CS 442
Introduction to Cryptography

Lecture 15: Message Integrity

Instructor: Aarushi Goel

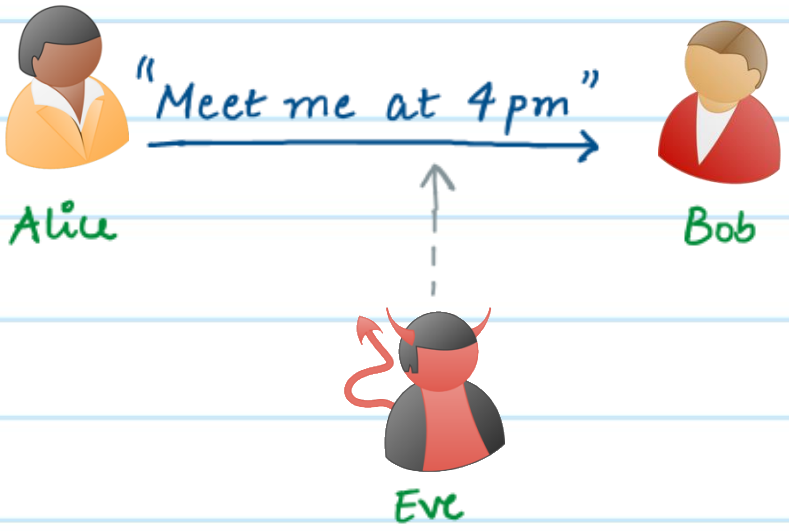
Spring 2026

Agenda

* Message Authentication Codes (MACs)
* Construction of One-time MACs
* (Many-time) MACs.

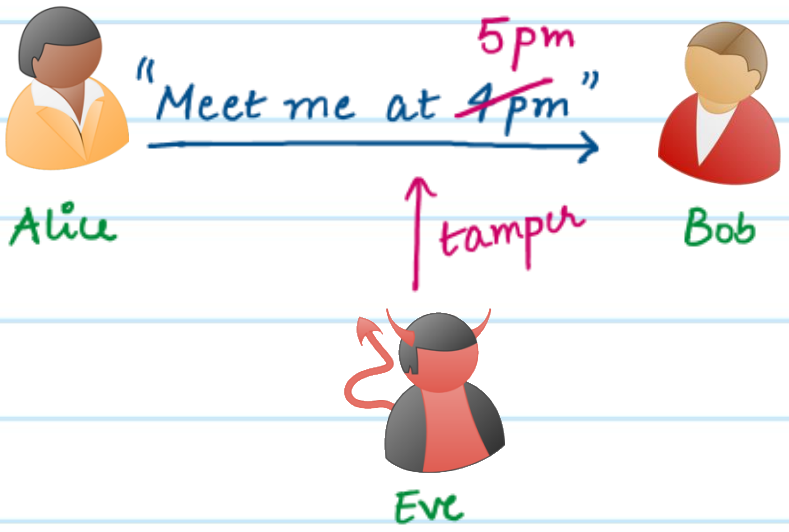
Private Communication

* So far we have considered the problem of private communication, where Alice wants to send a secret message to Bob, while making sure Eve who might be eavesdropping on their communication.
* Encryption helps with this.
* But an Eve who is only listening to a private communication, is still a pretty "well-behaved" adversary.

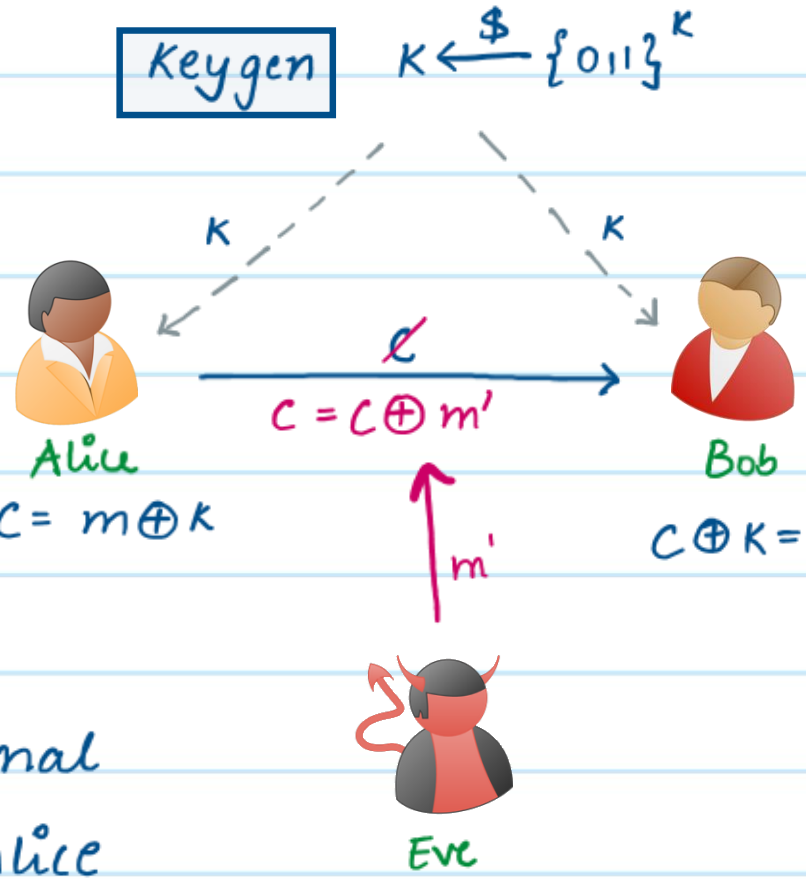


Tamper-Proof Communication

* In reality, adversaries could be much worse.
* What if the adversary can not only evesdrop on a communication channel, but also *tamper* with messages on this communication channel.
* Does encryption naturally help prevent this? No!



Q Why doesn't one-time pad encryption make the underlying message tamper-proof?



* Eve can change the ciphertext $C \to C \oplus m'$ for any m' of its choice.
* Bob then ends up decrypting $m \oplus m'$ instead of the original message m that Alice intended to send.
* Bob has no way to tell whether this is the original message or tampered message.
* Note that Eve does not even need to know the key or message m to launch this attack.

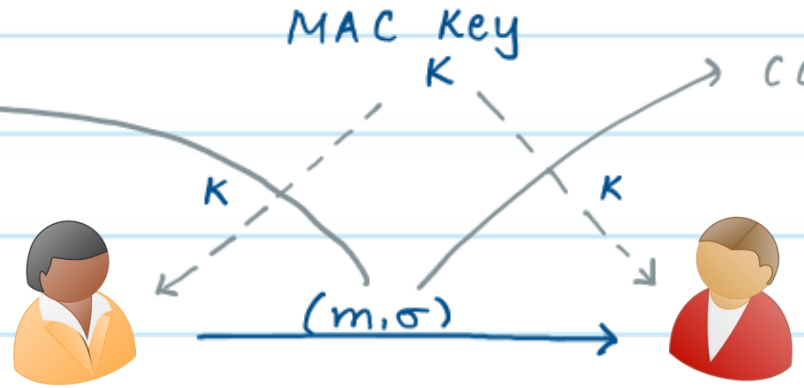
Message Integrity in the Non-Digital World

* In the physical world, we use signatures on documents to ensure unauthorised entities cannot tamper with them, without forging the signature, which is hard to do.
* We need something similar in the digital world.

Message Authentication Codes

* Digital Analogue of Physical Signatures.
* Alice (signer) signs a message m to produce a signature σ.
* Bob (verifier) can verify that σ is indeed generated for m.
* We will assume Alice and Bob share a key that is used for signing and verification.
* Adversary does not have this key.
* It should be impossible (or very hard) for an adversary to forge a valid signature with respect to this key on any other message.

Message Authentication Codes (MACs)



called signature or tag.

* Observe that our goal here is not to hide m. For now, we are just focusing on message integrity

Sign(m,K) → σ

Verify(m,σ,K) → yes/no

Adversary

* If the adversary tries of change m → m', they will also have to find a corresponding σ', such that Verify(m,σ,K) → yes. This should be hard if they don't have K.

* We will later see how to combine Encryption and MACs to ensure integrity of encrypted messages.

One-time Message Authentication Codes (MACs)

→ each key will only be used to sign one message.

Definition: A one-time message athentication code scheme with message space $\mathcal{M}$, key space $\mathcal{K}$ and signature/tag space $\mathcal{S}$, comprises of the following algorithms:

* $\text{Keygen} \to K$: This algorithm samples a key $K \xleftarrow{\$} \mathcal{K}$.
* $\text{Sign}(m, K) \to \sigma$: On input a message $m \in \mathcal{M}$ & key $K \in \mathcal{K}$, it outputs a signature $\sigma \in \mathcal{S}$.
* $\text{Verify}(m, \sigma, K) \to b$: On input a message $m \in \mathcal{M}$, key $K \in \mathcal{K}$ and signature $\sigma \in \mathcal{S}$, it outputs a bit $b \in \{0,1\}$ (where 1 means yes, 0 means no).

These algorithms must satisfy the following:

→ Correctness: $\forall K \in \mathcal{K}$, $m \in \mathcal{M}$, it holds that $\Pr[\text{Verify}(m, \text{Sign}(m, K), K) = 1] = 1$

→ Unforgeability: For all PPT adversaries, there exists a negligible function $\nu(\cdot)$, such that the following holds in the game below:

$$\Pr[\text{Adv wins}] \leq \nu(|K|)$$

One-time message Authentication game:

Adversary

Challenger

KeyGen → K

$m \in \mathcal{M}$ →

Sign(m, K) → σ

← σ

(m', σ') →

Adv wins if:

① $m' \neq m$

② Verify$(m', \sigma', K) = 1$

Constructing One-time MACs.

Let $\mathcal{M} = \mathbb{Z}_p$ ($\{0, 1, \ldots, p-1\}$, p is prime), $\mathcal{K} = \mathbb{Z}_p \times \mathbb{Z}_p$, $\mathcal{S} = \mathbb{Z}_p$. Recall $(\mathbb{Z}_p, +\bmod p, \times \bmod p)$ is a field.

* Keygen: Sample $k_1 \xleftarrow{\$} \mathbb{Z}_p$ and $k_2 \xleftarrow{\$} \mathbb{Z}_p$. Set $K = (k_1, k_2)$
* Sign(m, K): Parse $K = (k_1, k_2)$.

  $\sigma = (k_1 \cdot m + k_2) \bmod p$.
* Verify(m, σ, K): Parse $K = (k_1, k_2)$

$$b = \begin{cases} 1, & \text{if } (k_1 \cdot m + k_2) \bmod p = \sigma \\ 0, & \text{otherwise} \end{cases}$$

Q Does this scheme satisfy correctness & unforgeability?

* Correctness: We need to show that $\forall m \in \mathbb{Z}_p$, $\forall K \in \mathbb{Z}_p \times \mathbb{Z}_p$,

$$Pr[Verify(m, Sign(m,K), K) = 1] = 1$$

Proof:

$$\begin{aligned} & Verify(m, Sign(m,K), K) \\ = \; & Verify(m, Sign(m, (K_1, K_2)), (K_1, K_2)) \\ = \; & Verify(m, K_1 \cdot m + K_2 \bmod p, (K_1, K_2)) \\ = \; & 1. \end{aligned}$$

This holds with probability 1.

* **Unforgeability:** We need to show that the adversary can win in the one-time message authentication game with at most negligible probability.

Proof: The adversary first recieves σ on a message m of its choice.

where $\sigma = (K_1 \cdot m + K_2) \bmod p$.

To win, it must send a valid σ' on another message $m' = m$.

$\Rightarrow \sigma' = (K_1 \cdot m' + K_2) \bmod p$ (if σ' is valid)

$\Rightarrow K_2 = (K_1 \cdot m' - \sigma') \bmod p$ (we know that $K_2 = (K_1 \cdot m - \sigma) \bmod p$.

$\Rightarrow (K_1 \cdot m - \sigma) \bmod p = (K_1 \cdot m' - \sigma') \bmod p$

$\Rightarrow K_1 (m - m') = (\sigma - \sigma') \bmod p$

$\Rightarrow K_1 = \frac{(\sigma - \sigma')}{(m - m')} \bmod p$

But since K_1 was chosen at random,

$$Pr\left[K_1 = \frac{(\sigma - \sigma')}{(m - m')} \bmod p\right] = \frac{1}{p}$$

In other words, the probability that adversary can find m, m' such that $K_1 = \frac{(\sigma - \sigma')}{(m - m')} \bmod p$ is $\frac{1}{p}$.

* If p is very large (eg. 2^{128}), then this probability is very small (or negligible) and we can safely assume that no adv will be able to win except with negligible probability.

Should this be "no" or "no PPT"?

* But what if p is not large enough, i.e., our messages come from a small space?

Constructing One-time MACs for Small Message Spaces

Let $\mathcal{M} = \mathbb{Z}_p$. Let $n = 128$. $\mathcal{K} = \mathbb{Z}_p^n \times \mathbb{Z}_p^n$, $\mathcal{S} = \mathbb{Z}_p^n$

* Keygen: $\forall i \in [n]$, sample $k_1^i \xleftarrow{\$} \mathbb{Z}_p$ and $k_2^i \xleftarrow{\$} \mathbb{Z}_p$. Set $K = (\{k_1^i\}_{i\in[n]}, \{k_2^i\}_{i\in[n]})$
* Sign(m, K): $\forall i \in [n]: \sigma_i = (k_1^i \cdot m + k_2^i) \bmod p$.

  Set $\sigma = (\sigma_1, \ldots, \sigma_n)$
* Verify(m, σ, K): $\forall i \in [n]$:

$$b_i = \begin{cases} 1, & \text{if } (k_1 \cdot m + k_2) \bmod p = \sigma \\ 0, & \text{otherwise} \end{cases}$$

Set $b = b_1 \wedge b_2 \wedge \ldots \wedge b_n$ (AND)

Q Is this scheme unforgeable? Why?

MACing Many Messages using the same Key.

Q Can we use these schemes to sign more than one message using the same key?

A No. If the adversary sees two signatures in this construction, they will be able recover the Key.

eg Let's assume they see (m_1, σ_1) & (m_2, σ_2).

where $\sigma_1 = (K_1 \cdot m_1 + K_2) \bmod p$ & $\sigma_2 = (K_1 \cdot m_2 + K_2) \bmod p$.

* The adv can compute $K_1 = \left(\frac{\sigma_2 - \sigma_1}{m_2 - m_1}\right) \bmod p$ and then compute $K_2 = (\sigma_1 - K_1 \cdot m_1) \bmod p$.

* Once it recovers (K_1, K_2), it can use this Key to forge signatures on any messages of its choice.

MACing Many Messages using the same Key

(Many time)

* Message Authentication Codes that allow the key to be reused for safely signing many messages are defined exactly as one-time MACs, except that message authentication game for unforgeability now looks like the following:

Many-time message Authentication game:

Adversary

Challenger

KeyGen → K

polynomial number of queries

$m_i \in M$

σ_i

Sign $(m_i, K) \rightarrow \sigma_i$

(m, σ)

Adv wins if:

① $\forall i \quad m \neq m_i$

② Verify $(m, \sigma, K) = 1$

Construction of Many-time MAC Scheme

Let $\{f_K\}$ be a family of PRFs.

* Keygen: Sample a PRF Key K.
* Sign(m, K) : $\sigma = f_K(m)$
* Verify(m, K) : $b = \begin{cases} 1, & \text{if } \sigma = f_K(m) \\ 0, & \text{otherwise} \end{cases}$

Think: why is this a good scheme?