MATH/CMSC 456 :: UPDATED COURSE INFO

Instructor: Gorjan Alagic (galagic@umd.edu); ATL 3102, office hours: by appointment

Textbook: *Introduction to Modern Cryptography*, Katz and Lindell;

Webpage: alagic.org/cmsc-456-cryptography-spring-2020/ (slides, reading);

Piazza: piazza.com/umd/spring2020/cmsc456

ELMS: active, slides and reading posted there, **first homework is up (due midnight Thursday.)**

Gradescope: active, access through ELMS.

TAs (Our spot: shared open area across from IRB 5234)

- Elijah Grubb (egrubb@cs.umd.edu) 11am-12pm TuTh (Iribe);
- Justin Hontz (jhontz@terpmail.umd.edu) 1pm-2pm MW (Iribe);

Additional help:

- Chen Bai (cbai1@terpmail.umd.edu) 3:30-5:30pm Tu (2115 ATL, starting Feb 4)
- Bibhusa Rawal (bibhusa@terpmail.umd.edu) 3:30-5:30pm Th (2115 ATL, starting Feb 6)

HOMEWORK RULES AND GUIDELINES:

First homework is up (due midnight Thursday.)

Rules

- collaboration ok, solutions must be written up by yourself, in your own words;
- late homeworks will not be accepted (*no exceptions*, but lowest grade will be dropped.)

Explanations and proofs

- correct answers with no explanation will get a zero score;
- explain your ideas clearly and completely;
- write in complete sentences, use correct and complete mathematical notation (as in lectures and book);
- proofs need to be rigorous, clear, and complete (consider all cases, prove counterexamples, etc.)

Suggestions

- work on your own at least some of the time for each assignment
- work in 25+ minute chunks of uninterrupted, distraction-free, device-free time
- develop intuition: try lots of examples, ask yourself questions, “play” with the concepts

RECAP: SEMANTIC SECURITY

Recall: "semantic security" is a meaningful, intuitive notion;
It says something like this:

"observing the ciphertext doesn't help the adversary learn anything ***new*** about the plaintext."

A bit more carefully:

"no matter what the adversary $\underline{\text{already knows}}$ about the plaintext…
… observing the ciphertext $\underline{\text{doesn't help}}$ him $\underline{\text{learn anything}}$ more."

Things to capture formally:

- existing knowledge;
- new knowledge;
- "doesn't help".

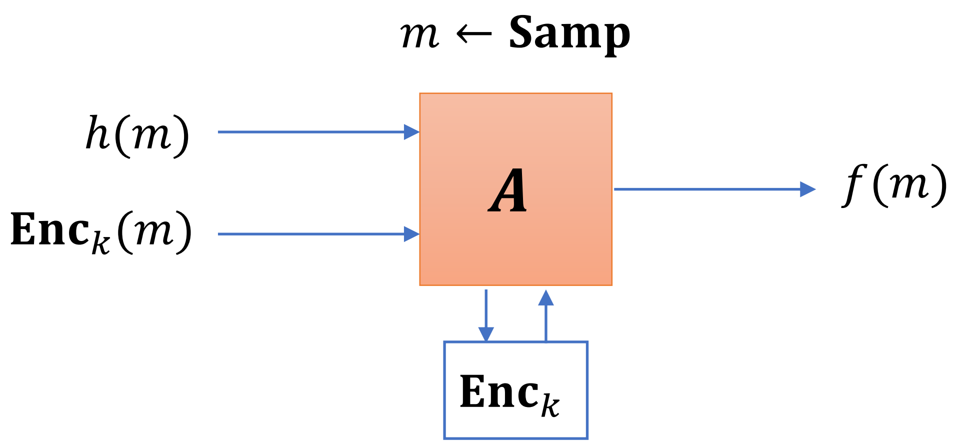
RECAP: SEMANTIC SECURITY under CHOSEN PLAINTEXT ATTACK

Formally:

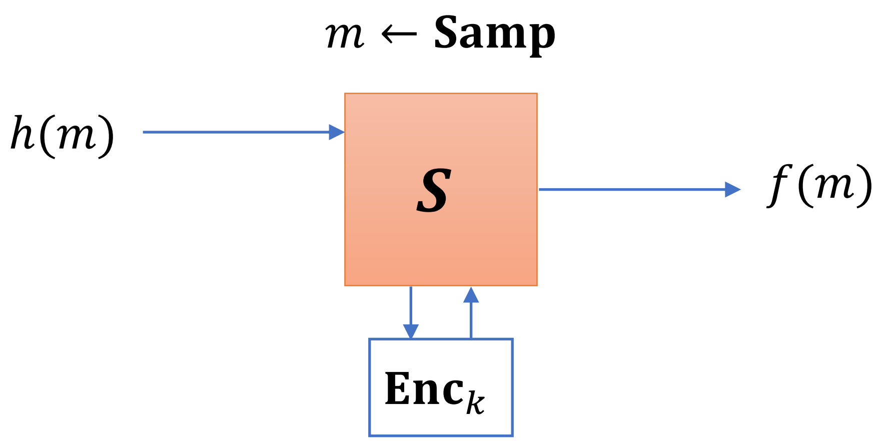
Definition. An encryption scheme $(\mathbf{KeyGen}, \mathbf{Enc}, \mathbf{Dec})$ is **SEM-CPA secure** if, for every PPT algorithm $\boldsymbol{A}$ ("adversary") there exists a PPT algorithm $\boldsymbol{S}$ ("simulator") such that the following holds.

For every PPT algorithm $\mathbf{Samp}$ and every pair of poly-time computable functions h and f,

$$\left|\Pr_{m \leftarrow \mathbf{Samp}}\left[\boldsymbol{A}^{\mathbf{Enc}_k}(h(m), \mathbf{Enc}_k(m)) = f(m)\right] - \Pr_{m \leftarrow \mathbf{Samp}}\left[\boldsymbol{S}^{\mathbf{Enc}_k}(h(m)) = f(m)\right]\right| \leq \mathrm{negl}(n).$$



vs



RECAP: SEM-CPA vs IND-CPA

Definition. An encryption scheme $(\mathbf{KeyGen}, \mathbf{Enc}, \mathbf{Dec})$ is **SEM-CPA** if, for every PPT algorithm $\boldsymbol{A}$ ("adversary") there exists a PPT algorithm $\boldsymbol{S}$ ("simulator") such that the following holds.

For every PPT algorithm $\mathbf{Samp}$ and every pair of poly-time computable functions h and f,

$$\left| \Pr_{m \leftarrow \mathbf{Samp}}\left[A^{\mathbf{Enc}_k}(h(m), \mathbf{Enc}_k(m)\ ) = f(m)\right] - \Pr_{m \leftarrow \mathbf{Samp}}\left[S^{\mathbf{Enc}_k}\big(h(m)\big) = f(m)\right] \right| \le \operatorname{negl}(n).$$

This is really messy. IND-CPA is way simpler to work with.

Theorem. **IND-CPA** $\Leftrightarrow$ **SEM-CPA**.

Totally awesome:

- we get the real, meaningful security strength promised by semantic security…
- … but we can use the much simpler and cleaner indistinguishability definition.
- (for example, when doing proofs!)

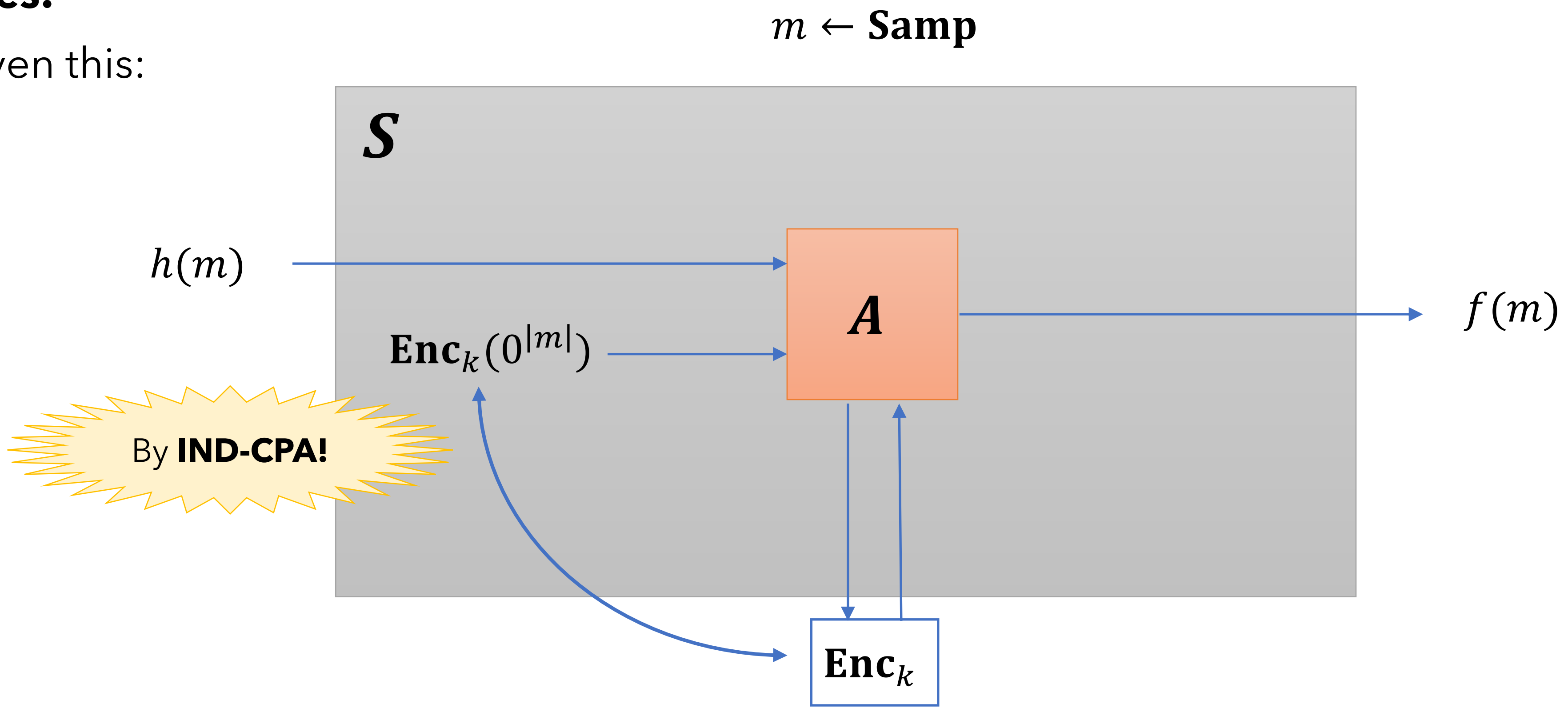
RECAP: SEM-CPA = IND-CPA

Claim: **IND-CPA** ⇒ **SEM-CPA**.

- suppose a scheme is **IND-CPA**;
- let's prove that it must also be **SEM-CPA**;
- direct approach: show how to, given any adversary $\boldsymbol{A}$, construct a simulator $\boldsymbol{S}$.

In pictures:

We're given this:



RECAP: PRF ENCRYPTION is IND-CPA

Construction (PRF encryption). Let $\boldsymbol{F}: \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^\ell$ be a PRF. Define a scheme:

- $\mathbf{KeyGen}$: sample a PRF key $k \leftarrow \{0,1\}^n$;
- $\mathbf{Enc}$: on input a message $m \in \{0,1\}^\ell$, sample $r \leftarrow \{0,1\}^m$ and output $(r, \boldsymbol{F}_k(r) \oplus m)$;
- $\mathbf{Dec}$: on input a ciphertext (r, c), output $c \oplus \boldsymbol{F}_k(r)$.

Theorem. The PRF scheme is **IND-CPA**.

What's the proof idea?

1. by the PRF property, I should be able to replace $\boldsymbol{F}_k$ above with a totally random function $\boldsymbol{R}$…

   … without the adversary noticing the difference.
2. but after that replacement, look at how the scheme works:
   - for each message m…
   - … we pick a random string of the same length as m (specifically, $\boldsymbol{R}(r)$);
   - and we use it to one-time pad m!
3. this is perfectly secret!

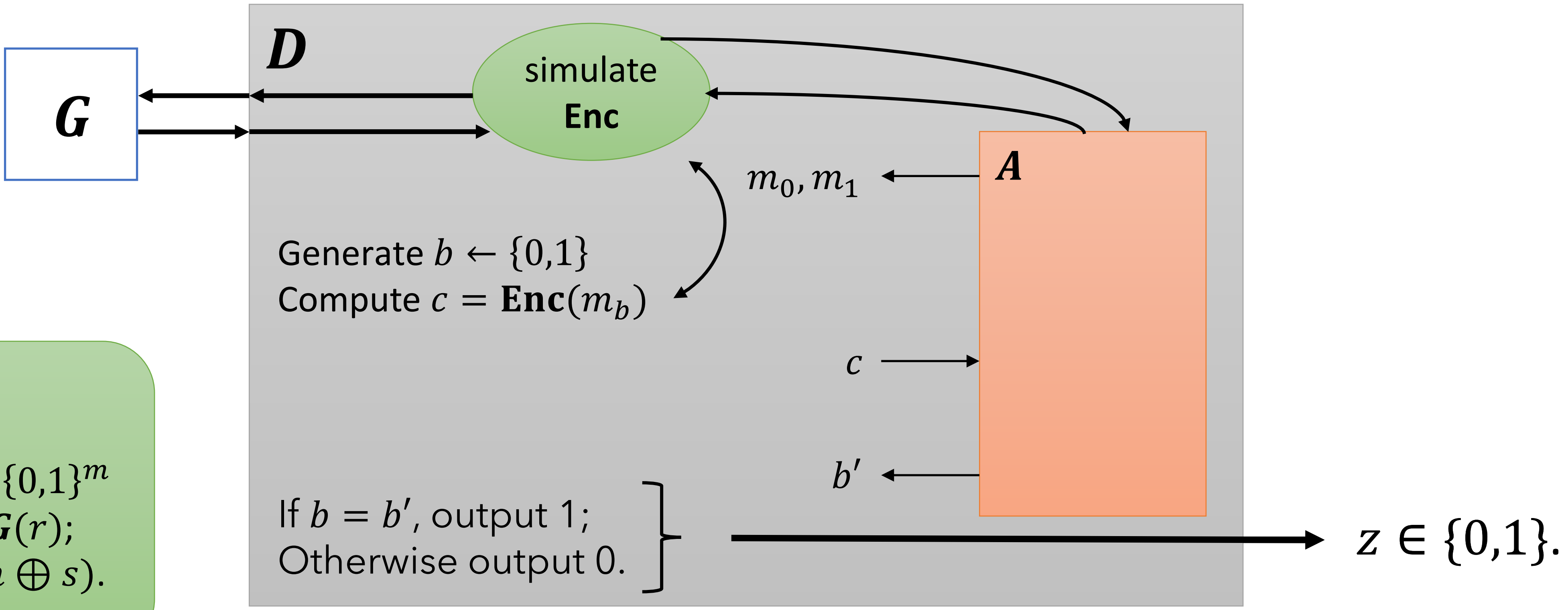
RECAP: PRF SCHEME IS IND-CPA

Key claim. For every PPT $\boldsymbol{A}$,

$$|\Pr[\boldsymbol{A} \text{ wins INDCPA experiment vs } \Pi_{\mathbf{PRF}}] - \Pr[\boldsymbol{A} \text{ wins INDCPA experiment vs } \Pi_{\boldsymbol{RF}}]| \leq \mathrm{negl}(n).$$

Strategy: if claim is false, then we can build a distinguisher between **PRF** and **RF** (& violate **PRF** property!)

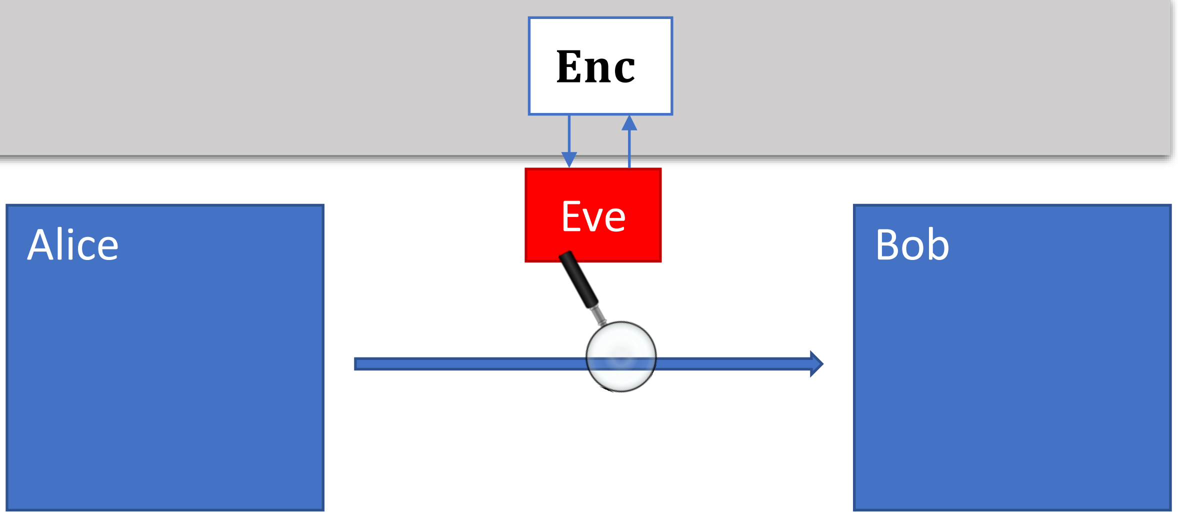
What's the distinguisher? *It's a simulation of the INDCPA experiment vs* $\boldsymbol{A}$*!*



simulate **Enc**:

- on input m;
- sample $r \leftarrow \{0,1\}^m$
- query: $s = \boldsymbol{G}(r)$;
- output $(r, m \oplus s)$.

RECAP: SECRECY and ENCRYPTION



Our focus so far: secrecy.

Two settings:

1. Perfect secrecy (information-theoretic)
    - One-time pad: n bit key encrypts n-bit message
    - Shannon's theorem: one-time pad is optimal.

2. Computational secrecy
    - pseudorandom generators: n-bit key, $\mathrm{poly}(n)$-bit messages;
    - pseudorandom functions: n-bit key, $\mathrm{poly}(n)$-many $\mathrm{poly}(n)$-bit messages;
    - much more powerful security: even secret against *chosen plaintext attacks* (CPA);
    - security proofs.

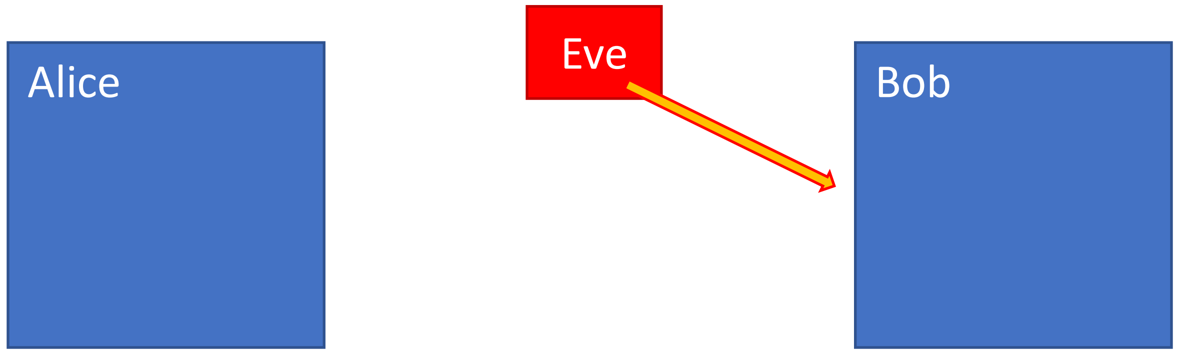
IS CRYPTO JUST SECRECY?

Secrecy: protects against Eve learning our message.

What else could go wrong?

Eve could ***spoof***!

Is this possible? The message is encrypted!



Consider OTP:

- Eve simply sends a string c;
- Bob decrypts: $\mathbf{Dec}_k(c) = c \oplus k = m$.
- Bob has no way to tell if the message was authentic (besides looking at its structure.)

If messages are supposed to be highly structured (e.g., English sentences), maybe this is ok.

If it's just data, then maybe not.

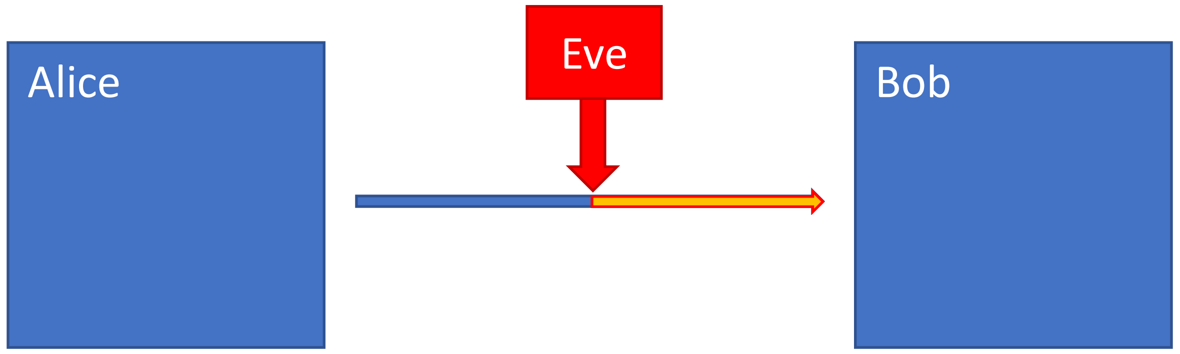
IS CRYPTO JUST SECRECY?

Secrecy: protects against Eve learning our message.

What else could go wrong?

Eve could ***interfere***!

Is this possible? The message is encrypted!



Consider OTP:

- Eve observes a ciphertext $c = \mathbf{Enc}_k(m) = \boxed{m} \oplus k$;
- She flips some bits: $c \mapsto c \oplus s$;
- Bob decrypts: $\mathbf{Dec}_k(c \oplus s) = c \oplus s \oplus k = s \oplus c \oplus k = \boxed{s \oplus m.}$
- *Eve's attack was directly applied to the message!*

If m was a bank deposit, Eve could flip the bits that add thousands (or millions) to the amount!

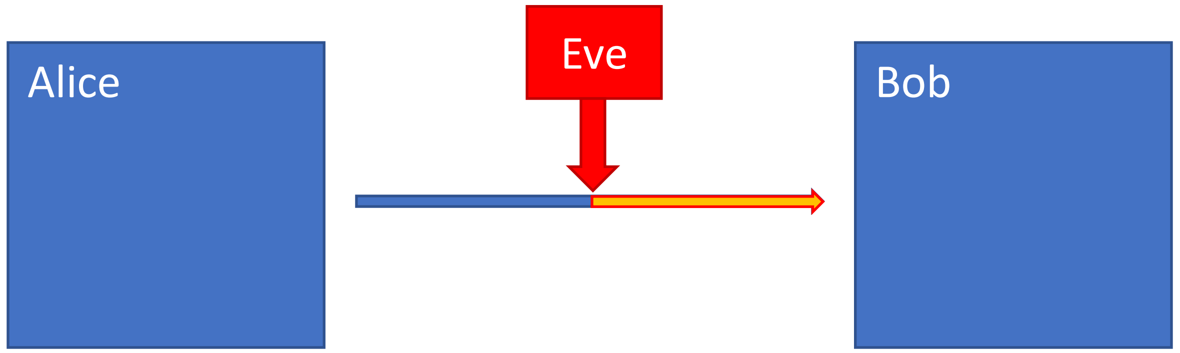
IS CRYPTO JUST SECRECY?

Secrecy: protects against Eve learning our message.

What else could go wrong?

Eve could ***interfere***!

Is this possible? The message is encrypted!



Consider OTP:

- Eve observes a ciphertext $1111 = 0000 \oplus 1111$
- She flips some bits: $1111 \mapsto 1110$
- Bob decrypts: $1110 \oplus 1111 = 0001.$
- *Eve's attack was ("flip last bit") directly applied to the message!*

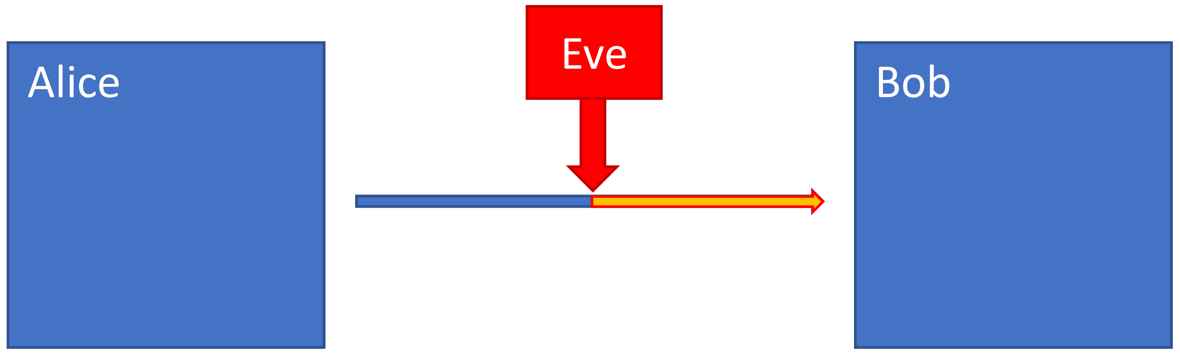
for example

WHAT ABOUT FANCIER ENCRYPTION?

What about PRG and PRF encryption?

Both based on OTP!

So same attacks work!



For example, interference against PRF scheme:

- Eve observes a ciphertext $(r, c) \coloneqq \mathbf{Enc}_k(m) = (r, m \oplus \boldsymbol{F}_{\mathrm{k}}(r))$;
- She flips some bits: $(r, c) \mapsto (r, c \oplus s)$;
- Bob decrypts: $\mathbf{Dec}_k(r, c \oplus s) = c \oplus s \oplus \boldsymbol{F}_k(r) = s \oplus m$.
- *Eve's attack was directly applied to the message!*

All the extra ***secrecy*** protection of the PRF scheme did not help at all!

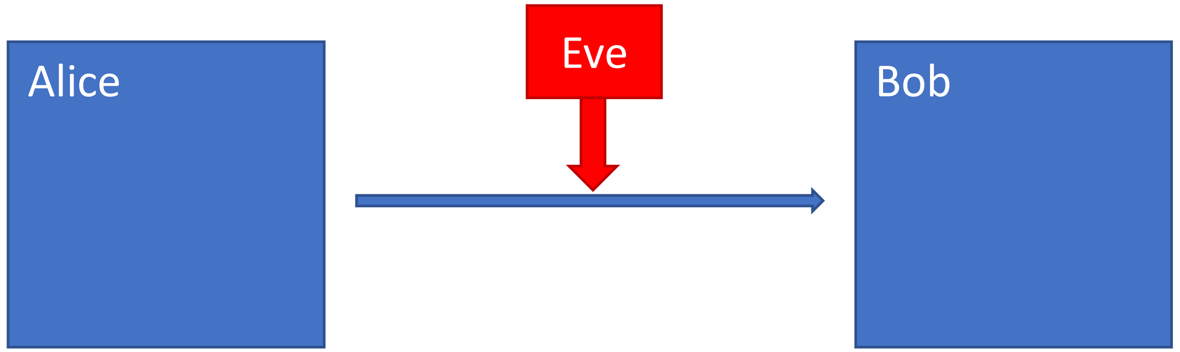
V. AUTHENTICATION

Reading: (p.107-123, 142-145)

AUTHENTICATION

We now change tasks:

- forget *secrecy* for the moment!
- and instead consider *authenticity.*
- (we will talk about combining them later.)



The task:

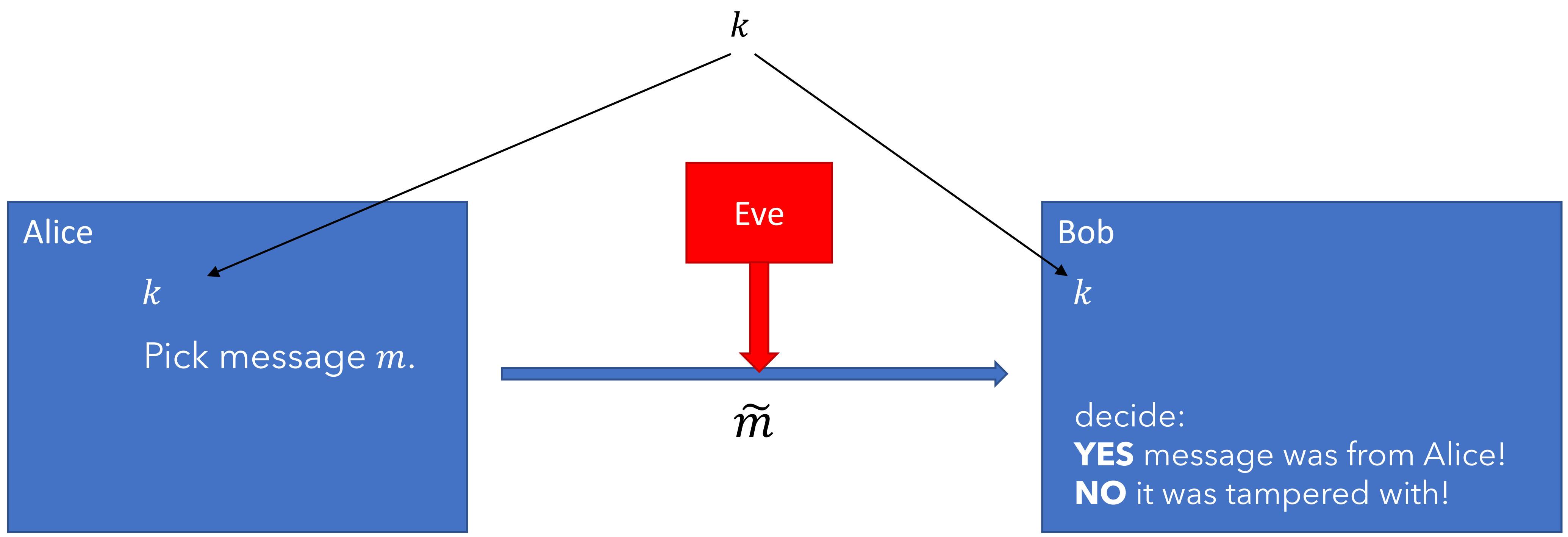
- Alice wants to send a message to Bob;
- Bob’s goal: make sure message is really from Alice…
- … and nobody else!

Assumptions:

- Alice and Bob can share a secret in advance (and have private spaces);
- Alice can send only one transmission (for now);
- *Eve can change (or replace) the transmission however she likes!*
- *(… but we don’t care if she can learn the message.)*

AUTHENTICATION

In a bit more detail.

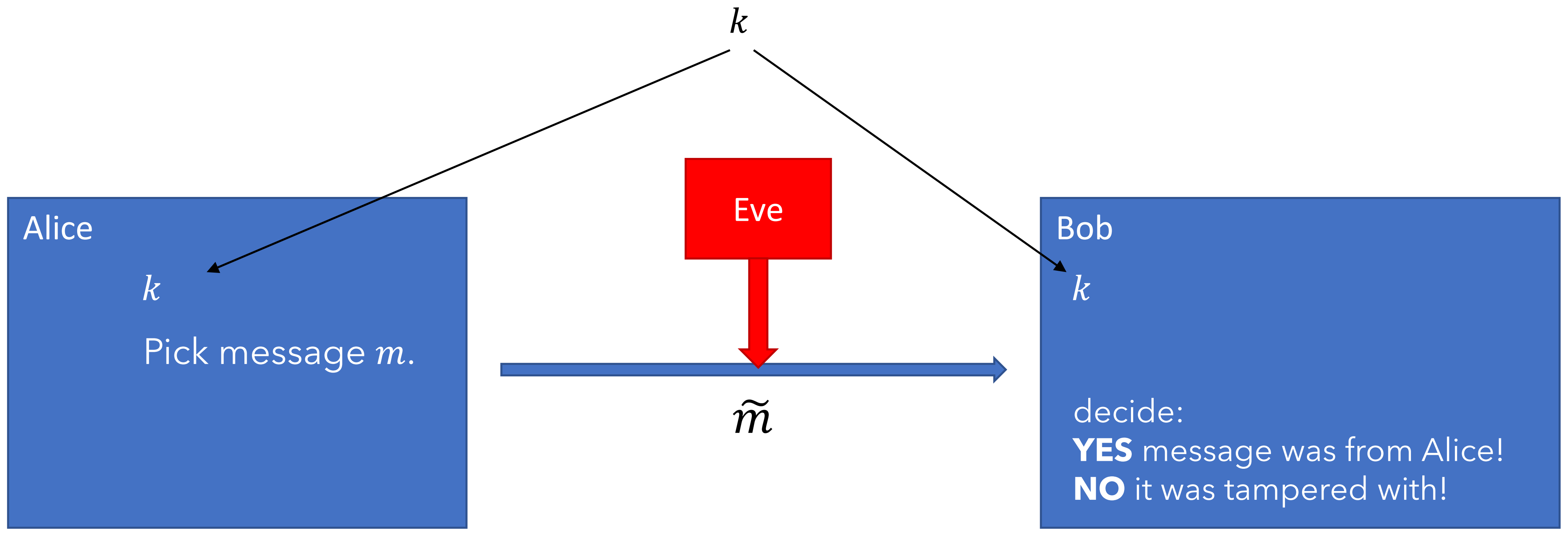


New: Eve can always guess $\tilde{m}$!

- since at least one string must get accepted, this gives her success probability $2^{-|\tilde{m}|}$.
- compare: in secrecy, we could get *perfect* security. Here we have to pay, at least a little.

AUTHENTICATION

In a bit more detail.



How should Alice and Bob proceed?

For secrecy, they used *encryption schemes*.

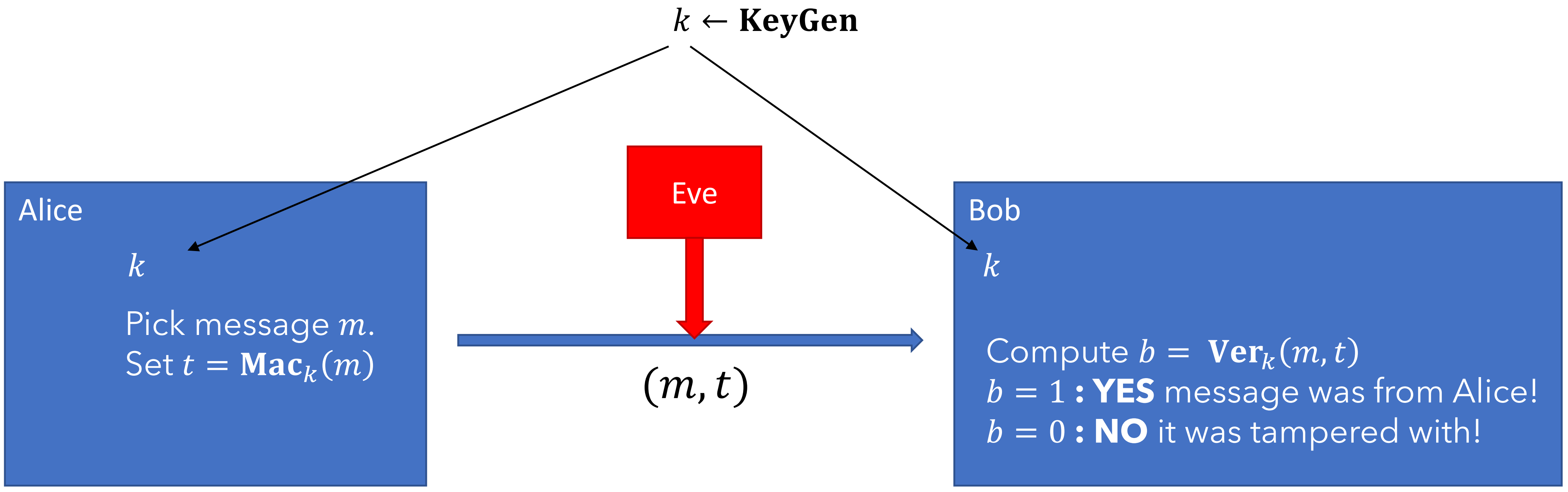
For authentication, it will be *message authentication codes (MACs).*

MESSAGE AUTHENTICATION CODES

Message authentication code (MAC):

- generate key: $k \leftarrow \mathbf{KeyGen}$
- generate **tag**: $t \leftarrow \mathbf{Mac}_k(m)$
- verify (message, tag) pair: $b \leftarrow \mathbf{Ver}_k(m, t)$ [$b = 1$ (valid) or $b = 0$ (invalid)]

Correctness:
$\mathbf{Ver}_k(m, \mathbf{Mac}_k(m)) = 1.$


$k \leftarrow \mathbf{KeyGen}$
Alice
k
Pick message m.
Set $t = \mathbf{Mac}_k(m)$
Eve
(m, t)
Bob
k
Compute $b = \mathbf{Ver}_k(m, t)$
$b = 1$: **YES** message was from Alice!
$b = 0$: **NO** it was tampered with!

MESSAGE AUTHENTICATION CODES

How to use a MAC.

First, generate a key and share it: $k \leftarrow \mathbf{KeyGen}$;

To send a message:

1. pick message m you want to send;
2. compute tag $t \leftarrow \mathbf{Mac}_k(m)$;
3. send message together with tag: (m, t).

To receive an authenticated message (m, t):

1. verify the pair: $b \leftarrow \mathbf{Ver}_k(m, t)$;
2. if $b = 1$, accept the message; otherwise reject it.

Important observations:

- This process *does not* "check that the message came from Alice!"
- *All it does* is check that the tag is consistent with the MAC defined by the shared key!
- In order for that to mean "the message came from Alice", the MAC function needs to have some special properties.
- It needs to be **unpredictable:** the only way to "predict" tags is to have Alice's key!

MESSAGE AUTHENTICATION CODES

Formal definition.

Definition. A message authentication code (MAC) is a triple of PPT algorithms:

- (key generation) $\mathbf{KeyGen}$: on input 1^n, outputs a key $k \in \{0,1\}^n$;
- (tag generation) $\mathbf{Mac}$: on input a key k and message $m \in \{0,1\}^*$, outputs a tag $\mathbf{Mac}_k(m)$;
- (verification) $\mathbf{Ver}$: on input a key k and a message-tag pair (m, t), outputs 1 (valid) or 0 (invalid);

satisfying *correctness*: for all k and all m, $\mathbf{Ver}_k\big(m, \mathbf{Mac}_k(m)\big) = 1$.

Canonical verification.

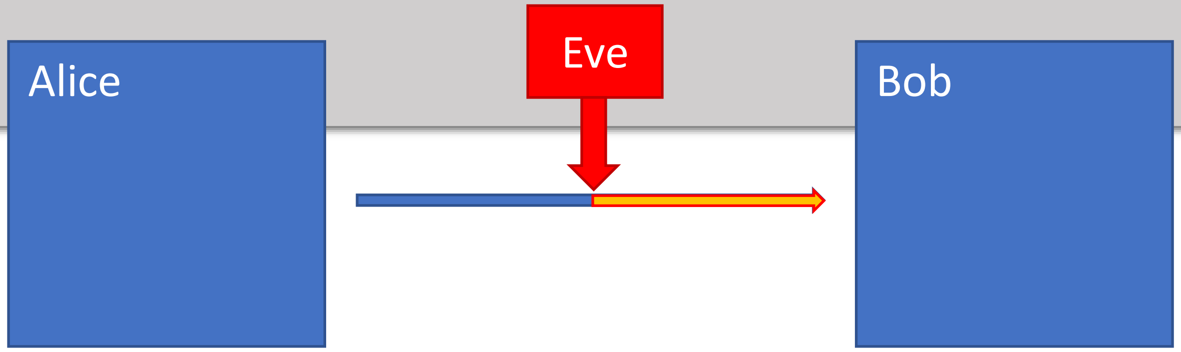
If $\mathbf{Mac}_k$ is a deterministic algorithm…

Then there's an optimal way to verify:

1. On input (m, t), re-compute the tag $t' = \mathbf{Mac}_k(m)$;
2. If $t = t'$, accept (output 1); otherwise reject (output 0).

So then we can just think about a MAC as a pair ($\mathbf{KeyGen}$, $\mathbf{Mac}$).

SECURE AUTHENTICATION



How to define security for MACs?

Unforgeability.

1. it should be impossible to predict tags, for any message
2. this should remain true even if adversary sees some *valid (message, tag) pairs!*

Why the latter?

- second point above sounds a bit like CPA from secrecy;
- but we had schemes where we didn't care about CPA. Why have it right away?
- **new:** even if we send only a single transmission, adversary can read it, and then replace it!
- so we have to account for that.

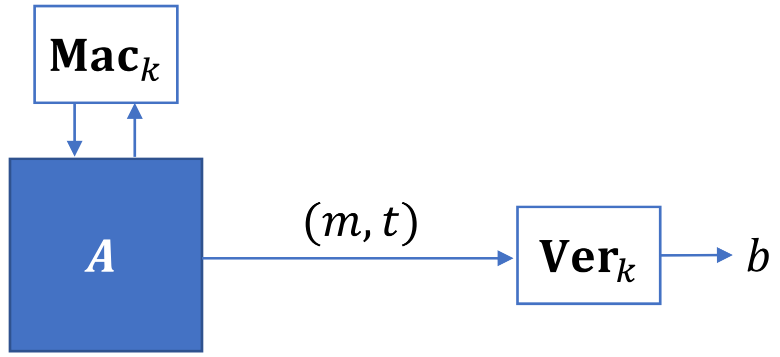
As it turns out, we can for free get a little more: *we can let adversary pick the message*.

UNFORGEABILITY

How to define security for MACs? Unforgeability.

Let's use a game: $\mathrm{MacForge}(\Pi, n)$, where Π is a MAC and n the security parameter.

1. A key is sampled: $k \leftarrow \mathbf{KeyGen}(1^{\mathrm{n}})$;
2. Adversary $\boldsymbol{A}$ is given oracle access to $\mathbf{Mac}_k$;
3. $\boldsymbol{A}$ outputs a pair (m, t); set $b = \mathbf{Ver}_k(m, t)$;



We say $\boldsymbol{A}$ wins the experiment if:

- $b = 1$ (valid), **and**
- m is not in the set of queries $\boldsymbol{A}$ made to the oracle.

The latter condition is to prevent trivial "replay" attacks.

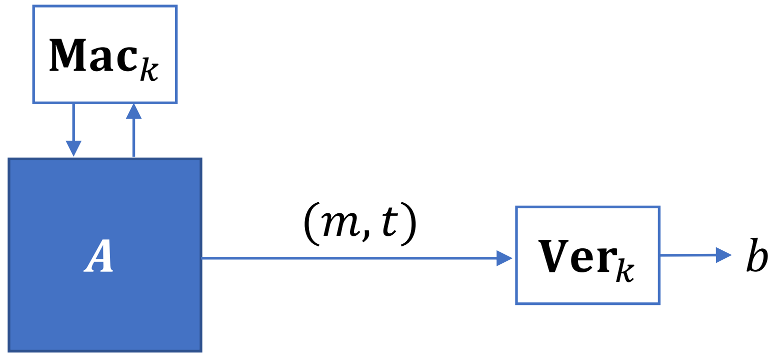
(In the real world, replay attacks are an actual problem, and one also needs to deal with them.)

UNFORGEABILITY

How to define security for MACs? Unforgeability.

Let's use a game: $\mathrm{MacForge}(\Pi, n)$, where Π is a MAC and n the security parameter.

1. A key is sampled: $k \leftarrow \mathbf{KeyGen}(1^{\mathrm{n}})$;
2. Adversary $\boldsymbol{A}$ is given oracle access to $\mathbf{Mac}_k$;
3. $\boldsymbol{A}$ outputs a pair (m, t); set $b = \mathbf{Ver}_k(m, t)$;



We say $\boldsymbol{A}$ wins the experiment if:

- $b = 1$ (valid), **<u>and</u>**
- m is not in the set of queries $\boldsymbol{A}$ made to the oracle.

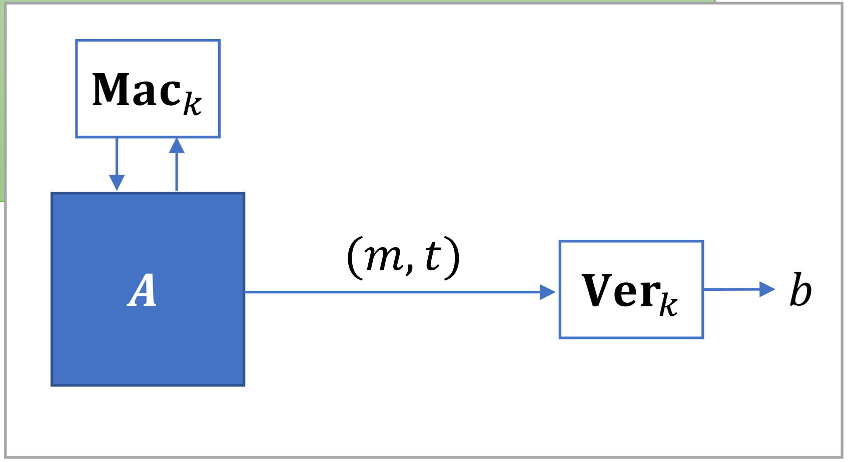
Definition. A message authentication code Π is **existentially unforgeable under chosen message attack (EUF-CMA)** if, for every PPT adversary $\boldsymbol{A}$,

$$\Pr[\boldsymbol{A} \text{ wins } \mathrm{MacForge}(\Pi, n)] \leq \mathrm{negl}(n).$$

UNFORGEABILITY

Definition. A message authentication code Π is **existentially unforgeable under chosen message attack (EUF-CMA)** if, for every PPT adversary $\boldsymbol{A}$,

$$\Pr[\boldsymbol{A} \text{ wins MacForge}(\Pi, n)] \leq \text{negl}(n).$$



EUF: existential unforgeability

- "existential" means that forging on *any fresh message*…
- … is considered a break.
- could define weaker notion: (i.) $\boldsymbol{A}$ declares message m, (ii.) $\boldsymbol{A}$ receives oracle, (iii.) $\boldsymbol{A}$ forges a tag for m.
- but we want stronger security, and (as we will see) we can achieve it.

CMA: chosen message attack

- indicates that adversary has oracle access to $\mathbf{Mac}_k$ and can query it on messages of their choice;
- can limit number of queries to some number q: we call that q-**EUF-CMA**.
- clearly, **EUF-CMA** implies q-**EUF-CMA** for all polynomials q.

CONSTRUCTING SECURE MACs

How do we construct secure MACs?

Let's consider 1-**EUF-CMA** : adversary gets only one (m, t) pair.

First, let's see why OTP is a bad MAC, even for this.

Set $f_k(m) = m \oplus k$.

Good: for any m, and all t, $\Pr_k[f_k(m) = t] = 1/2^n$.

… in other words, *for a single message*, the tag distribution is uniform!

Bad: if you know $(m, f_k(m))$, it's trivial to determine $k = m \oplus f_k(m)$…

… and then you know $f_k(m')$ for all m'!

… i.e., the tag distribution becomes deterministic!

Trivial attack: use your one $\mathbf{Mac}_k$ query to learn k, then forge wherever you want.

CONSTRUCTING SECURE MACs

How do we construct secure MACs?

Let's consider 1-**EUF-CMA** : adversary gets only one (m, t) pair.

- what we want: for any **pair** of messages (m, m'), the tag distribution is uniform.
- that way, adversary can query however they want (i.e., on any m)…
- … and that should tell them *nothing* about the tag for any other m'.

Formally:

Definition. A keyed function family $f: K \times M \to T$ is **pairwise independent** if, for every $m \neq m'$ in M and all t, t' in T, we have

$$\Pr_{k \in K}[f_k(m) = t \wedge f_k(m') = t'] = \frac{1}{|T|^2}$$

Suppose we had one of these.

Can we construct a secure (one-time) MAC?

CONSTRUCTING SECURE MACs

Construction (Carter-Wegman). Let $f: K \times M \to T$ be a pairwise-independent function family. Define a MAC (with canonical verification) as follows:

- $\mathbf{KeyGen}$: output uniformly random $k \leftarrow K$;
- $\mathbf{Mac}$: on input a key k and message $m \in M$, output tag $f_k(m)$.

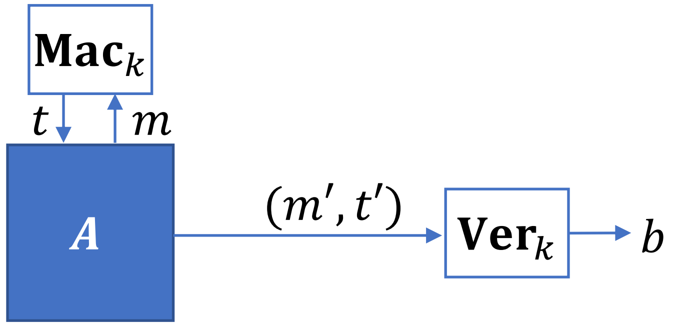
Theorem. The Carter-Wegman MAC with a pairwise-independent function is 1-**EUF-CMA** *against arbitrary adversaries.*

Proof.

- info-theoretic setting: $\boldsymbol{A}$ is all-powerful
- can assume $\boldsymbol{A}$ is deterministic (if not, there exists a *stronger* $\boldsymbol{A}'$ which runs $\boldsymbol{A}$ with best possible coins.)
- the initial query message m is thus fixed;
- and the claimed forgery m' is a fixed function $\boldsymbol{A}(t)$ of the query response t.

$$\begin{aligned}\Pr[\boldsymbol{A} \text{ wins}] &= \textstyle\sum_{t\in T} \Pr[\boldsymbol{A} \text{ wins} \wedge f_k(m) = t] \\ &= \textstyle\sum_{t\in T} \Pr[f_k(m') = \boldsymbol{A}(t) \wedge f_k(m) = t] \\ &= \textstyle\sum_{t\in T} \frac{1}{|T|^2} = \frac{1}{|T|}.\end{aligned}$$

We used pairwise independence in line 2 → line 3. $\square$



CONSTRUCTING SECURE MACs

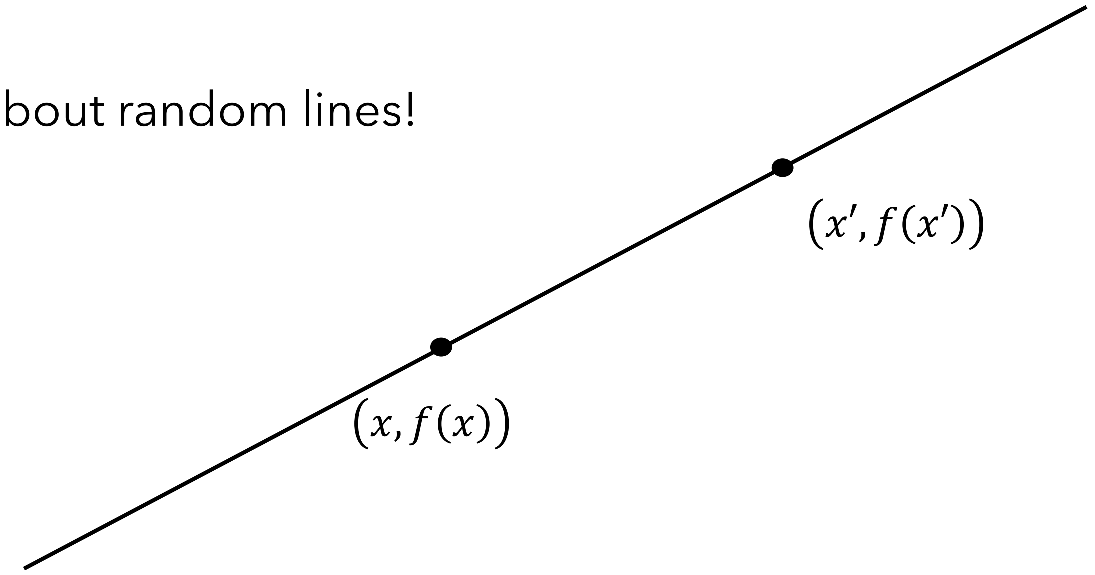
This is great and all…

but do pairwise-independent functions even exist?

What do they need, intuitively?

- even if you know an input-output pair $(x, f(x))$…
- you don't know anything about any other pair;
- moreover, the values are random.

Think about random lines!



CONSTRUCTING SECURE MACs

Pairwise-independent functions: random lines in $\mathbb{Z}_p$.

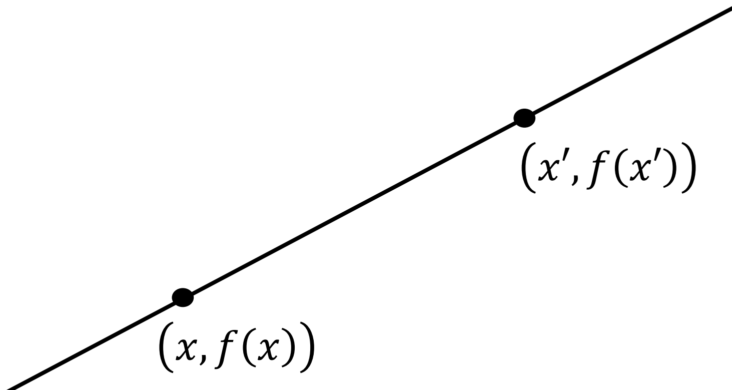
- Input and output spaces: $\mathbb{Z}_p = \{0,1,2,\dots,p-1\}$ for a **prime** p.
- Key space: $\mathbb{Z}_p \times \mathbb{Z}_p$.
- All arithmetic will be modulo p.
- Recall: since p is a prime, we have multiplicative inverses (and can easily compute them.)

For any pair $(a,b) \in \mathbb{Z}_p \times \mathbb{Z}_p$, define

$$f_{a,b}(x) \coloneqq a \cdot x + b$$

Prove pairwise independence:

- given: $x \neq x'$ and arbitrary y, y' in $\mathbb{Z}_p$;
- easy: let $a = (y - y') \cdot (x - x')^{-1}$ and $b = y - a \cdot x$;
- it follows that there is a *unique* line $f_{a,b}$ through the given points;
- if we pick a uniformly random key $(a,b) \in \mathbb{Z}_p \times \mathbb{Z}_p \dots$
- … the probability of sampling that unique line is $1/p^2$. $\square$



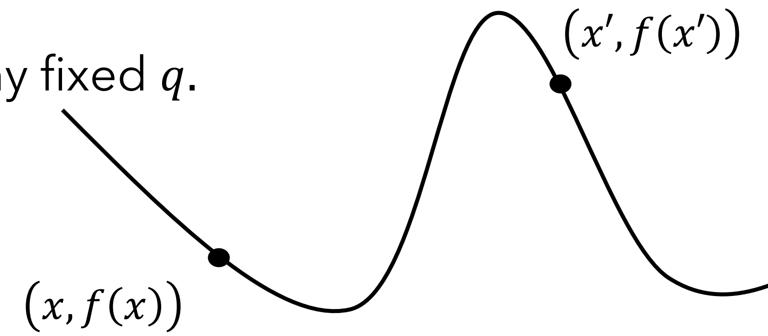
CONSTRUCTING SECURE MACs

Great!

- we constructed a one-time secure MAC;
- we can use it to securely authenticate **one** message!
- if we used it more than once, forging becomes possible (think about the line.)

Can we get more?

- Sure! let's take random *polynomials* over $\mathbb{Z}_p$;
- Keys get a bit bigger, but now you need *degree-many* points to learn the function;
- There's plenty to check…
- … but this does give us information-theoretically secure q-time MACs for any fixed q.



CONSTRUCTING SECURE MACs

Problem:

What if I don't know *in advance* how many messages I want to authenticate?

- random polynomials don't work;
- degree is fixed at key generation time;
- so if adversary (or honest party) authenticates more than degree-many messages…
- … the polynomial is now completely known!

What do we need?

- a function for computing tags, which is unpredictable…
- which remains unpredictable *on any input*…
- … regardless of how many times it has been used previously!

Where have we seen this before?

SIMPLE PRF MAC

Construction (PRF MAC). Let $\boldsymbol{F}\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^\ell$ be a PRF. Define a MAC (with canonical verification) as follows:

- $\mathbf{KeyGen}$: output uniformly random $k \leftarrow \{0,1\}^n$;
- $\mathbf{Mac}$: on input a key k and message $m \in \{0,1\}^m$, output tag $\boldsymbol{F}_k(m)$.

Notes.

- messages are of fixed length;
- tags are of length $\ell(n)$; we can pick this however we want (by selecting the right PRF)…
- … but careful: recall trivial tag-guessing attack, which succeeds with probability $2^{-\ell(n)}$.

Proof.

- similar to IND-CPA proof:

1. show that a scheme with a *perfectly random* function is statistically unforgeable;
2. then show that a forger for the PRF MAC would imply a distinguisher for the PRF.