CS716
Data Security

Week 4:

- Building CPA-Secure Encryption Schemes
- CCA-Security (definition)
- Message Authentication Codes

Fall 2025

Assoc. Prof. Hisham Dahshan

hishamdahshan@aast.edu

Our Goal Today

- Building CPA-Secure Encryption Schemes
- CCA-Security (definition)
- Message Authentication Codes

Chosen-Plaintext Attacks

- Model ability of adversary to control or influence what the honest parties encrypt.
- During World War 2 the British placed mines at specific locations, knowing that the Germans, upon finding the mines, would encrypt the location and send them back to headquarters. The encrypted messages helped cryptanalysts at Bletchley Park to break the German encryption scheme.

Bletchley Park

Chosen-Plaintext Attacks

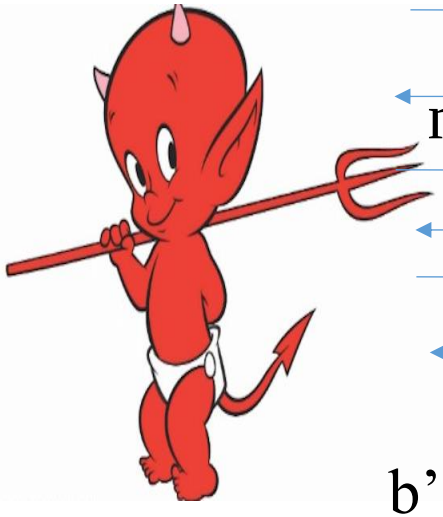
- Model ability of adversary to control or influence what the honest parties encrypt.
- Battle of Midway (WWII). US Navy cryptanalysts intercept and encrypted message which they are able to partially decode (May 1942).
  - The message stated that the Japanese were planning an attack on AF?
  - Cryptanalysts could not decode ciphertext fragment AF.
  - Best Guess: AF = “Midway Island.”

The United States was aware that the Japanese were planning an attack in the Pacific (on a location the Japanese code-named “AF”) because Navy cryptanalysts had begun breaking Japanese communication codes in early 1942. The attack location and time were confirmed when the American base at Midway sent out a false message that it was short of fresh water. Japan then sent a message that “AF” was short of fresh water, confirming that the location for the attack was the base at Midway. Station Hypo (where the cryptanalysts were based in Hawaii) was able to also give the date (June 4 or 5) and the order of battle of the Imperial Japanese Navy.

Multiple Message Security and CPA-Attacks

- Multiple Message Security
  - Attacker must select all messages at the same time.
  - Significant Limitation!
- In the WWII attacks cryptanalysts selected the message adaptively
  - Selected message(s) to encrypt *after* observing target ciphertext

CPA-Security (Single Message)



m_0, m_1

$c = Enc_K(m_b)$

m_2

$c_2 = Enc_K(m_2)$

m_3

$c_3 = Enc_K(m_3)$

$\vdots$

b’

$\forall PPT\ A\ \exists \mu$ (negligible) s. t

Random bit b

K ← Gen(1^n)

$$\Pr[A\ \ Guesses\ b' = b] \leq \frac{1}{2} + \mu(n)$$

CPA-Security (Single Message)

Formally, let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ denote the encryption scheme, and define a random variable $\text{PrivK}^{\text{cpa}}_{\text{A},\Pi}(1^n)$

$$\text{PrivK}^{\text{cpa}}_{\text{A},\Pi}(1^n) = \begin{cases} 1 & \text{if } b = b' \\ 0 & \text{otherwise} \end{cases}$$

Π *has* indistinguishable encryptions under a chosen plaintext attack if for all PPT adversaries A, there is a negligible function μ such that

$$\Pr\left[\text{PrivK}^{\text{cpa}}_{\text{A},\Pi}(1^n) = 1\right] \leq \frac{1}{2} + \mu(n)$$

CPA-Security

Theorem: An encryption scheme $\Pi = (Gen, Enc, Dec)$ that is CPA-Secure for single encryptions is also CPA-secure for multiple encryptions.

- We will simply say CPA-security for simplicity

- To show CPA-Security it suffices to show CPA-security for single encryptions.

CCA-Security

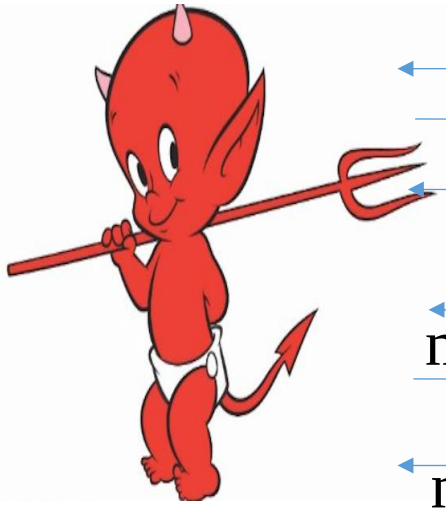
Chosen Ciphertext Attacks

- Sometimes an attacker has ability to obtain (partial) decryptions of ciphertexts of its choice.
- CPA-Security does not model this ability.
- In a chosen-ciphertext attack, the adversary has the ability not only to obtain encryptions of messages of its choice (as in a chosen plaintext attack), but also to obtain the decryption of ciphertexts of its choice.
- Formally, we give the adversary access to a decryption oracle in addition to an encryption.

Examples:

- An attacker may learn that a ciphertext corresponds to an ill-formed plaintext based on the reaction (e.g., server replies with “invalid message”).
- Monitor enemy behavior after receiving an encrypted message.
- **Authentication Protocol:** Send $Enc_k(\mathbf{r})$ to recipient who authenticates by responding with **r**.

CCA-Security (Ind-CCA2)



m_{-1}

c_{-2}

m_0, m_1

m_2

c_3

$c_4 = c$

b’

$c_{-1} = Enc_K(m_{-1})$

$m_{-2} = Dec_K(c_{-2})$

$c = Enc_K(m_b)$

$c_2 = Enc_K(m_2)$

$m_3 = Dec_K(c_3)$

“No Way!”

We could set $m_0 = m_{-1}$ or $m_1 = m_{-2}$ etc…

However, we could still flip 1 bit of c and ask challenger to decrypt

Random bit b

K = Gen(.)

CCA-Security $\left(PrivK_{A,\Pi}^{cca}(n)\right)$

1. Challenger generates a secret key k and a bit b
2. Adversary (A) is given oracle access to Enc_k and Dec_k
3. Adversary outputs m_0, m_1
4. Challenger sends the adversary $c=Enc_k(m_b)$.
5. Adversary maintains oracle access to Enc_k and Dec_k, however the adversary is not allowed to query $Dec_k(c)$. This restriction is necessary or else there is clearly no hope for any encryption scheme to satisfy the definition.
6. Eventually, Adversary outputs b'.

$$PrivK_{A,\Pi}^{cca}(n) = 1 \text{ if } b = b'; \text{otherwise } 0.$$

Definition 3.33: An encryption scheme Π is CCA-secure if for all PPT A there is a negligible function negl(n) such that

$$\mathbf{Pr}\left[PrivK_{A,\Pi}^{cca}(n) = 1\right] \leq \frac{1}{2} + negl(n)$$

CCA-Security

- CCA-Security is strictly stronger than CPA-Security
- **Note:** If a scheme has indistinguishable encryptions under one chosen-ciphertext attack then it has indistinguishable multiple encryptions under chosen-ciphertext attacks.
- None of the encryption schemes we have considered so far are CCA-Secure ☹

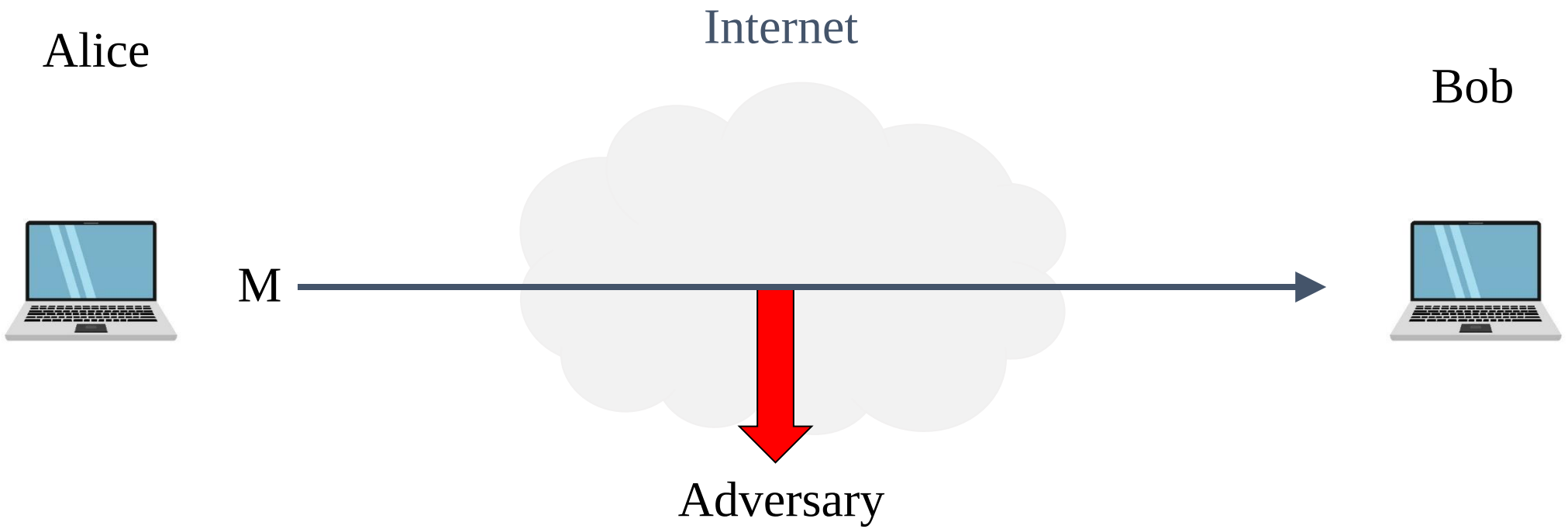
Common CCA-Secure Encryption Schemes

- **RSA-OAEP** - RSA with Optimal Asymmetric Encryption Padding
  - Provably CCA-secure in the random oracle model
- **Cramer-Shoup Cryptosystem**
  - First practical encryption scheme proven CCA-secure under standard assumptions
  - Based on the decisional Diffie-Hellman problem
- **Hybrid Encryption Schemes** that combine:
  - CCA-secure key encapsulation mechanisms (KEM)
  - Authenticated symmetric encryption
- **CCA-Secure Key Encapsulation Mechanisms (KEMs)**

1. **Kyber -** Post-quantum lattice-based KEM (NIST PQC standard)
2. **NTRU -** Lattice-based KEM
3. **Classic McEliece -** Code-based KEM

Message Authentication Codes

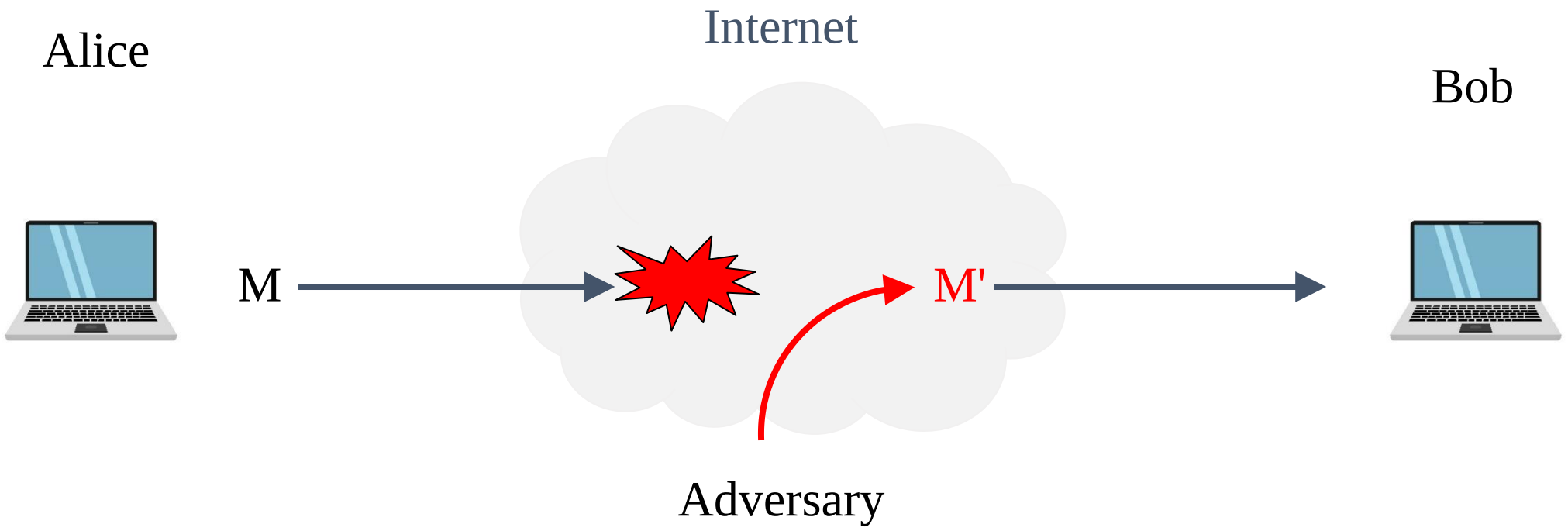
What is cryptography?



Security goals:

- **Data privacy:** adversary should not be able to read message M
- **Data integrity:** adversary should not be able to modify message M
- **Data authenticity:** message M really originated from Alice

What is cryptography?



Security goals:

- **Data privacy:** adversary should not be able to read message M
- **Data integrity:** adversary should not be able to modify message M
- **Data authenticity:** message M really originated from Alice

Basic goals of cryptography

	Message privacy	**Message integrity / authentication**
Symmetric keys	Symmetric encryption	Message authentication codes (MAC)
Asymmetric keys	Asymmetric encryption (a.k.a. public-key encryption)	Digital signatures

MAC Motivation

- Goal: *integrity*, not privacy

- Examples:
  - Protecting OS system files against tampering
  - Browser cookies stored by web servers
  - Control signals in network management

Encryption ≠ integrity

CTR. Enc(K,·)

0110100 1110 100101100110

⊕

"Send Bob $10"

1001010 1010 000000001010

1111110 0100 **0110**01101100

CTR. Dec(K,·)

1111110 0100 **0110**01101100

⊕

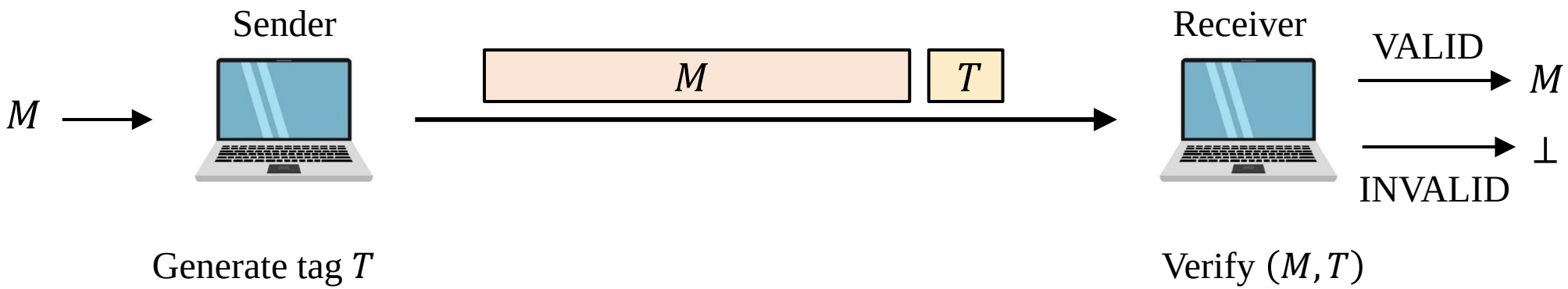
0110100 1110 100101100110

=

1001010 1010 **1111**00001010

"Send Bob **$3850**"

Message authentication – idea



Message Authentication Code Syntax

Definition 4.1: A message authentication code (MAC) consists of three algorithms

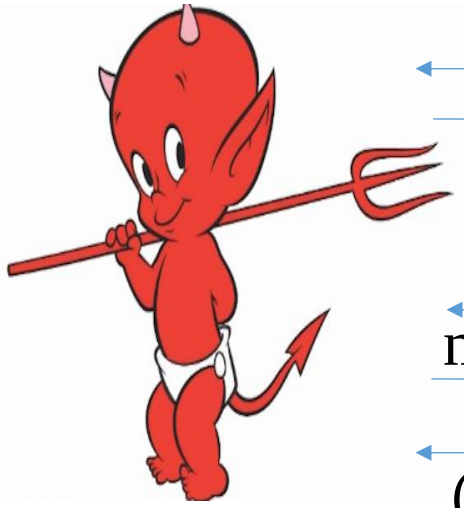
- $\text{Gen}(1^n; R)$ (Key-generation algorithm)
  - Input: security parameter 1^n (unary) and random bits R
  - Output: Secret key k $\in \mathcal{K}$
- $\text{Mac}_k(m; R)$ (Tag Generation algorithm)
  - Input: Secret key k $\in \mathcal{K}$ and message m $\in \mathcal{M}$ and random bits R
  - Output: a tag ***t***
- $\text{Vrfy}_k(m, \boldsymbol{t})$ (Verification algorithm)
  - Input: Secret key k $\in \mathcal{K}$, a message m and a tag ***t***
  - Output: a bit b (b=1 means "valid" and b=0 means "invalid")

$$\text{Vrfy}_k(m, \text{Mac}_k(m; R)) = 1$$

Message Authentication Code

- Two users who wish to communicate in an authenticated manner begin by generating and sharing a secret key ***k*** in advance of their communication.
- The sender of a message ***m*** computes ***t*** = MAC(***m***) and transmits (***m***, ***t***) to the receiver.
- Upon receiving (***m***, ***t***), the second party verifies whether ***t*** is a valid tag on the message ***m*** (with respect to the shared key) or not by running a verification algorithm Vrfy that takes as input the shared key ***k*** as well as a message ***m*** and a tag ***t*** and indicates whether the given tag is valid.

MAC Authentication Game ($\text{Macforge}_{A,\Pi}(n)$)



m_1

$t_1 = \text{Mac}_K(m_1)$

m_2

$t_2 = \text{Mac}_K(m_2)$

$\vdots$

m_q

$t_q = \text{Mac}_K(m_q)$

(m, t) s.t $m \notin \{m_1, \ldots, m_q\}$

$\text{Macforge}_{A,\Pi}(n) = \text{Vrfy}_k(m, t)$

K = Gen(.)

$\forall PPT\ A\ \exists \mu$ (negligible) s. t

$\Pr[\text{Macforge}_{A,\Pi}(n) = 1] \leq \mu(n)$

Discussion

- Is the definition too strong?
  - Attacker wins if he can forge any message
  - Does not necessarily attacker can forge a “meaningful message”
  - “Meaningful Message” is context dependent
  - Conservative Approach: Prove security against more powerful attacker
- Replay Attacks?
  - **$t=Mac_K$(“Pay Bob $1,000 from Alice’s bank account”)**
  - Alice cannot modify message to say $10,000, but…
  - She may try to replay it 10 times

Replay Attacks

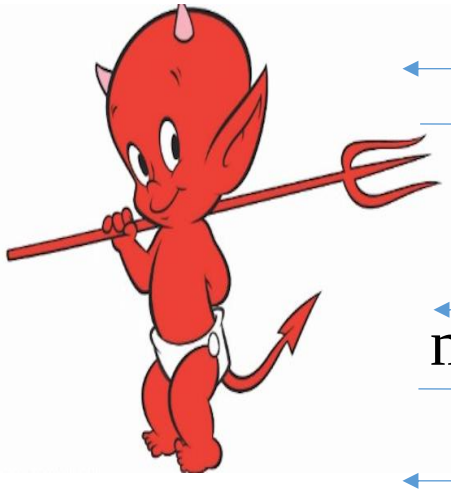
- MACs alone do not protect against replay attacks (they are stateless)

- Common Defenses:
  - Include Sequence Numbers in Messages (requires synchronized state)
    - Can be tricky over a lossy channel
  - Timestamp Messages
    - Double check timestamp before taking action

Strong MACs

- Previous game ensures attacker cannot generate a valid tag for a new message.
- However, attacker may be able to generate a second valid tag t' for a message m after observing (m, t)
- Strong MAC: attacker cannot generate second valid tag, even for a known message

Strong MAC Authentication ($\text{Macsforge}_{A,\Pi}(n)$)



m_1

$t_1 = \text{Mac}_K(m_1)$

m_2

$t_2 = \text{Mac}_K(m_2)$

$\vdots$

m_q

$t_q = \text{Mac}_K(m_q)$

(m, t) s.t $(m, t) \notin \{(m_1, t_1), \ldots, (m_q, t_q)\}$

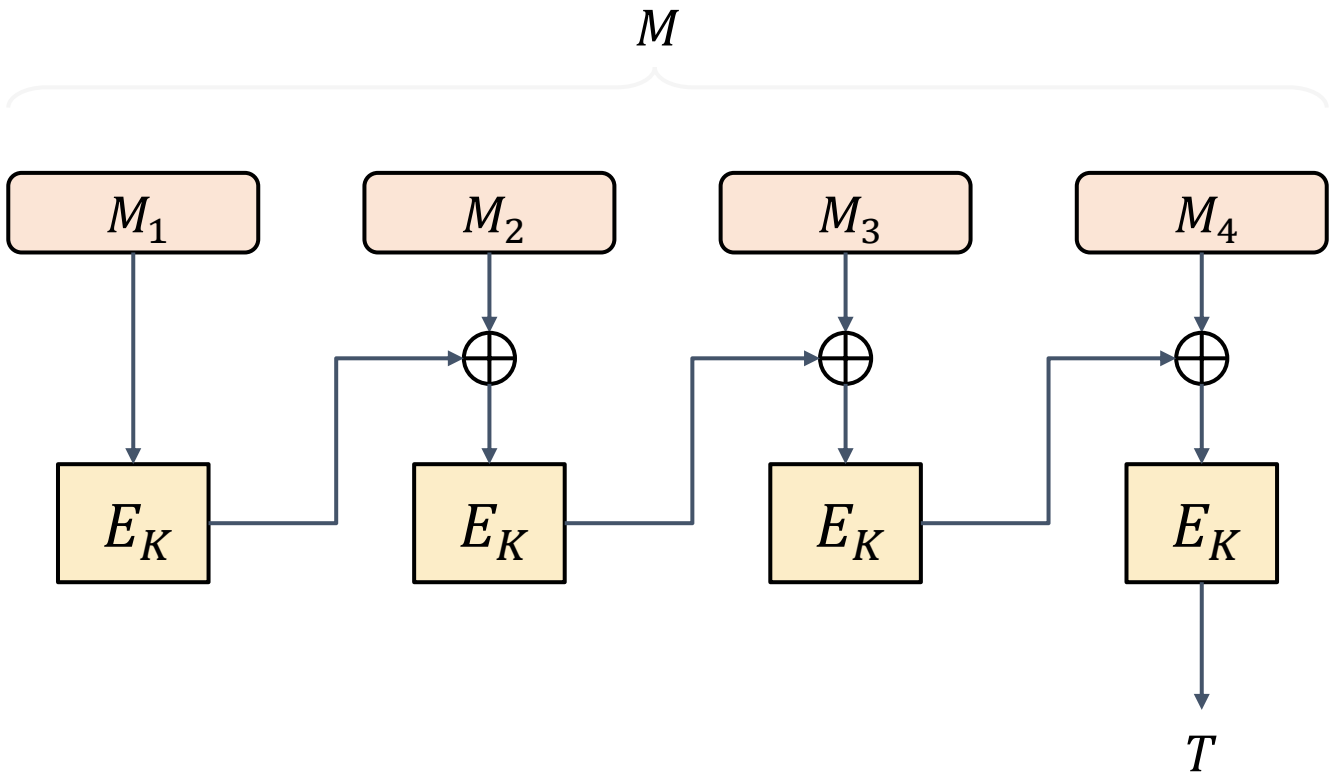
$\text{Macsforge}_{A,\Pi}(n) = \text{Vrfy}_k(m, t)$

K = Gen(.)

$\forall PPT\ A\ \exists \mu$ (negligible) s. t

$\Pr[\text{Macsforge}_{A,\Pi}(n) = 1] \leq \mu(n)$

CBC-MAC



Theorem: secure if all messages have the *same* length

Warning: not secure if messages have *different* lengths!

CBC MAC padding

For security, padding must be one-to-one (i.e., invertible)!

$$M_0 \neq M_1 \quad \Rightarrow \quad \text{pad}(M_0) \neq \text{pad}(M_1)$$

two distinct messages map to two distinct padded messages

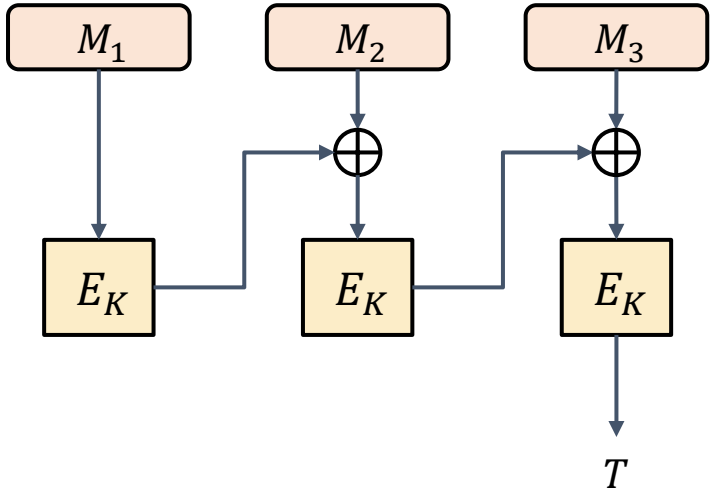
ISO: pad with "1000…00". Add new dummy block if needed.

- The "1" indicates beginning of pad.

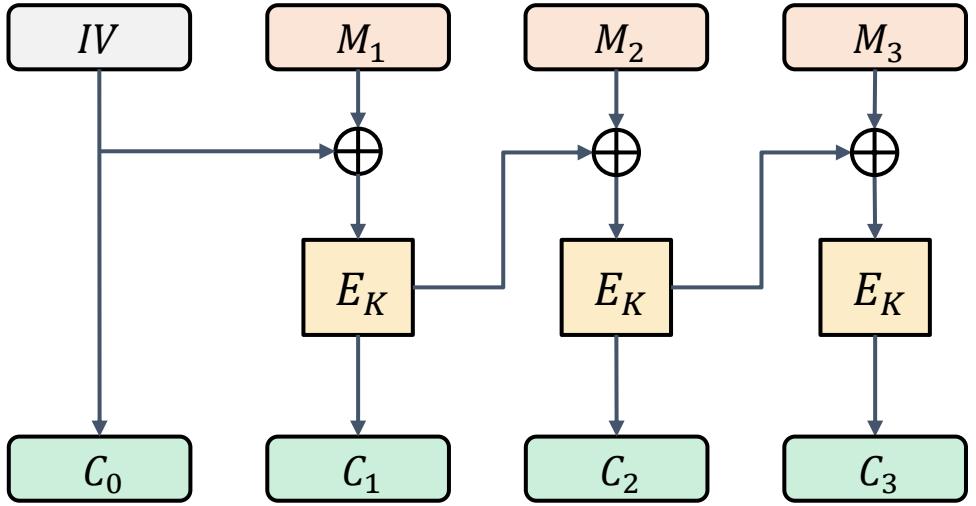
If m is same as block size, add 1 block pad for security

CBC-MAC vs. CBC-ENC

CBC-MAC



CBC-ENC



Warning 1: CBC-MAC is insecure *with* an IV!

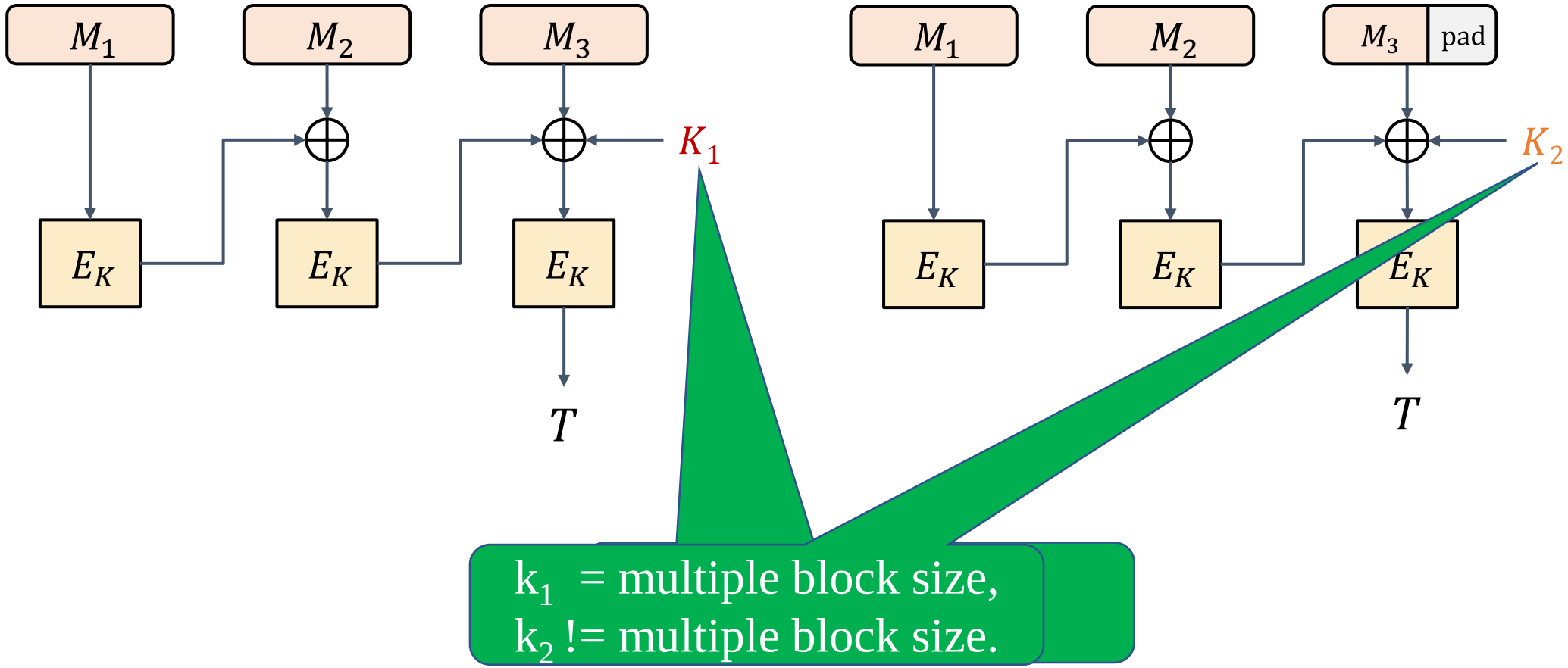
Warning 2: CBC$-ENC is insecure *without* an IV!

CMAC(NIST standard) a.k.a One-key MAC

CMAC: stands for Cipher-based Message Authentication Code.

Variant of CBC-MAC where key = (k, k_1, k_2)

No final encryption step, No dummy block (ambiguity resolved by use of k_1 or k_2)



CMAC common real-world applications:

1. **Secure Communication Protocols**
   - Ipsec
   - TLS/SSL
   - Wi-Fi Security (WPA2/WPA3)
2. **Financial and Payment Systems**
   - EMV Chip Cards (Credit/Debit Cards)
   - Secure Financial Messaging (e.g., SWIFT)

Questions?