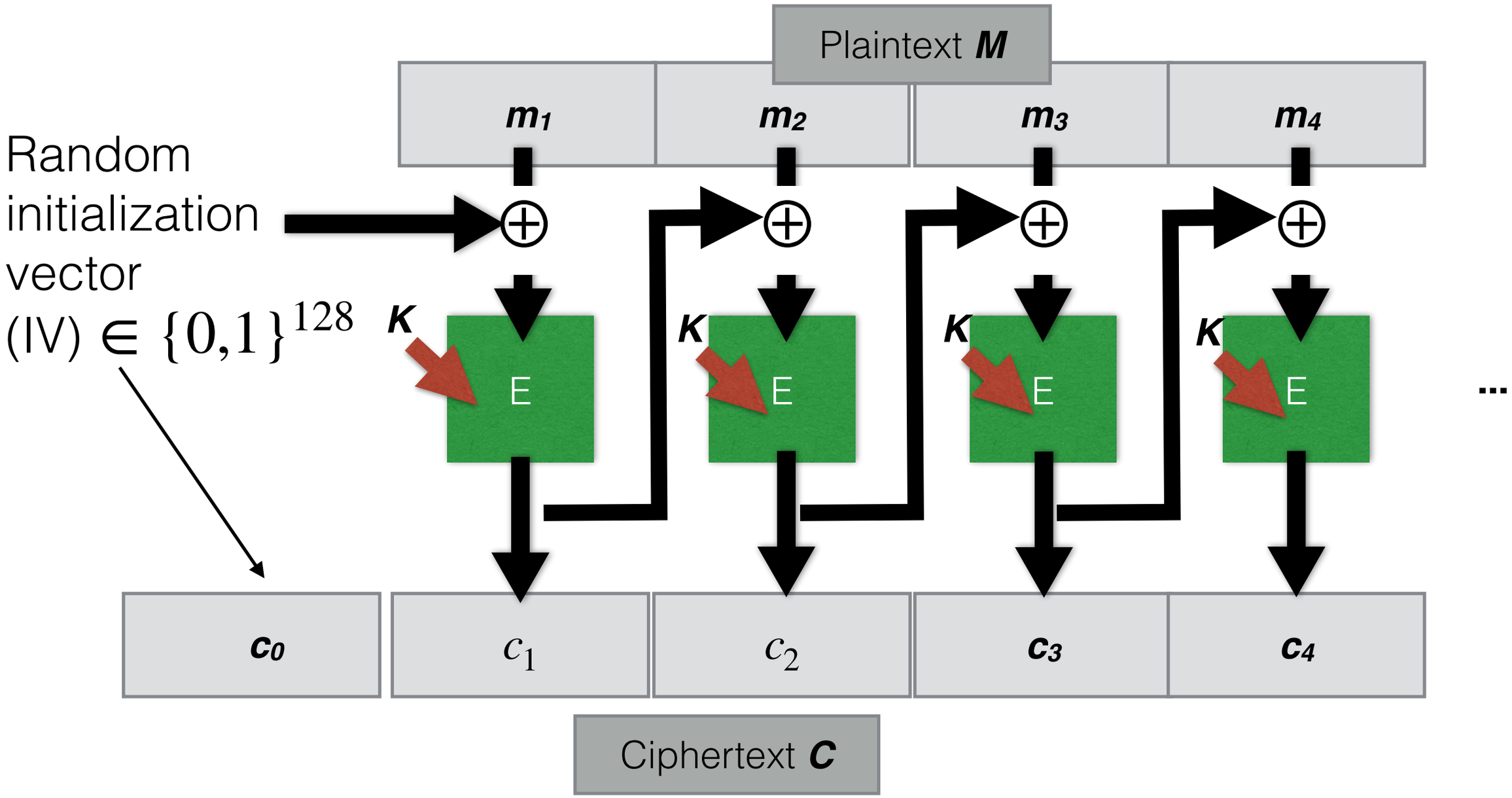
ECB leaks information

ECB encryption



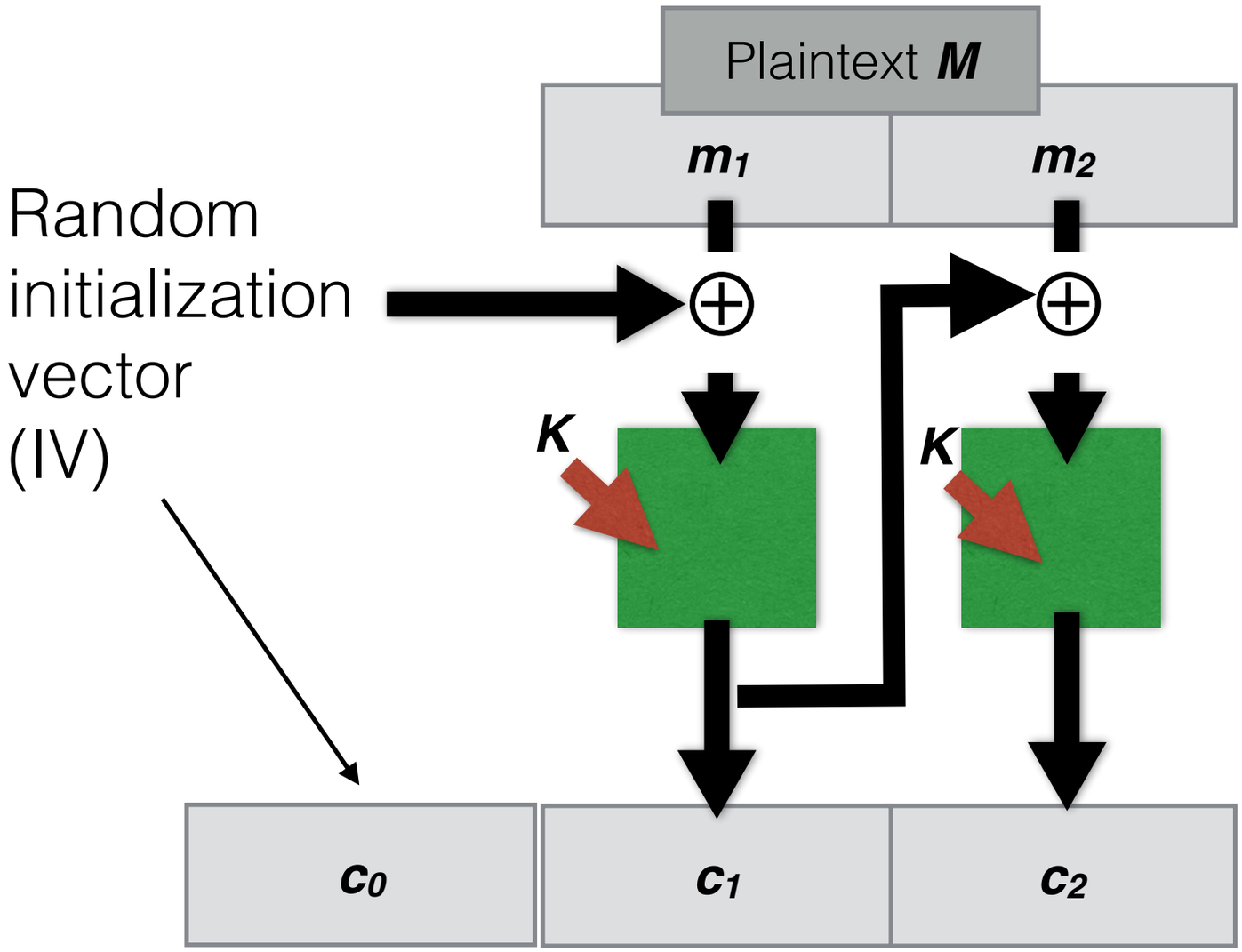
Never use ECB mode!
Not CPA secure.

Cipher-Block Chaining (CBC) mode



IV must be unique and unpredictable. E.g., Incremental counters don't work. See PS.

Decryption in CBC mode



- $c_0 = IV$
- $c_{i+1} = ?$

$$c_{i+1} = \mathrm{AES}(K, c_i \oplus m_{i+1})$$

Decryption?

$$m_{i+1} = \mathrm{AES}^{-1}(K, c_{i+1}) \oplus c_i$$

Observe: Flipping a bit in the ciphertext flips a bit in the decryption result. Culprit of Padding Oracle Attacks.

Counter (CTR) mode

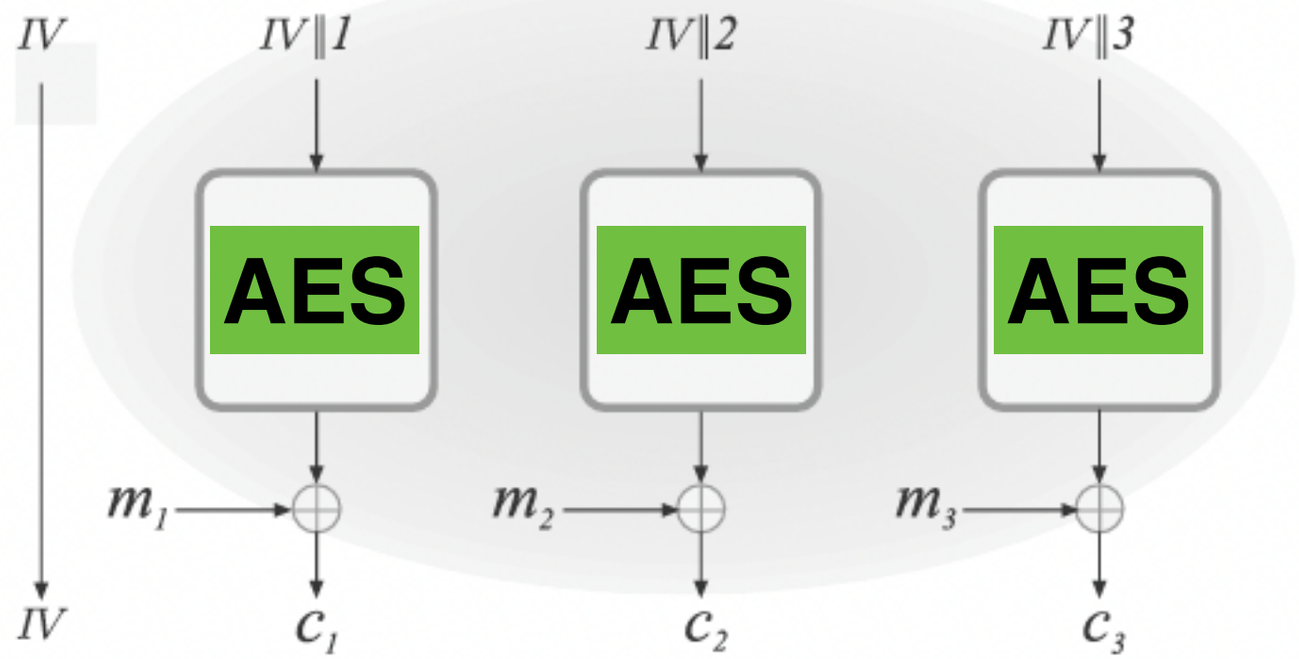


FIGURE 3.9: Counter (CTR) mode.

- AES-CTR uses AES to encrypt a nonce concatenated with a number (starting at 1) to produce a **key stream**
- Cipher text is the XOR between msg and key stream
- IV must be **unique** (why?)

Figure from *Introduction to modern cryptography*

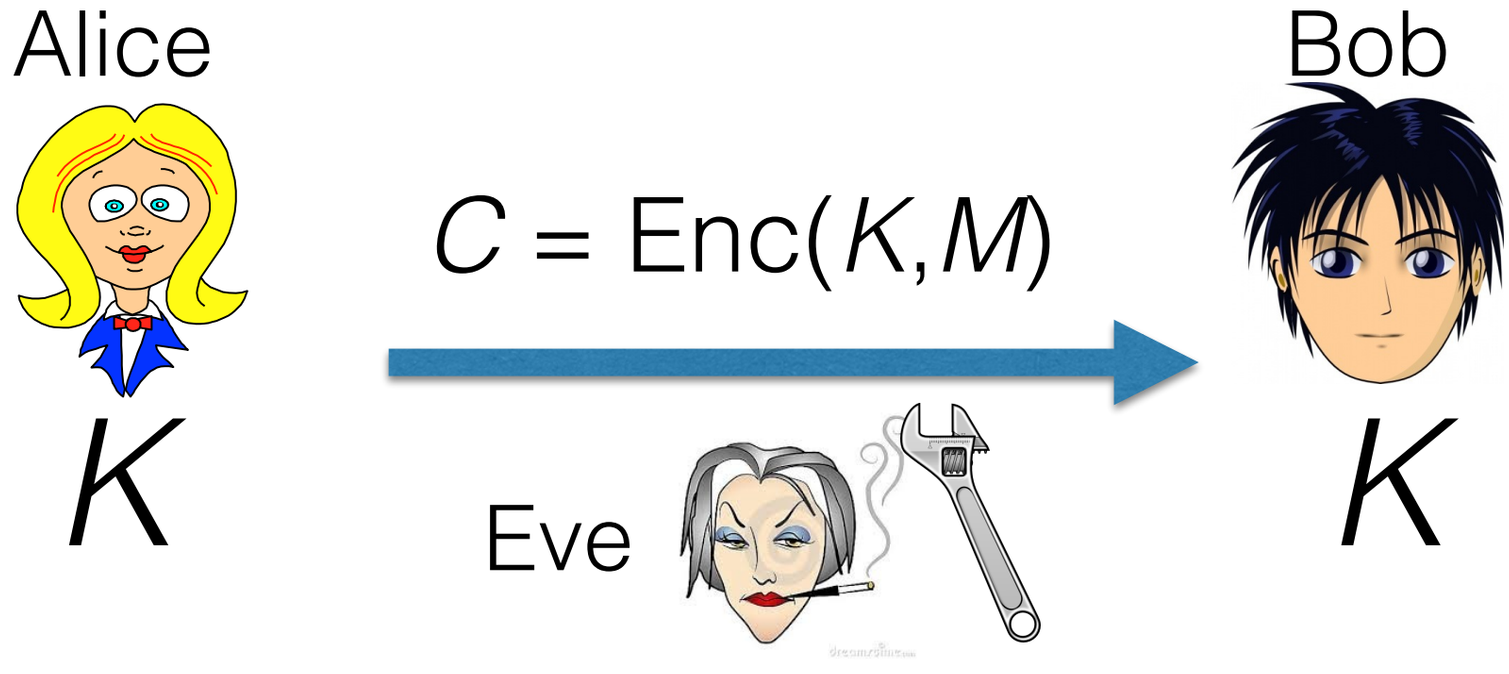
CTR mode vs CBC

- Both are commonly used
  - TLS 1.2 supports both
- Counter mode has some advantages
  - Can be parallelized (much faster)
  - Padding is not necessary, thus shorter ciphertext
  - Only uses AES (CBC uses AES and AES^{-1}) — matters for devices with constrained resources

Integrity

Encryption alone is useless if message integrity is not protected

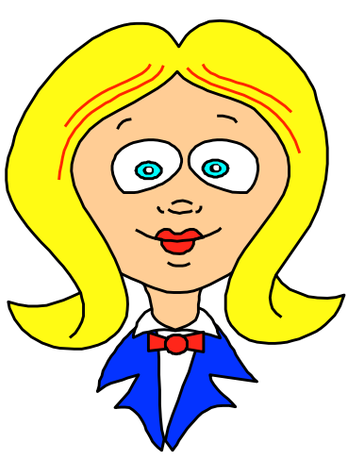
- Observe: AES-CBC or CTR will happy decrypt any garbage you throw at it.
- **Integrity** protects against message modification/ manipulation
- Always use encryption with integrity protection.
  - Message Authentication Code (symmetric-key), or
  - Digital signature (public-key)



Syntax of Message-Authentication Code (MAC)

$K \leftarrow^{\$} \mathsf{Gen}(1^n)$

Alice



K

(x, T)

Bob

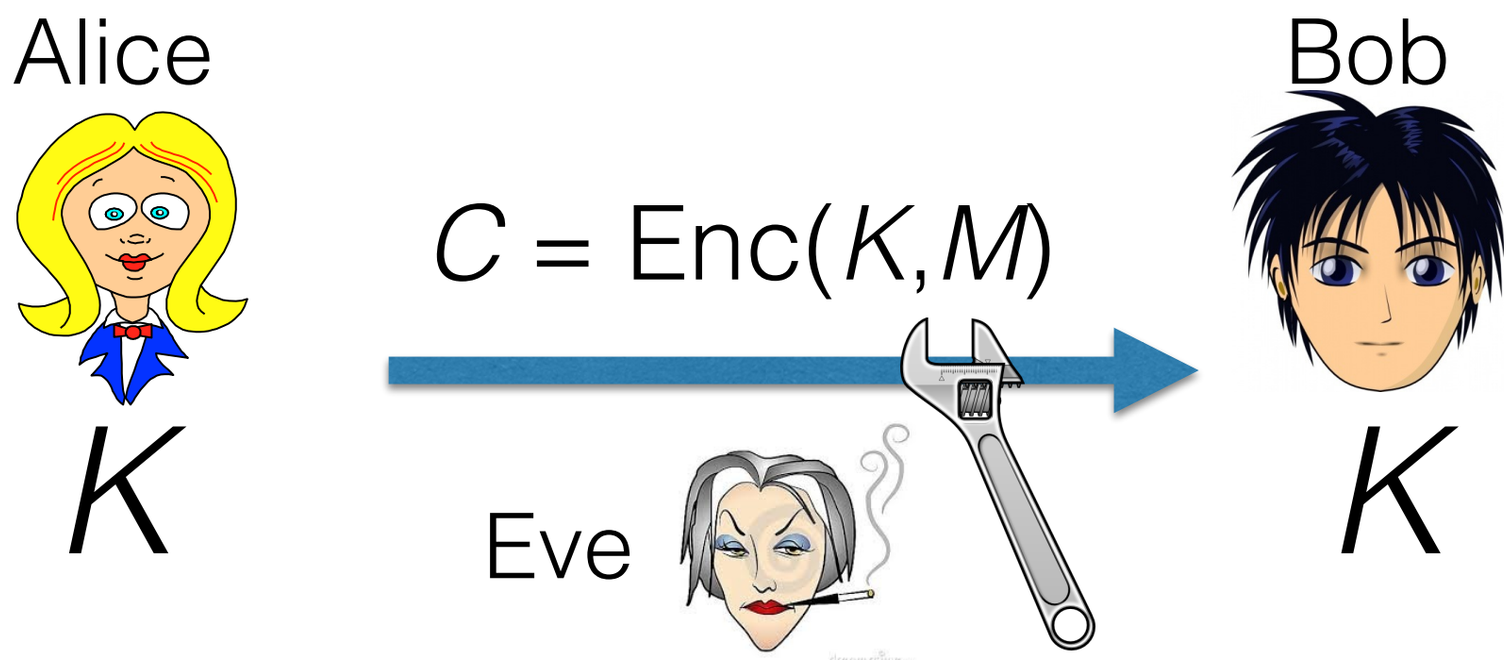
K

$T = \mathsf{Mac}(K, x)$

$T \stackrel{?}{=} \mathsf{Mac}(K, x)$

Security goals
[Our first *anti-forgery* goal]

- Adversary (Eve) can't **forge tags** that pass verification even if Eve can
  - eavesdrop & obtain many tags
  - generate tags on her own messages



Game based definition

The message authentication experiment $\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n)$**:**

1. *A key* k *is generated by running* $\mathsf{Gen}(1^n)$.
2. *The adversary* $\mathcal{A}$ *is given input* 1^n *and oracle access to* $\mathsf{Mac}_k(\cdot)$. *The adversary eventually outputs* (m, t). *Let* $\mathcal{Q}$ *denote the set of all queries that* $\mathcal{A}$ *submitted to its oracle.*
3. $\mathcal{A}$ succeeds *if and only if (1)* $\mathsf{Vrfy}_k(m, t) = 1$ *and (2)* $m \notin \mathcal{Q}$. *In that case the output of the experiment is defined to be* 1.

In English: after seeing many valid tags on messages of their choice, the attacker cannot produce a valid tag for a *new* message.

Denoted as $\mathsf{UF}_{\mathcal{A},\Pi}(n)$ for short in later slides.

MAC from PRF

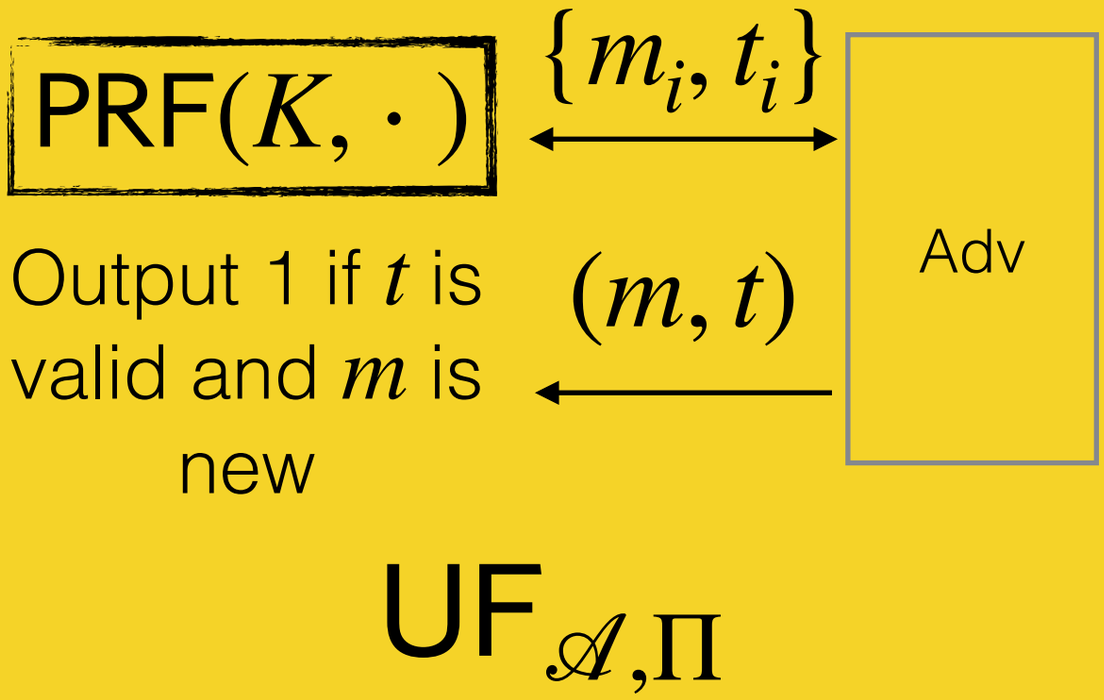
- PRF (e.g., AES) is also MAC:
  - We assume $m \in \{0,1\}^n$ // PRF input length
  - Gen: output a random n-bit string K
  - Mac(K, m) = PRF(K, m)
- Intuition:
  - for a new message m', guessing the tag PRF(K, m') right is negligible 2^{-n}

Proof by reduction

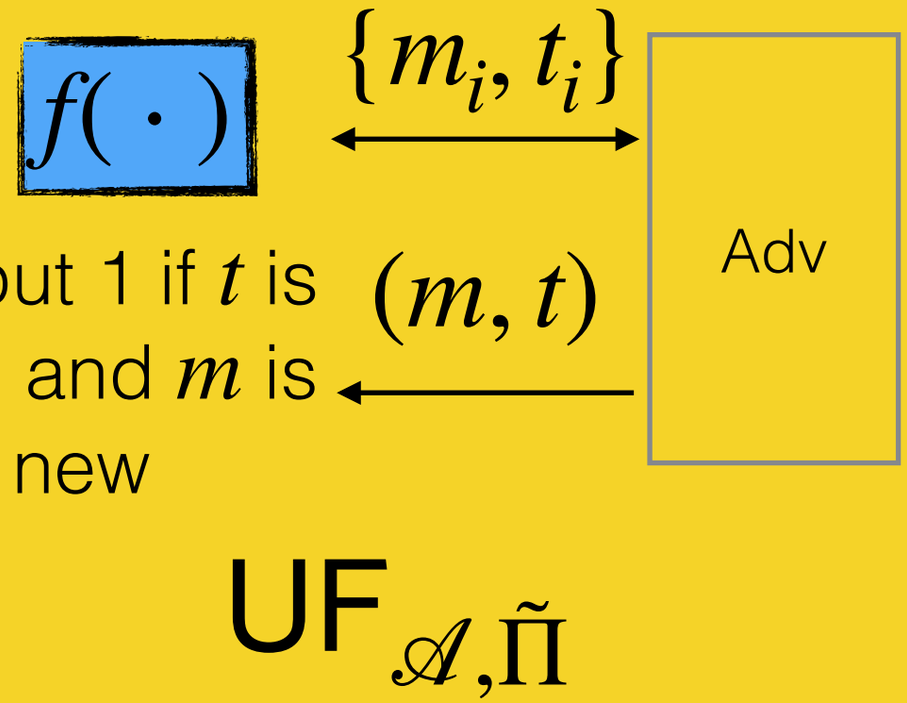
- Consider a modified scheme $\tilde{\Pi}$
  - Gen: outputs a random function f
  - Mac(f, m): outputs $f(m)$
- Clear that for all PPT adversary $\mathcal{A}$,
  $Pr[\text{UF}_{\mathcal{A},\tilde{\Pi}}(n) = 1] = 2^{-n}$
  - i.e., the best thing to do is guess
- Next, we want to show that $\text{UF}_{\mathcal{A},\Pi}$ is almost as hard as $\text{UF}_{\mathcal{A},\tilde{\Pi}}$ for all PPT adversary.

WTS: for all PPT $\mathcal{A}$, the probability difference

$|Pr[\text{UF}_{\mathcal{A},\Pi} = 1] - Pr[\text{UF}_{\mathcal{A},\tilde{\Pi}} = 1]|$ must be negligible.



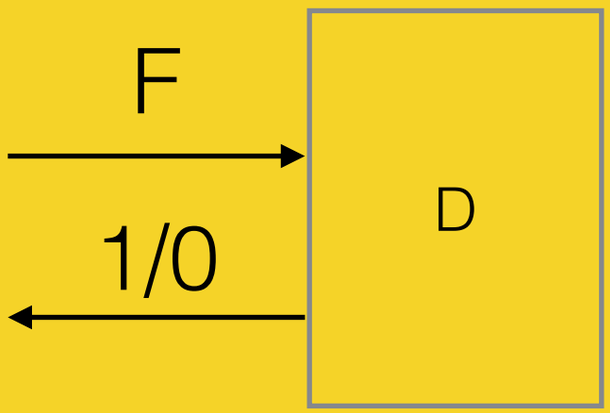
$\text{UF}_{\mathcal{A},\Pi}$



$\text{UF}_{\mathcal{A},\tilde{\Pi}}$

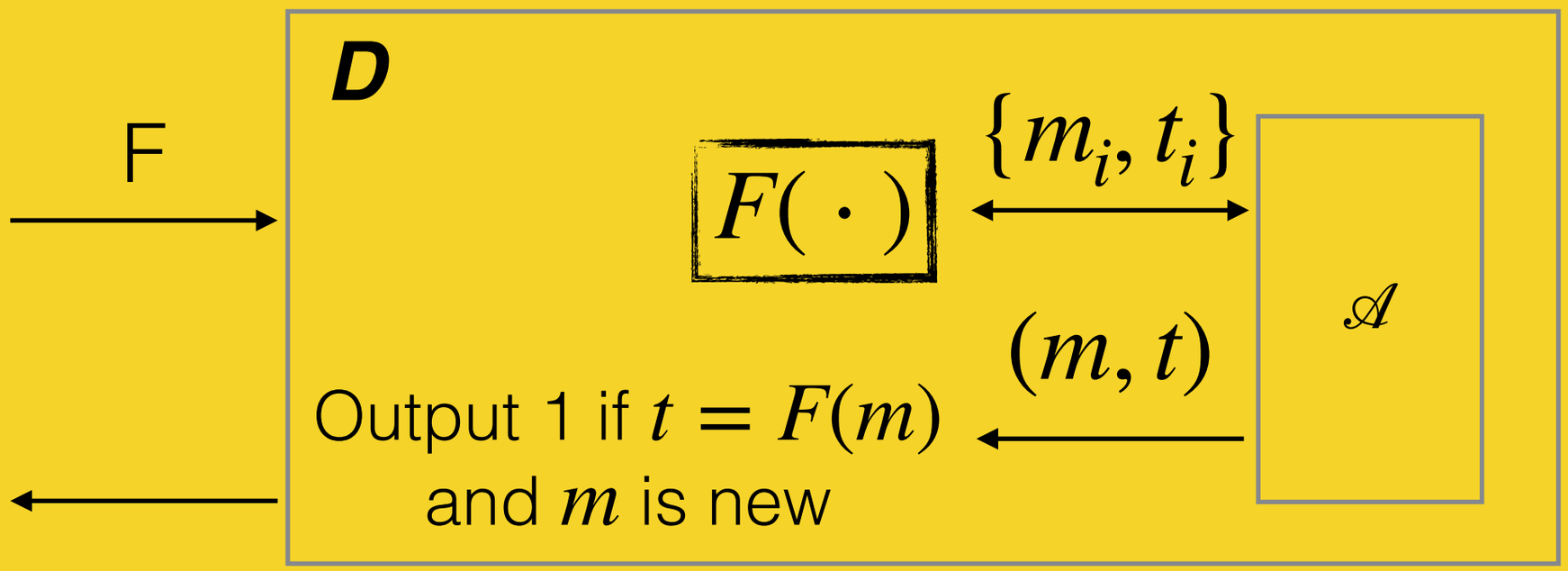
If we can prove this, then we are done.

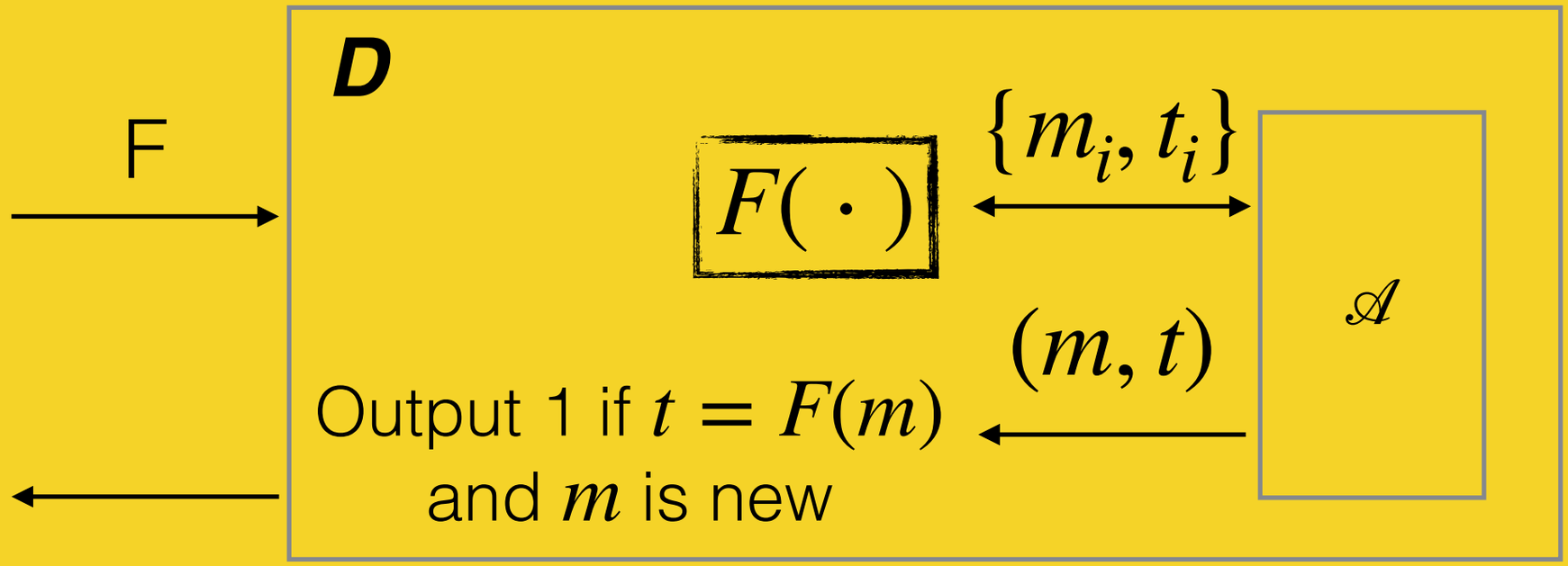
Recall PRF security definition



- Given D, let P_1 be the probability where D outputs 1 when given $F \leftarrow^{\$} \textbf{Func}_n$
- Let P_2 be the probability where D outputs 1 when given $F := F_k$ for $k \leftarrow^{\$} \{0,1\}^n$)
- PRF is secure if $|P_1 - P_2|$ is neg. **for all PPT D**

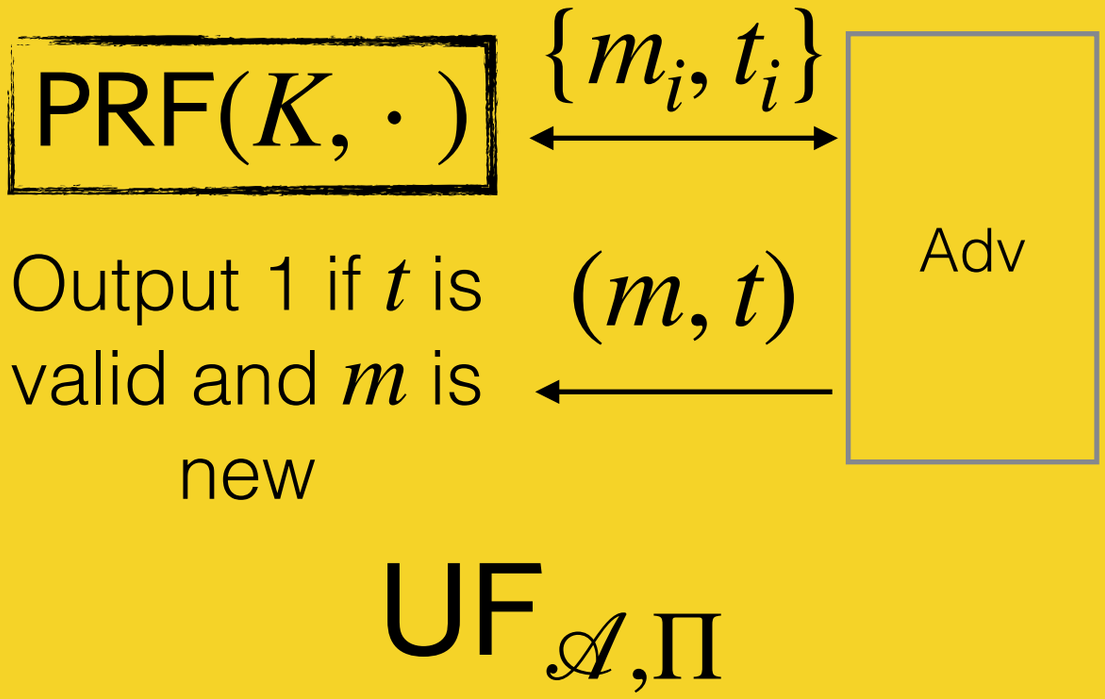
Consider this ***D:***



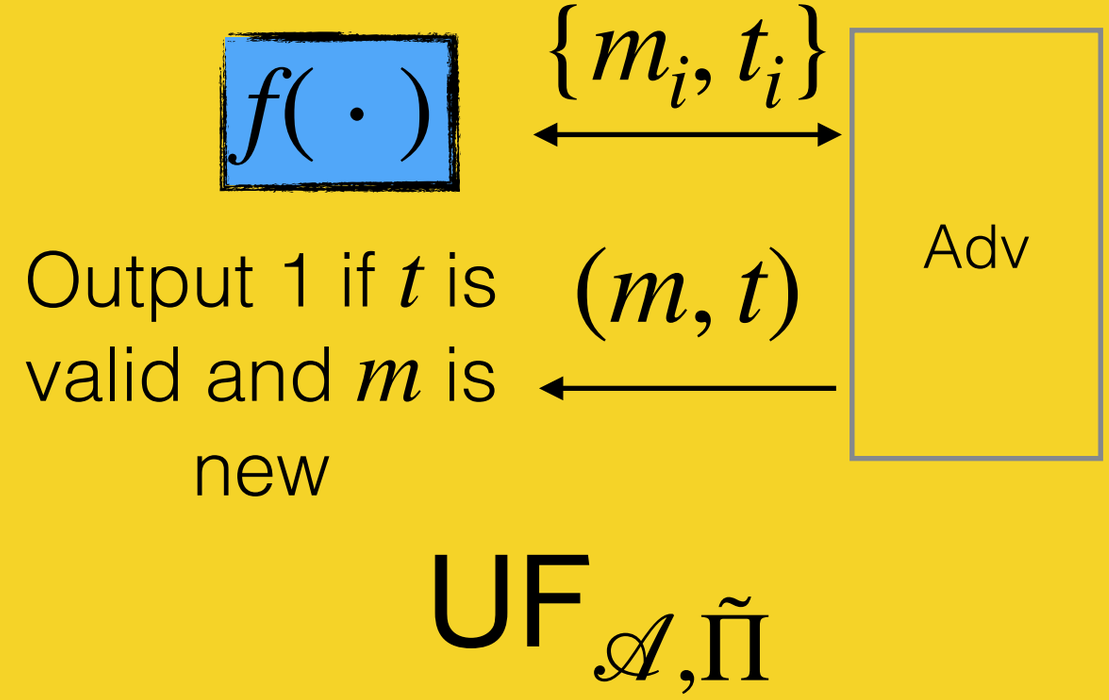


$P_2 = Pr[\mathsf{UF}_{\mathcal{A},\Pi} = 1]$

$P_1 = Pr[\mathsf{UF}_{\mathcal{A},\tilde{\Pi}} = 1]$



$\mathsf{UF}_{\mathcal{A},\Pi}$



$\mathsf{UF}_{\mathcal{A},\tilde{\Pi}}$

Proof by reduction

- Conclusion: assuming PRF security,
  $Pr[\mathsf{UF}_{\mathcal{A},\Pi}(n) = 1] \leq 2^{-n} + \mathsf{neg}(n)$
- I.e., Π is secure MAC (for n-bit inputs)

How to go longer?

m 

How about

$t = (\mathsf{Mac}(K, m_1), \mathsf{Mac}(K, m_2), \mathsf{Mac}(K, m_3))$?

How about

$t = (\mathsf{Mac}(K, 1 \| m_1), \mathsf{Mac}(K, 2 \| m_2), \mathsf{Mac}(K, 3 \| m_3))$

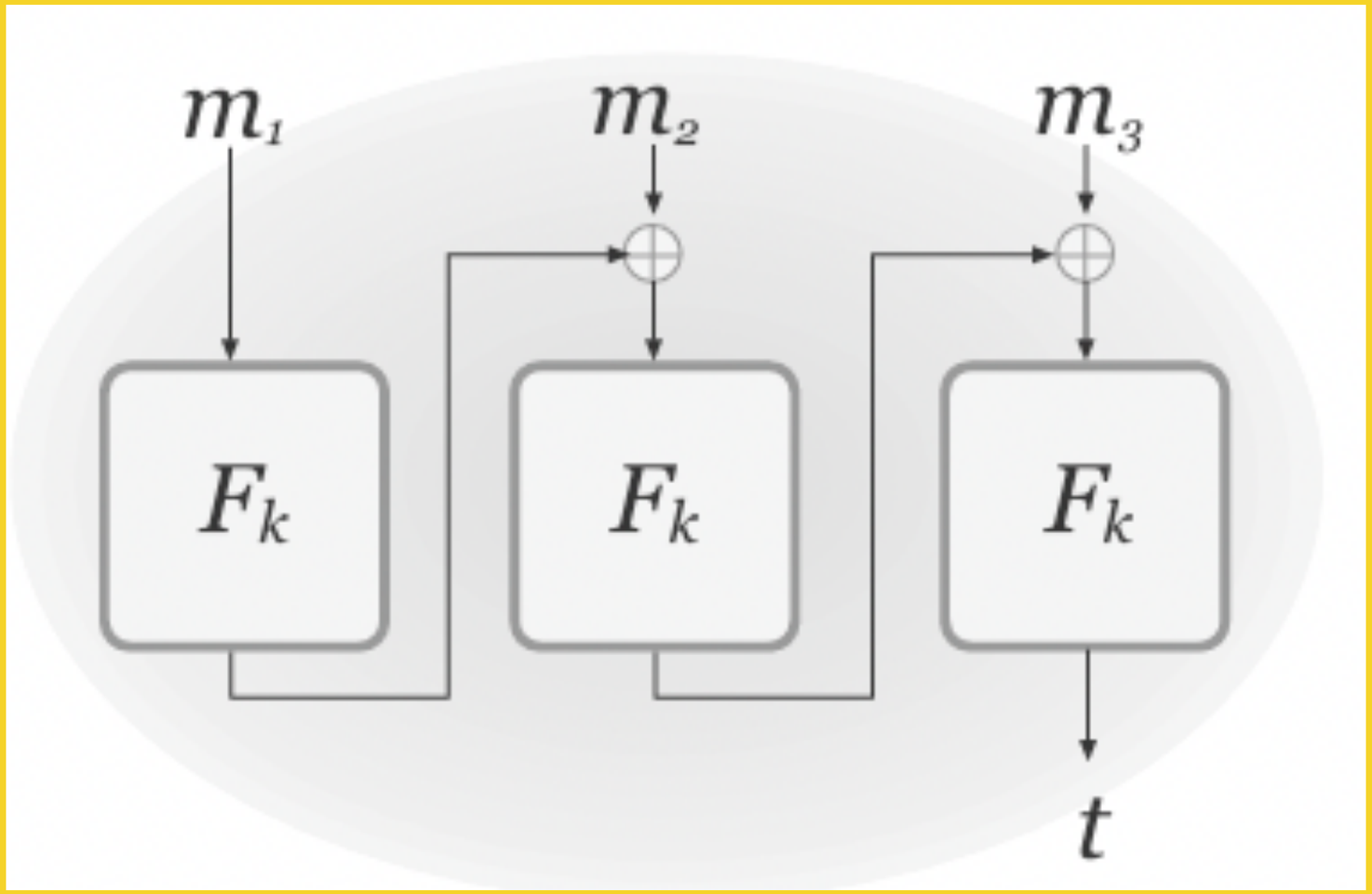
How about

$t = (\mathsf{Mac}(K, 3 \| 1 \| m_1), \mathsf{Mac}(K, 3 \| 2 \| m_2), \mathsf{Mac}(K, 3 \| 3 \| m_3))$?

See [KL, Construction 4.7] for a working construction.

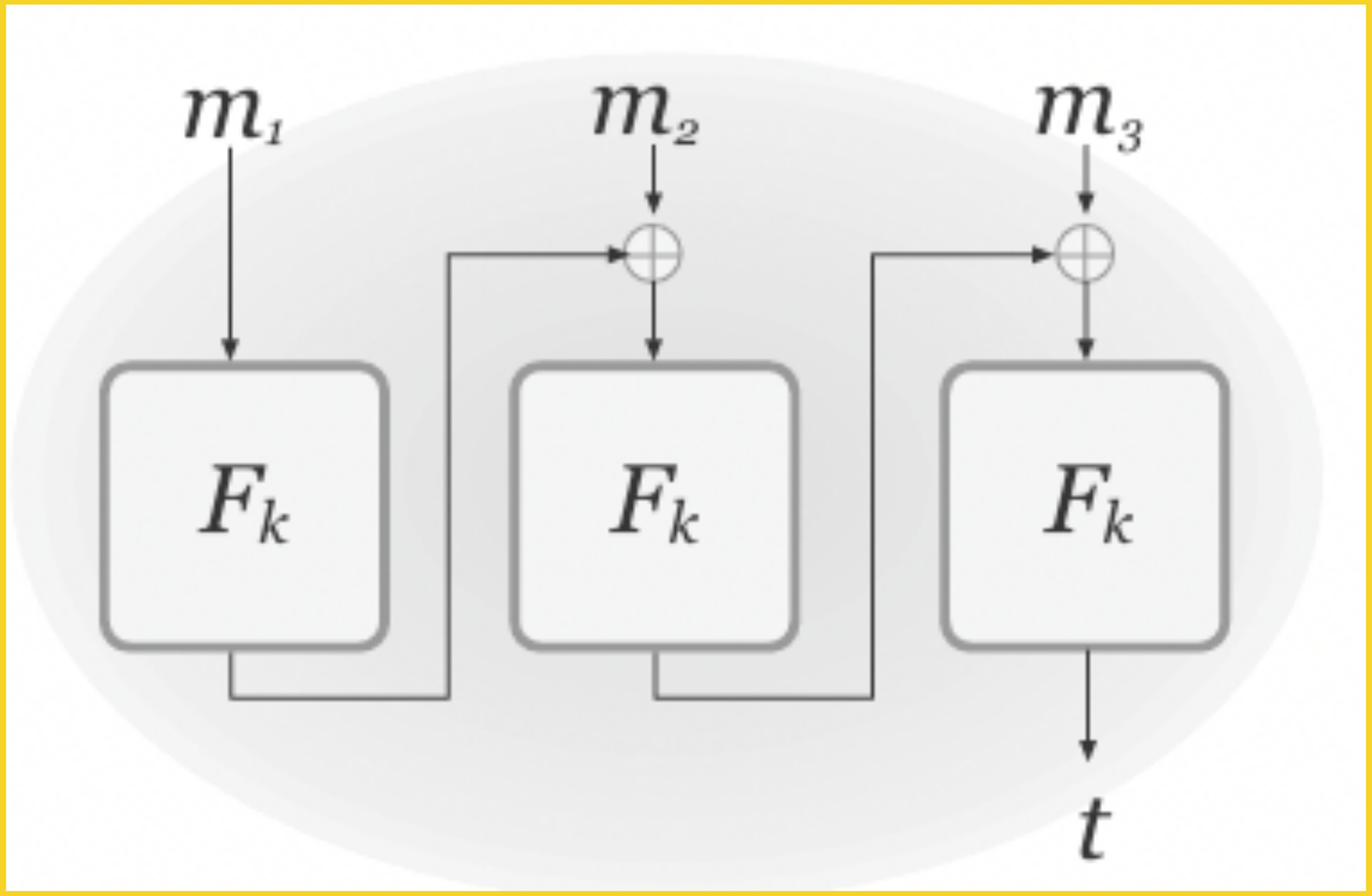
Domain extension for PRF: CBC-MAC

- Chain PRF together just like CBC mode
- Used as part of the CCM mode (see Wikipedia if you are curious)
- **Caution: This construction is only secure if message length is fixed (known by sender and receiver a priori)**



Extension Attack

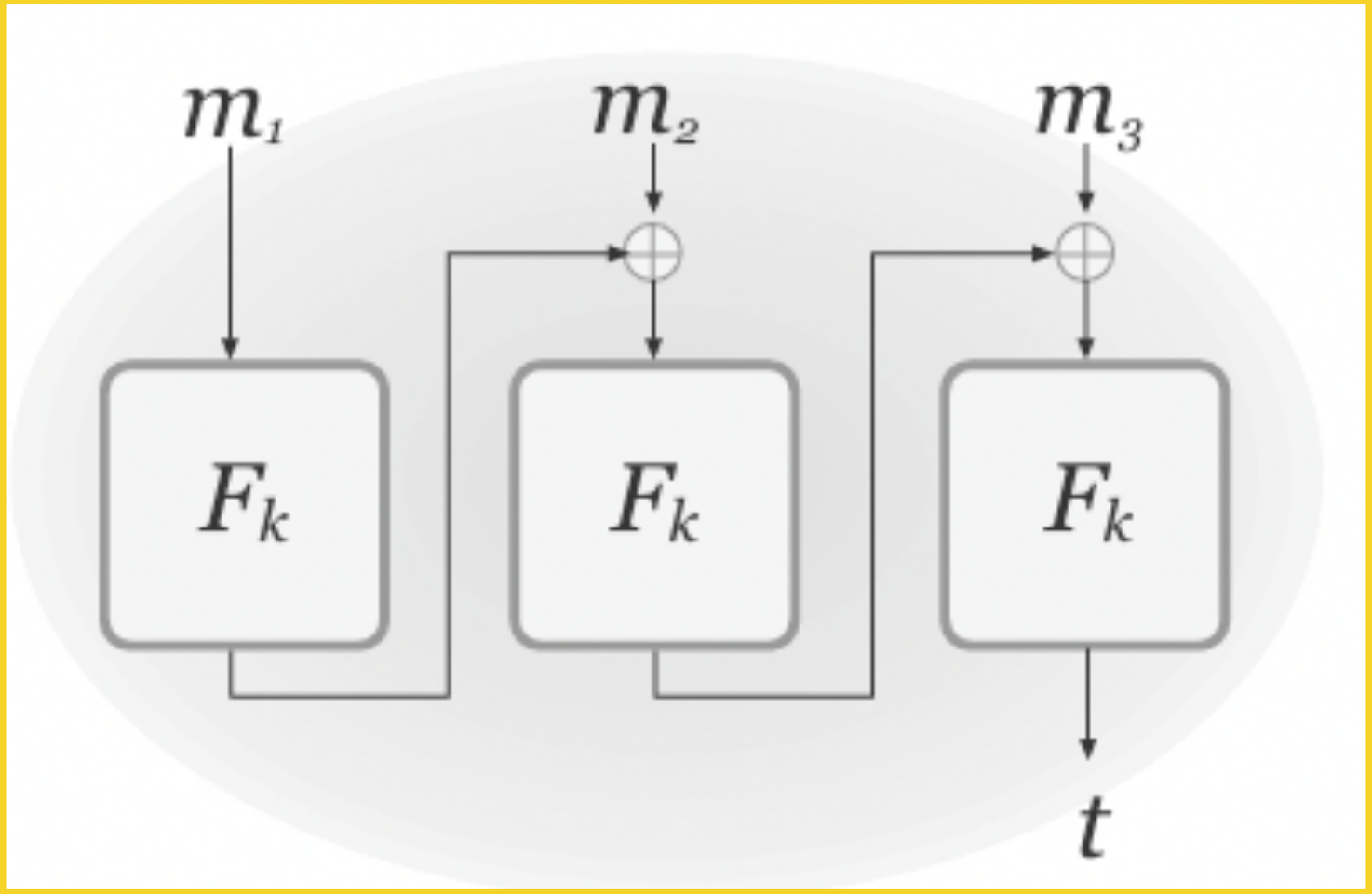
- In the example below, how to forge a tag for a longer message?
- Answer: query MAC oracle for $t \oplus m_4$



m_4

AES$_K$

Another extension attack



- Suppose m_1 has tag t, and (m_1, m_2) has tag t'
- What's the CBC-MAC tag for $(m_1, m_1 \oplus t, m_2)$?
- There are **two fixes**. Read [KL Ch 4.4] for more.
- We don't care too much about CBC-MAC b/c variable-length PRF is more easily obtained from **hash functions**.

HMAC: handle variable lengths

- MAC are more conveniently obtained from **Hash Functions**
- HMAC is a standardized MAC for variable-length inputs.
- Widely used: IPsec, SSH and TLS, etc.
- When you need a MAC, use HMAC

MACs we will use

- HMAC (using collision-resistant hash functions)
  - The go-to MAC used everywhere
    - E.g., TLS 1.2 supports AES-CBC-HMAC
- GMAC (using universal hash function)
  - Primarily used in AES-GCM
  - Your HW1.

Collision-resistant hash functions (CRH)

- Examples: SHA-2, SHA-3, BLAKE3, etc.
- Compress arbitrary inputs to a fixed-length digest (e.g. 256 bits)

	Name	Last modified	Size	Description
	Parent Directory		-	
	SHA256SUMS	2025-11-26 19:49	291	
	SHA256SUMS.gpg			
	netboot/			
	ubuntu-26.04-desktop-amd64.iso	2025-11-01 06:37	5.2G	Desktop image for 64-bit PC (AMD64) computers (standard download)

```
7463ed557a8f9d6af6ea3ff7181a6b8605ae32bd8e1830b83f58cbd071ff4d08 *ubuntu-26.04-desktop-amd64.iso
82cbf409b18c5f8278eb2133426f81d05e33fc661b3ca9c358809de1ff46649f *ubuntu-26.04-live-server-amd64.iso
c77c9e8a5b0255cd02f5edbcd612663976d995980107ce116b2b61f9244cce79 *ubuntu-26.04-wsl-amd64.wsl
```

- Key properties: hard to find *collisions*
  - collisions must *exists* due to the pigeonhole principle

Collision-resistant hash functions (CRH)

- CR of practical hash functions is supported by *cryptanalytic evidence*.
  - i.e., The design follows sound principles. No known attacks despite years of expert analysis. But no theorem reducing security to a math problem.
- There are mathematically CR hash functions (e.g., Lattice based) but they are ~10^4–10^6× slower

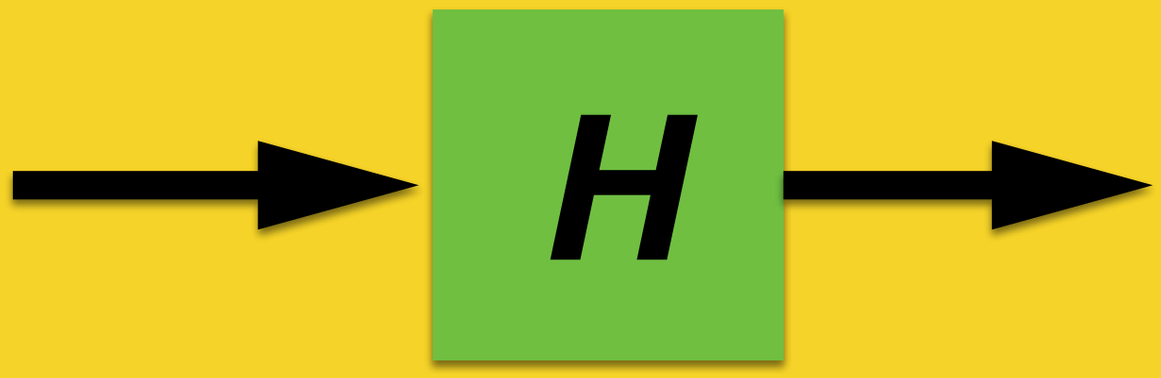
Hash functions

The workhorses of modern cryptography

Hash function

Maps *message x* of any length to fixed-length *digest H*(*x*)

message $x \in \{0,1\}^*$



digest $H(x) \in \{0,1\}^n$

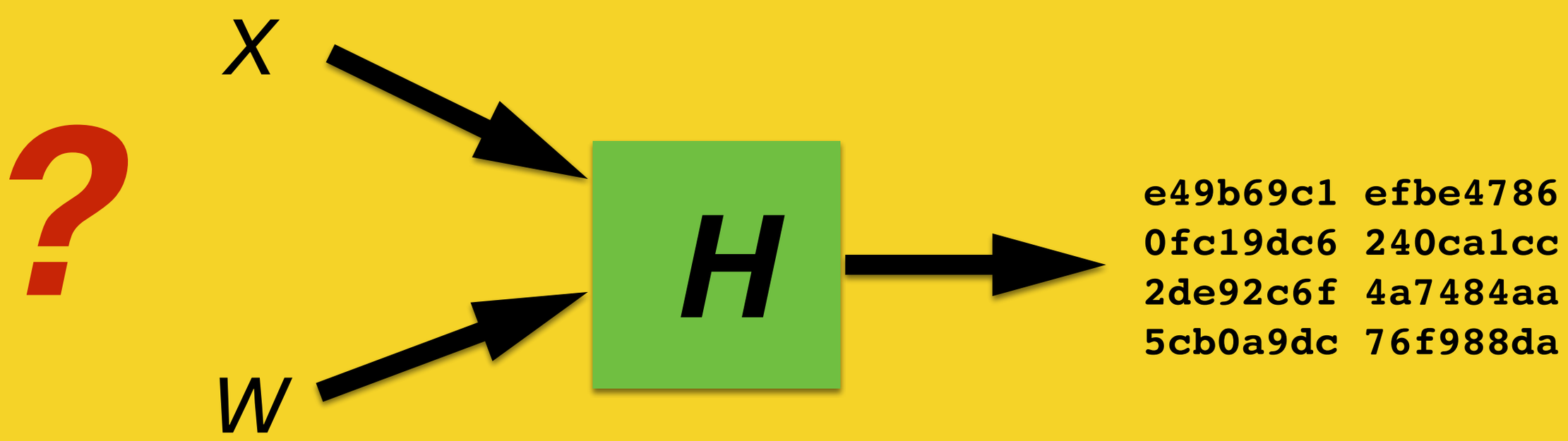
e.g., $n = 256$

```
e49b69c1 efbe4786
0fc19dc6 240ca1cc
2de92c6f 4a7484aa
5cb0a9dc 76f988da
```

Collision-resistance

Collision-resistance: It is hard to find any pair of inputs x, x' such that

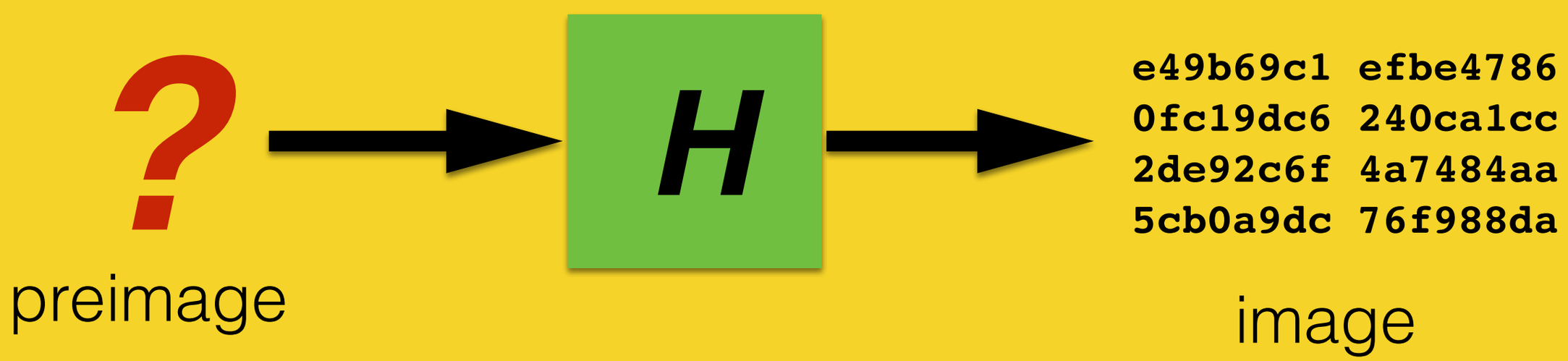
$$H(x) = H(x')$$



Pre-image resistance

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

Pre-image resistance: Given y, no PPT adversary can find any x such that $H(x) = y$



Note: for significantly compressing hash functions, collision resistance implies pre-image resistance [KL Sec 6.1.2].

Birthday paradox

- What's the generic algorithm to find a collisions?

  Do

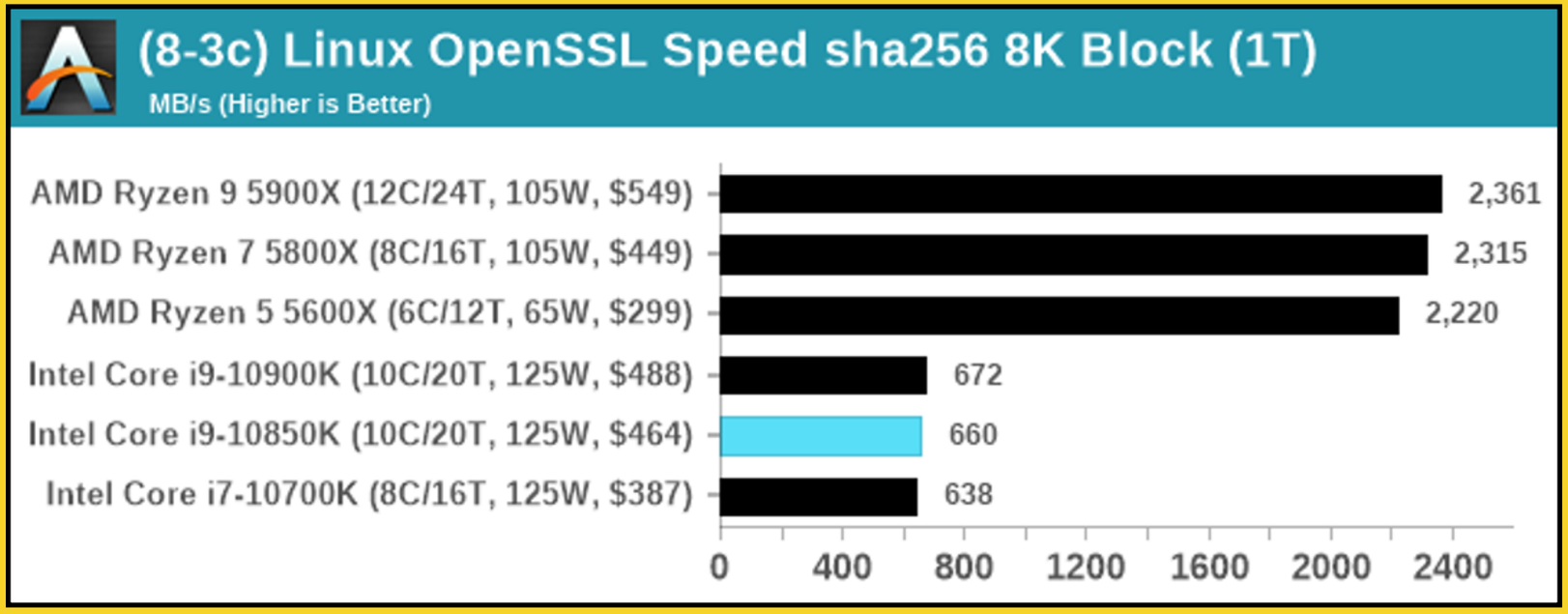
  - Pick a random x
  - Compute $y = H(x)$ and store it

  Until a collision is found

- Given 256-bit hash, like throwing balls into 2^{256} buckets
- By *birthday paradox*, collision w.p. ≈ 1/2 on 2^{128} throws
- That's why you see 128-bit AES keys used with 256-bit hash functions.

Common CRHs

- Inputs to hash function are often large
- To get the fast speed we tend to use hash functions constructed using ad hoc mixing operations
- Common examples are SHA (Secure Hash Algorithm) Family
  - e.g., Modern CPU can compute SHA-256 at 600MB/s – 2GB/s



CRH design evolves

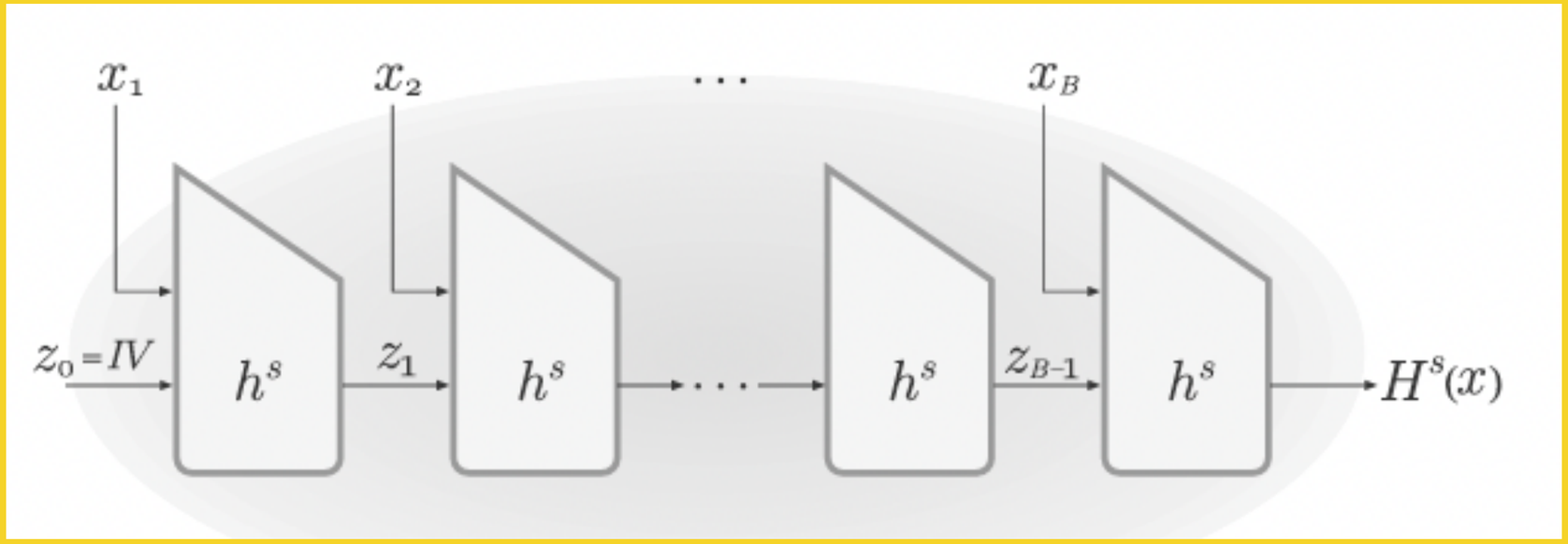
- SHA-1 was designed by NIST / NSA in 1995
  - In 2005, Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu announced an attack that can find collisions in 2^69 steps
  - First collision demonstrated on 23 Feb. 2017 by Google / CWI using 6,500 CPU-years and 110 GPU-years
- SHA-2 family is still believed to be secure and in wide use
  - Individual hash functions are named SHA-n (n∈{224,256,384,512})
- SHA-3 released by NIST in 2015 after a 5-year design competition
  - not doing Merkle-Damgård

Is SHA256($K||m$) a secure MAC?

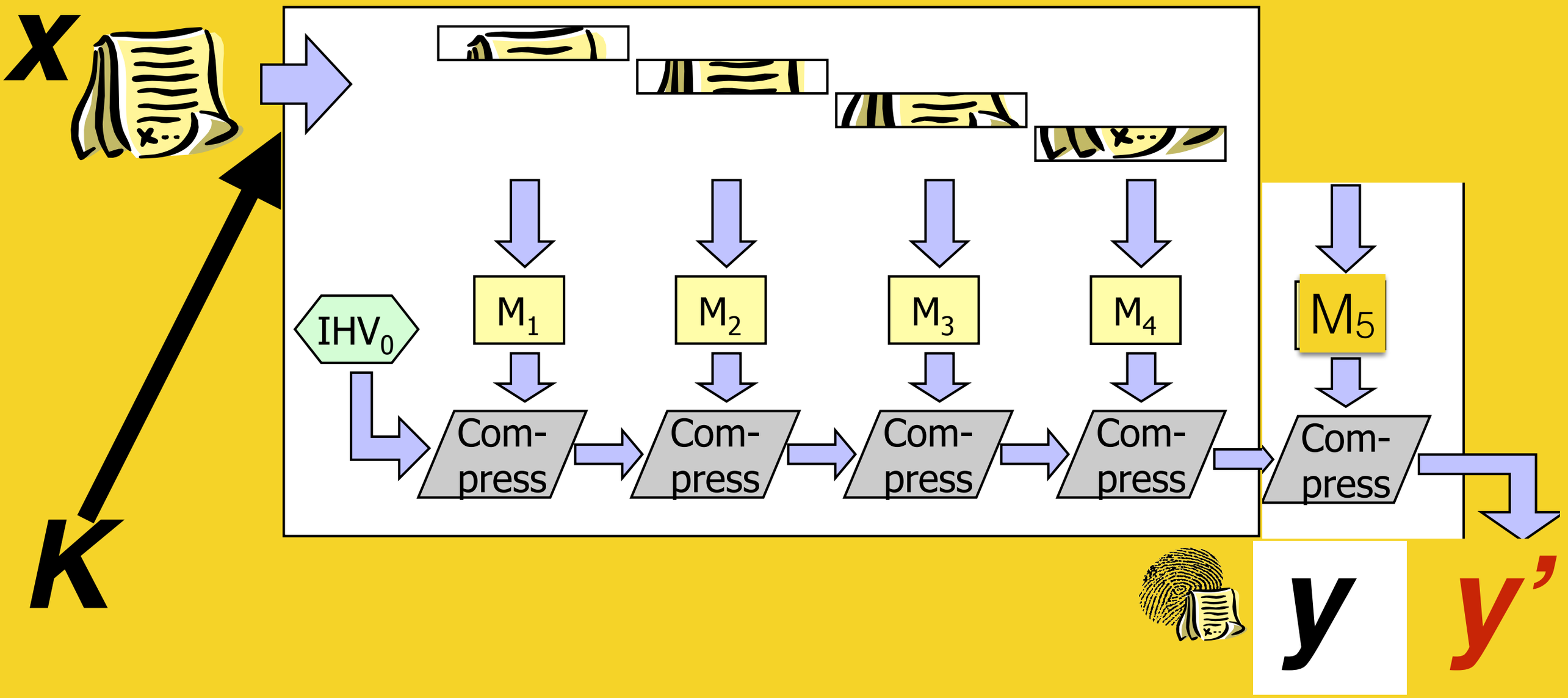
- Secure for <u>fixed-length</u> messages.
- Not secure for variable length inputs. Extension attacks possible for certain hash functions.

Merkle-Damgård Transformation

- Suppose $h : \{0,1\}^{256} \to h : \{0,1\}^{128}$, then we can build $H : \{0,1\}^* \to \{0,1\}^{128}$ as follows
  - Chop up x into 128-bit blocks $x_1, \cdots, x_B$
  - Let $z_0 = IV$ (fixed value); then $z_i = h(x_i \| z_{i-1})$
  - Output z_B as $H(x)$
- Security theorem: if h is collusion resistant, so is H.
- SHA-1 and SHA-2 families are built this way



Length extension attack



Given (x, T), one can forge a new pair: (x||m_5, Comp(T||m_5))

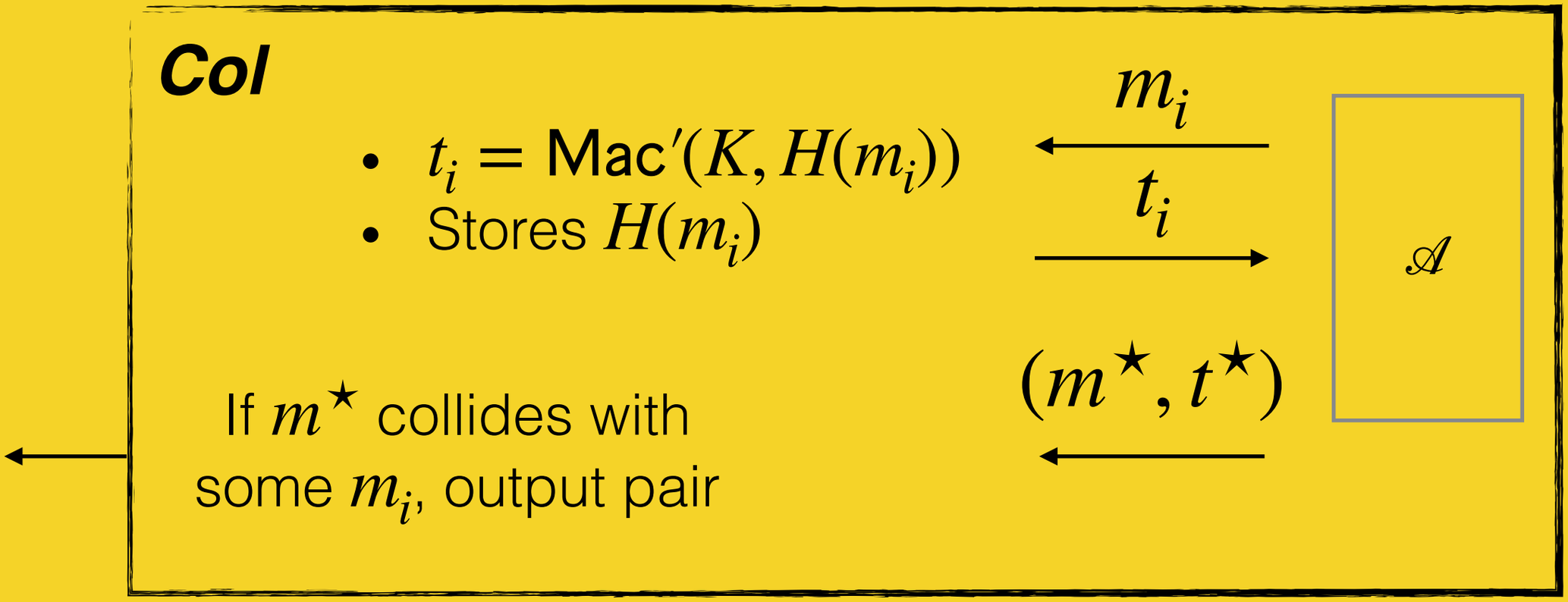
Hash then MAC

- How to convert messages to fixed length digest?
- Use cryptographic hash functions!
  - Let Mac' be a secure MAC for messages of length ℓ,
  - let H be a hash function with output length ℓ
  - then $\mathsf{Mac}(K, m) := \mathsf{Mac}'(K, H(m))$ is a secure MAC for arbitrary-length messages.

- Proof by construction: suppose $\mathcal{A}$ can win $\mathsf{UF}_{\mathcal{A},\mathsf{Mac}}$ with non-neg probability by outputting $(m^\star, t^\star)$.
- Let Q denote the set of messages $\mathcal{A}$ queried.
- Let C denote the event that $\exists m \in Q$ s.t. $H(m^\star) = H(m)$

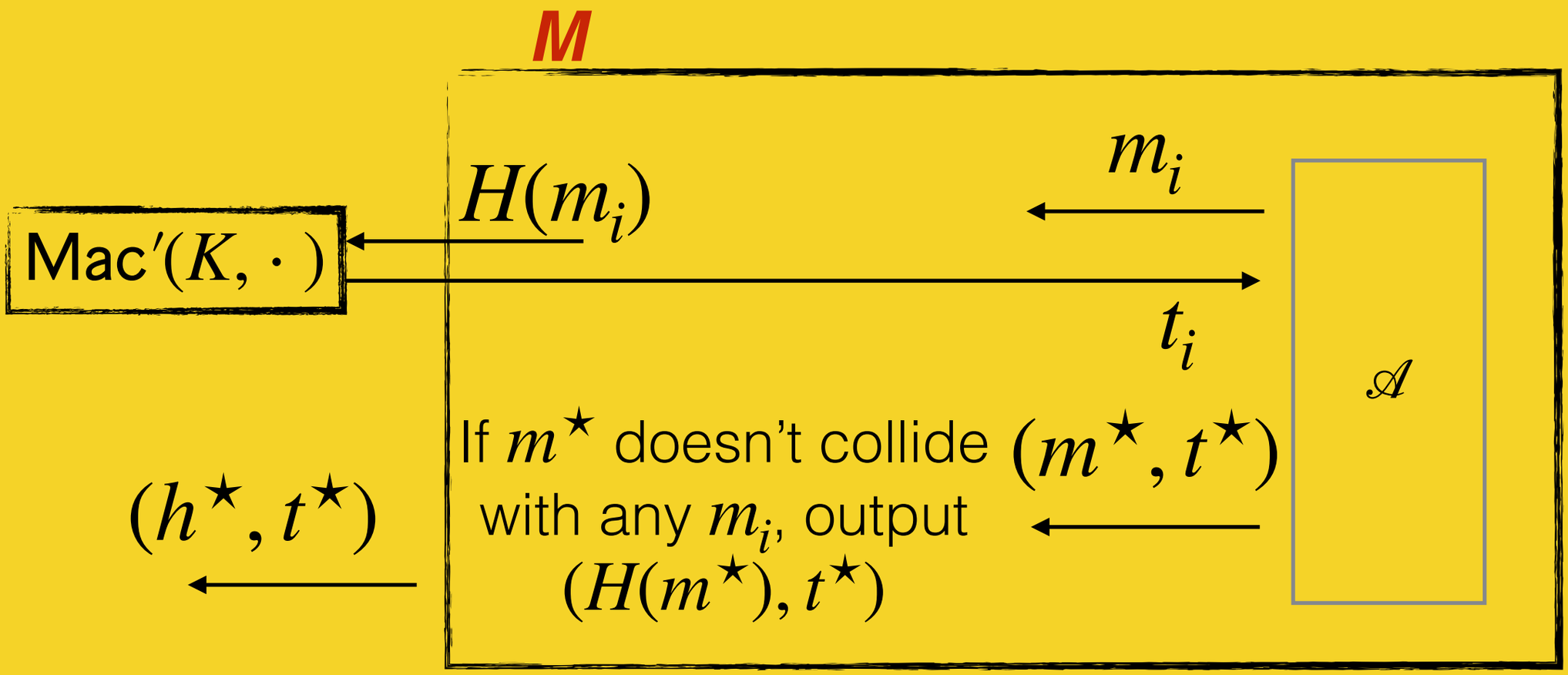
- Pr[win]= Pr[win ^ C] + Pr[win ^ ¬C]
- Pr[win]≤ Pr[C] + Pr[win ^ ¬C]
- Claim:
  - 1) Pr[C] is negligible // collision resistance
  - 2) Pr[win ^ ¬C] is negligible // security of MAC*

Claim 1



$$Pr[C] = Pr[\text{Col finds collisions}] = \mathbf{neg}(n)$$

Claim 2



$$Pr[\mathcal{A} \text{ wins} \wedge \neg C] = Pr[M \text{ breaks } \mathsf{Mac}'] = \mathsf{neg}(n)$$

Instantiation

- Hash then CBC-MAC
  - We first hash the message m to get a fixed-length digest $H(m)$
  - Outputs CBC-MAC of $H(m)$
  - Security: By collision resistance and unforgeability of CBC-MAC
    - H guarantees that inputs to CBC-HMAC is fixed-length
- Not popular in practice: partly b/c of the need of two crypto primitives.

1996 by M. Bellare, R. Canetti, and H. Krawczyk

HMAC

- Let H be a CRH
- k1 = k ⊕ ipad (ipad = 0x36...36)
- k2 = k ⊕ opad (opad = 0x5c...5c)
- HMAC(k, m) = H$(k_1 \| \text{H}(k_2 \| m))$

$$\mathrm{HMAC}(k, m) = \mathrm{H}(k_1 \| \mathrm{H}(k_2 \| m))$$

Security intuition

- *Why two keys?* HMAC derives k_1, k_2 from a single key to save key length.
- *Why inner hash?* Inter hash turns variable length m to *fixed length* H(m). Prevents Length-extension.
- *Why inner key? I.e., why not* $\mathrm{H}(k \| \mathrm{H}(m))$*?*
  - Intuitively, finding collusions in H is easier than finding collisions in $H(k, \cdot)$ without knowing k
  - HMAC is proven secure assuming “weakly” collision resistance of H

Other special hash functions

- We will introduce when needed.
  - Memory-hard hash functions
    - Designed to consume a lot of memory
  - Password hash functions
    - Designed to be slow
- SHA-256 and SHA-3 are the go-to *general purpose* hash functions today.

Combining Mac and Enc is non-trivial

- Enc then Mac
  - C = Enc(k, m), T = Mac(K, Enc(k, m))
- Mac then Enc
  - C = Enc(m || Mac(k, m))
- Enc and Mac
  - Enc(K, m), Mac (K, m)
- Enc-then-Mac has the strongest security. We will talk about CCA attacks later.

Usage in TLS

- TLS 1.2 supports **AES-CBC-HMAC**, **ACM-CTR + GMAC** (called AES-GCM)
- TLS 1.3 only supports **AES-GCM**
- CBC-HMAC is generally considered legacy
  - Combining two moving parts ⇒ more opportunities for mistakes
- HMAC is still used widely outside CBC-HMAC

Suggested way: AEAD

- Authenticated Encryption with Additional Data
- No moving parts: A single primitive for both confidentiality and integrity.
- (C, T) = AEAD-Encrypt(K, IV, m, ad)
  - T also covers plaintext additional data (ad).
  - E.g., seq num of TLS packets, useful for ordering
- AES-GCM (i.e., ACM-CTR + GMAC) is a primary example.

GMAC from Universal Hash Function*

GHASH

- $\mathrm{GHASH}_H(M) = m_1 \cdot H^\ell + m_2 \cdot H^{\ell-1} + \cdots + m_\ell \cdot H$
  - Given secret key H
  - Split message M into blocks $m_1, \ldots, m_\ell \in F$. Build an ℓ-degree polynomial $p_M(x) = m_1 x^\ell + \cdots + m_\ell x$
  - Output $p_M(H)$
- Property (universal hash): for random H, any pair of m, m', Pr[GHASH_H(m) = GHASH_H(m')] $\leq \frac{\ell + 1}{|F|}$
  - (In fact, a stronger property (DUF) holds for GHASH.)

GHASH → GMAC

- Observe that GHASH alone is not a secure MAC
  - $\mathrm{GHASH}_H(M_1 + M_2) = \mathrm{GHASH}_H(M_1) + \mathrm{GHASH}_H(M_2)$
  - Also, attacker can learn H by one query
- In AES-GCM, tag is encrypted GHASH value, i.e.,Tag $= \mathrm{GHASH}_H(M) + \mathrm{AES}_K(\mathrm{IV})$
- In AES-GCM, $F = \mathrm{GF}(2^{128})$
  - $+$ is XOR, and $\cdot$ is long multiplication with reduction

input: key $k \in \mathcal{K}$, message m, associated data d, and nonce $\mathcal{N} \in \{0,1\}^{96}$

$k_{\mathrm{m}} \leftarrow E(k,\ 0^{128})$ // *first, generate the key for GHASH (a variant of* H_{xpoly}*)*

Compute the initial value of the counter in counter mode encryption:

$x \leftarrow (\mathcal{N} \parallel 0^{31}1) \in \{0,1\}^{128}$

$x' \leftarrow x + 1$ // *initial value of counter*

$c \leftarrow \{\text{encrypt } m \text{ using counter mode starting the counter at } x'\}$

$d' \leftarrow \{\text{pad } d \text{ with zeros to closest multiple of 128 bits}\}$

$c' \leftarrow \{\text{pad } c \text{ with zeros to closest multiple of 128 bits}\}$

Compute the Carter-Wegman MAC:

$$h \leftarrow \mathrm{GHASH}\Big(k_{\mathrm{m}},\ \big(d' \parallel c' \parallel \mathrm{length}(d) \parallel \mathrm{length}(c)\big)\Big) \ \in \{0,1\}^{128}$$

$t \leftarrow h \oplus E(k,x) \ \in \{0,1\}^{128}$

output (c,t) // *encrypt-then-MAC ciphertext*

You can ignore jargons like H_{xpoly} and Carter-Wegman.

- AES-GCM is the recommended encryption to use in practice
- Homework 1 ask you to read NIST standard on AES-GCM and finish implementation in Python

NIST Special Publication 800-38D
November, 2007

National Institute of Standards and Technology

U.S. Department of Commerce

Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC

Morris Dworkin

Algorithm 4: GCM-AE$_K$ (IV, P, A)

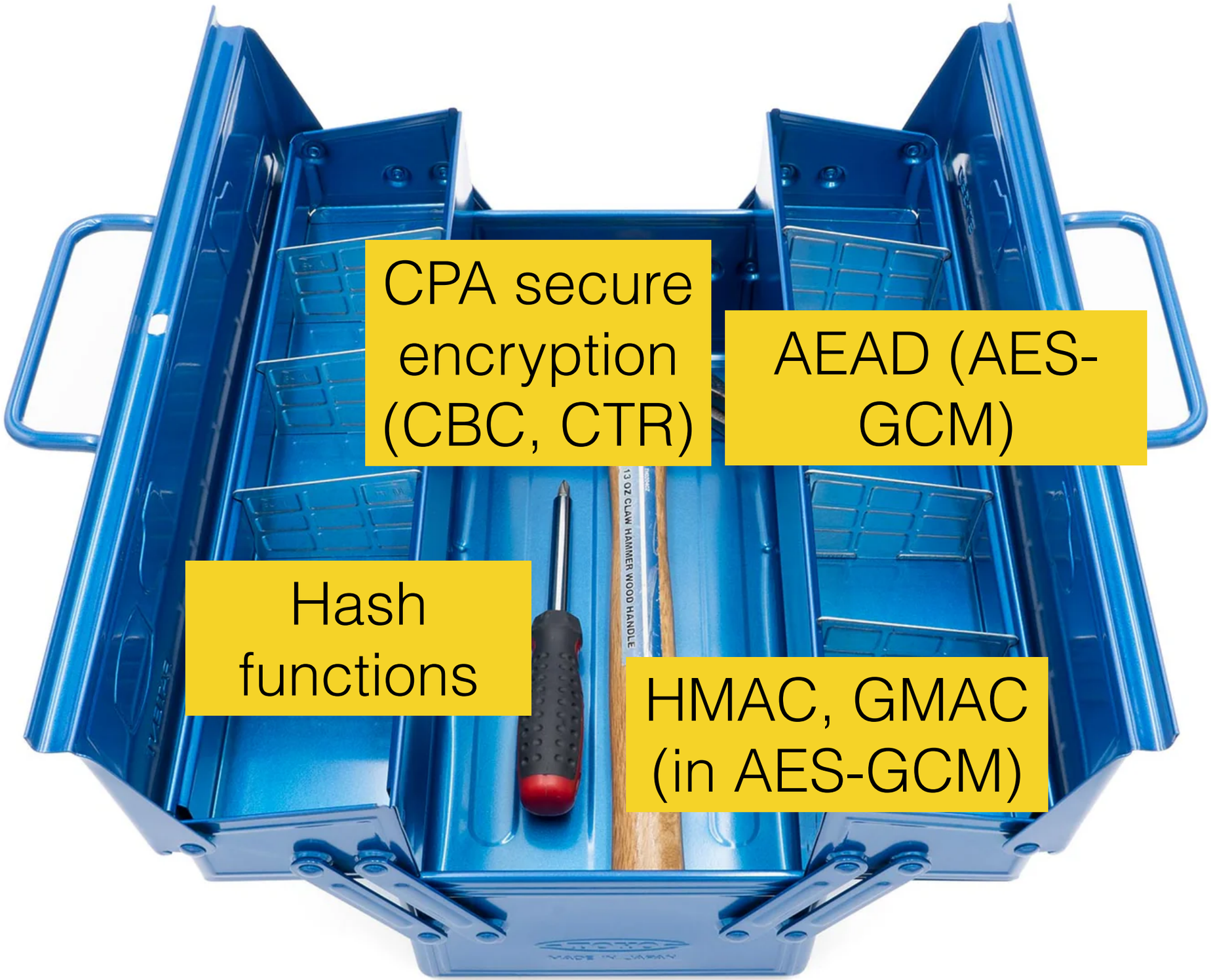
Prerequisites:
approved block cipher CIPH with a 128-bit block size;
key K;
definitions of supported input-output lengths;
supported tag length t associated with the key.

Input:
initialization vector IV (whose length is supported);
plaintext P (whose length is supported);
additional authenticated data A (whose length is supported).

Output:
ciphertext C;
authentication tag T.

Steps:

1. Let $H = \text{CIPH}_K(0^{128})$.
2. Define a block, J_0, as follows:
   If len(IV)=96, then let $J_0 = IV \,\|\, 0^{31} \,\|1$. `// we assume len(IV)=96`
   If len(IV) $\neq$ 96, then let $s = 128\lceil \text{len}(IV)/128 \rceil - \text{len}(IV)$, and let
   $J_0 = \text{GHASH}_H(IV \| 0^{s+64} \| [\text{len}(IV)]_{64})$.
3. Let $C = \text{GCTR}_K(\text{inc}_{32}(J_0), P)$.
4. Let $u = 128 \cdot \lceil \text{len}(C)/128 \rceil - \text{len}(C)$ and let $v = 128 \cdot \lceil \text{len}(A)/128 \rceil - \text{len}(A)$.
5. Define a block, S, as follows:
   $$S = \text{GHASH}_H(A \,\|\, 0^v \,\|\, C \,\|\, 0^u \,\|\, [\text{len}(A)]_{64} \,\|\, [\text{len}(C)]_{64}).$$
6. Let $T = \text{MSB}_t(\text{GCTR}_K(J_0, S))$.
7. Return (C, T).


CPA secure encryption (CBC, CTR)
AEAD (AES-GCM)
Hash functions
HMAC, GMAC (in AES-GCM)

Now, can we use them on the Internet?

Main challenge:
How to share *K* safely to begin with?

Alice

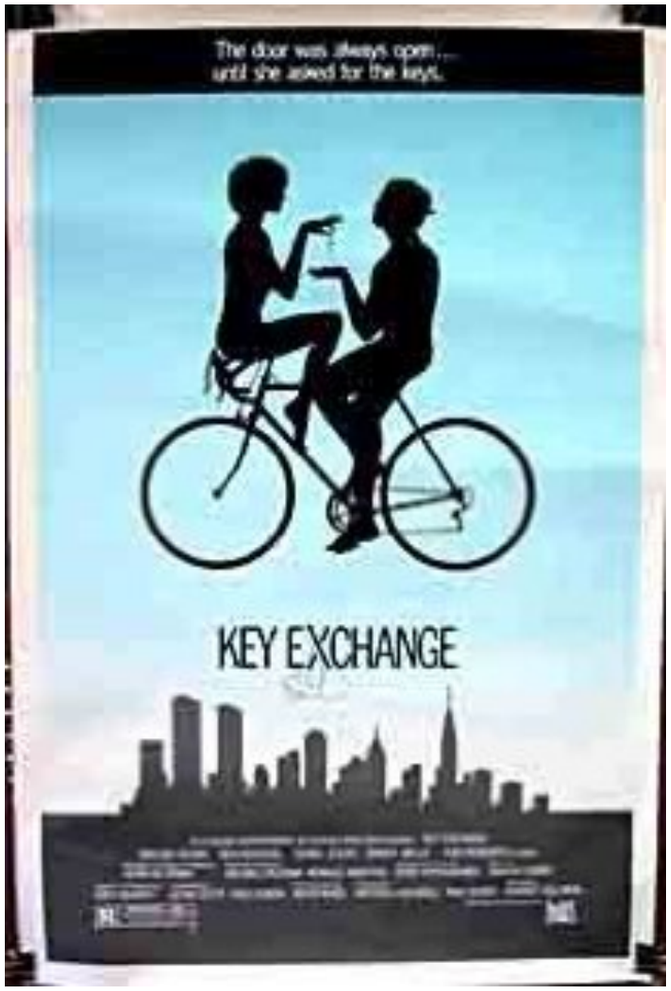
K

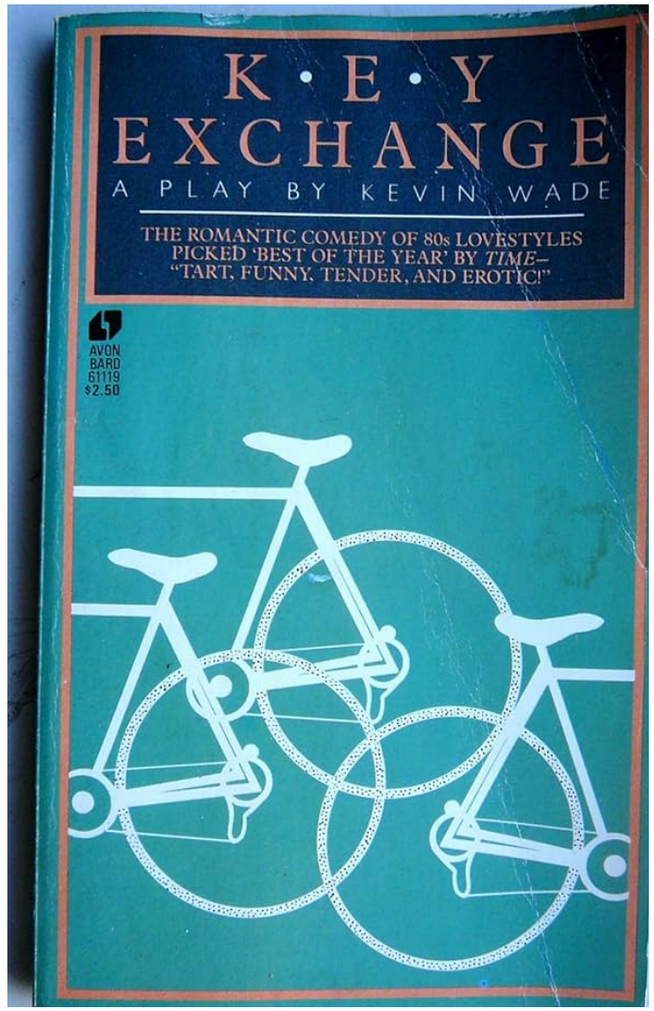
(secret)

Bob

K

Unfortunately, meeting under bridge at night not scalable for Internet


The door was always open...
until she asked for the keys.
KEY EXCHANGE


K·E·Y
EXCHANGE
A PLAY BY KEVIN WADE
THE ROMANTIC COMEDY OF 80s LOVESTYLES
PICKED 'BEST OF THE YEAR' BY TIME–
"TART, FUNNY, TENDER, AND EROTIC!"
AVON
BARD
61119
$2.50

Next: TLS