Block Cipher Modes, Integrity, and Authenticated Encryption

ENCS4320 - Applied Cryptography

Dr. Ahmed I. A. Shawahna

Electrical and Computer Engineering Department

Birzeit University

Presentation Outline

- **More on Block Cipher**
- Block Cipher Modes
  - ECB
  - CBC
  - CTR
- Integrity
- Authenticated Encryption

More on Block Cipher

- Block cipher can be used for many other tasks:
  - Different types of block-based encryption schemes
  - Stream cipher
  - Pseudo-random generators (PRNG)
  - Hash function
  - Message authentication codes, which are also knowns as MACs
  - Key establishment protocol

Presentation Outline

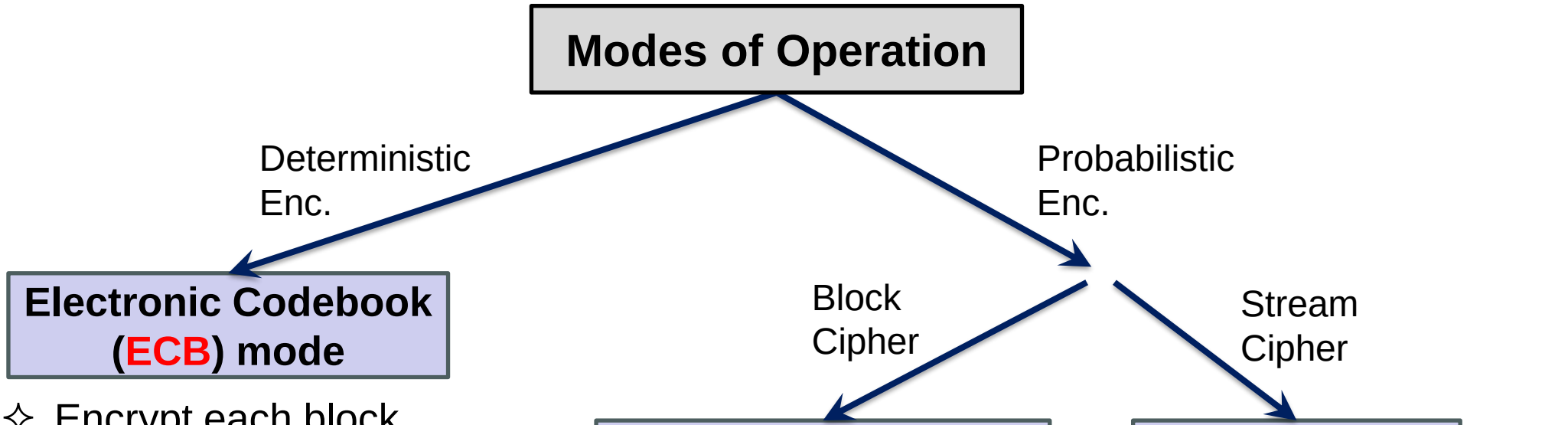
- More on Block Cipher
- **Block Cipher Modes**
  - ECB
  - CBC
  - CTR
- Integrity
- Authenticated Encryption

Multiple Blocks

- How to encrypt a long message (multiple blocks) using a primitive that only applies to n-bit blocks?
- Do we need a new key for each block?
  - As bad as (or worse than) a one-time pad!
- Encrypt each block independently?
- Make encryption depend on previous block?
  - That is, can we “chain” the blocks together?
- How to handle partial blocks?
  - We won’t discuss this issue

Modes of Operation

- Many modes (ECB, CBC, CTR, OFB, CFB)



- Encrypt each block independently
- Most obvious, but has a serious weakness

- Chain the blocks together
- More secure than ECB, virtually no extra work

- Block ciphers acts like a stream cipher
- Popular for random access

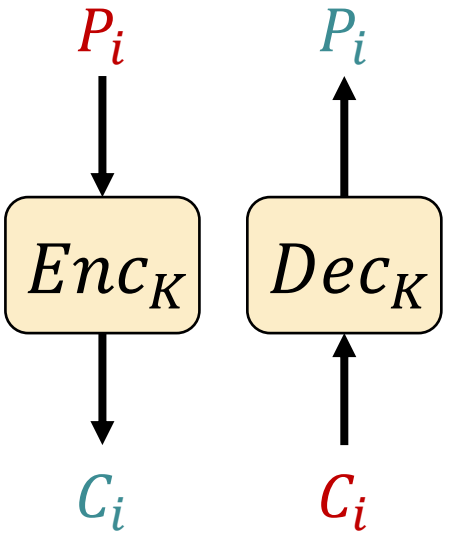
Presentation Outline

- More on Block Cipher
- **Block Cipher Modes**
  - **ECB**
  - CBC
  - CTR
- Integrity
- Authenticated Encryption

Electronic Codebook (ECB) Mode

- Notation: $C = Enc(P, K)$
- Message is broken into independent blocks of ***BlockSize*** bits
  - Given plaintext $P_1, P_2, \ldots, P_m, \ldots$
- Most obvious way to use a block cipher:

Encryption	**Decryption**
$C_1 = Enc(P_1, K)$	$P_1 = Dec(C_1, K)$
$C_2 = Enc(P_2, K)$	$P_2 = Dec(C_2, K)$
…	…
$C_i = Enc(P_i, K)$	$P_i = Dec(C_i, K)$ for i = 1, 2, …



  - Each block encrypted/decrypted separately
- For fixed key K, this is "electronic" version of a codebook cipher (without additive)
  - With a different codebook for each key

Electronic Codebook (ECB) Mode

Block cipher $\mathcal{E}: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

Alg $\mathrm{Enc}(K, M)$

---

1. **for** $i = 1, \ldots, m$ **do**
2. $\quad C_i \leftarrow \mathcal{E}_K(M_i)$
3. **return** $C_1 || C_2 || \cdots || C_m$

Alg $\mathrm{Dec}(K, C)$

---

1. **for** $i = 1, \ldots, m$ **do**
2. $\quad M_i \leftarrow \mathcal{E}_K^{-1}(C_i)$
3. **return** $M_1 || M_2 || \cdots || M_m$

Note: || refers to string concatenation

ECB Cut and Paste

- Suppose plaintext is

```
Money for Alice is $1000
Money for Trudy is $2
```

- Assuming 64-bit blocks and 8-bit ASCII:

  P_1 = "Money fo", P_2 = "r Alice ", P_3 = "is $1000",
  P_4 = "Money fo", P_5 = "r Trudy ", P_6 = "is $2 "

- Ciphertext: $C_1, C_2, C_3, C_4, C_5, C_6$
- Trudy cuts and pastes: $C_1, C_2, C_6, C_4, C_5, C_3$
- Decrypts as

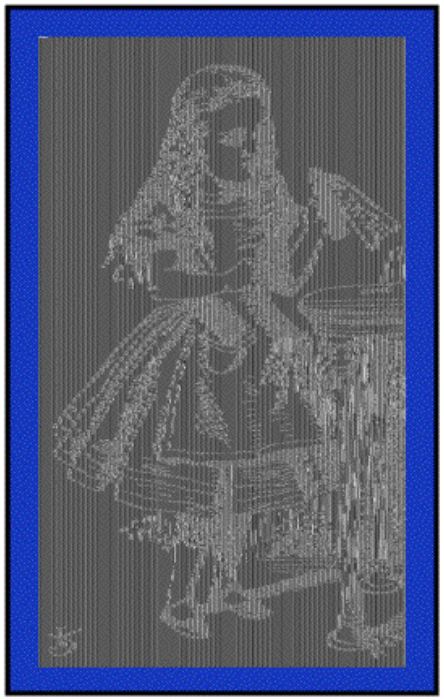
```
Money for Alice is $2
Money for Trudy is $1000
```

ECB Weakness

- Suppose $P_i = P_j$
- Then, $C_i = C_j$ and Trudy knows $P_i = P_j$
- This gives Trudy some information, even if she does not know P_i or P_j
- However, Trudy might know P_i
  - Suppose Trudy know that there are only two possible messages, for example, Vote: Yes or No
- Is this a serious issue?
  - Votes $M_1, M_2 \in \{Yes, No\}$ are ECB encrypted, and Trudy sees ciphertexts $C_1 = Enc(M_1, K)$ and $C_2 = Enc(M_2, K)$
  - Trudy may have cast the first vote and thus knows M_1; say $M_1 = Yes$, $M_1 = Dec(C_1, K)$. Then, Trudy can figure out M_2
  - If $C_2 = C_1$ then M_2 must be Yes, else M_2 must be No

Alice Hates ECB Mode

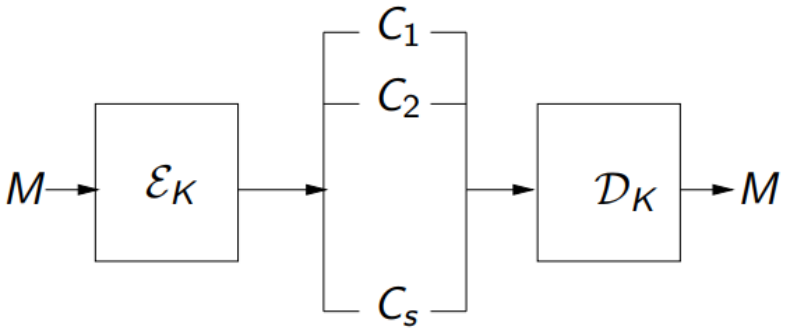
- Alice's uncompressed image, and the same image encrypted in ECB mode



- Why does this happen?
  - Because Enc is deterministic, same plaintext yields **same** ciphertext!

ECB Weakness

- Is this avoidable?
  - Yes, if Enc is probabilistic (non-deterministic)
    - That is, coins flipped internally by the Enc algorithm
  - Or, if Enc is nonce-based encryption algorithm
    - That is, encryption scheme itself is deterministic, non-determinism injected from the outside
  - Thus, if the same message is encrypted twice, we are likely to get back different answers
  - There are many possible ciphertexts corresponding to each message
- If so, how can we decrypt?
  - We will see examples soon



ECB Properties

- Deterministic
  - The same data block gets encrypted the same way; this reveals patterns of data when a data block repeats
- Malleable
  - Reordering ciphertext results in reordered plaintext
- Errors in one ciphertext block do not propagate
- Usage
  - *Not recommended* to encrypt more than one block of data

Presentation Outline

- More on Block Cipher
- **Block Cipher Modes**
  - ECB
  - **CBC**
  - CTR
- Integrity
- Authenticated Encryption

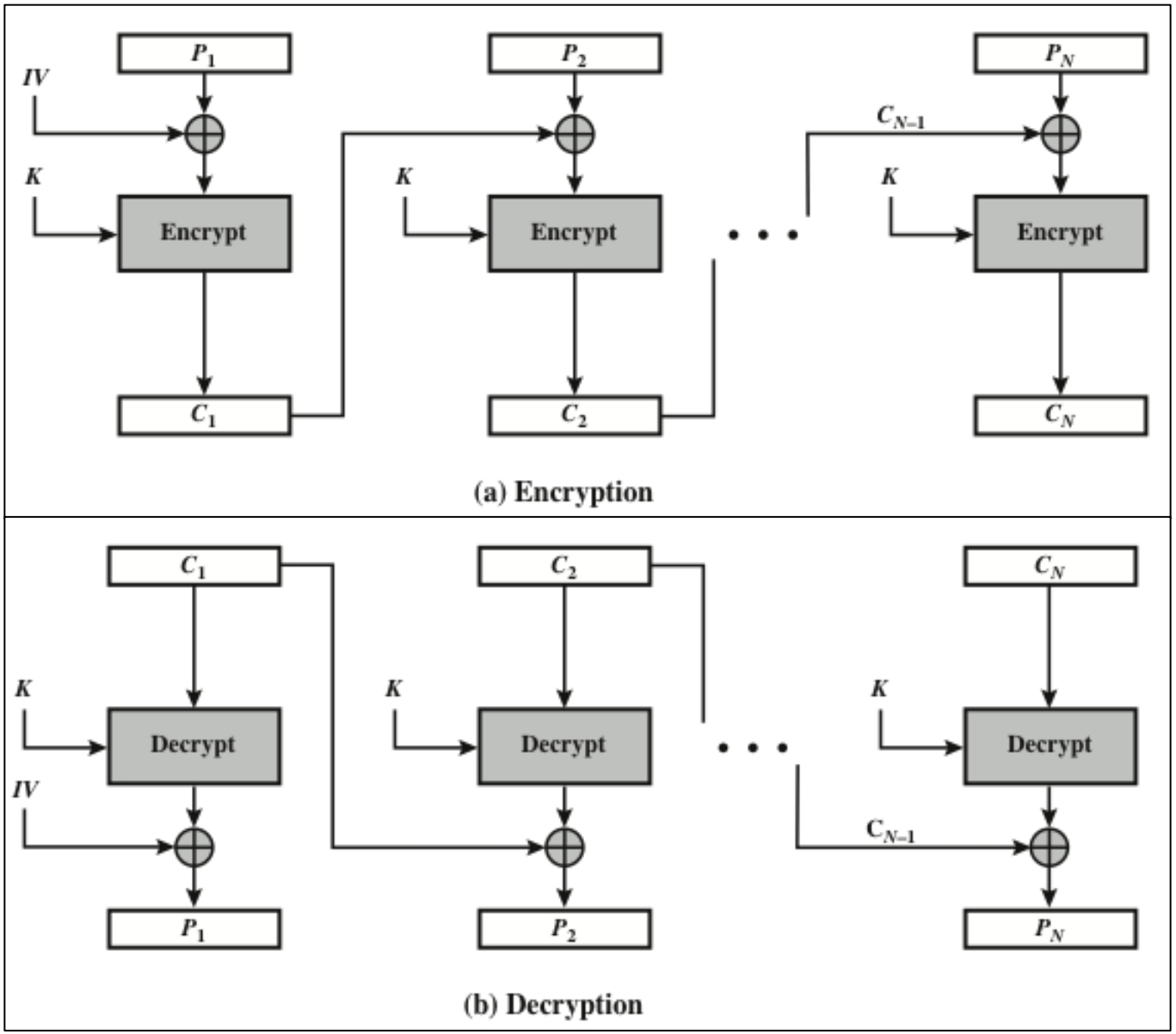
Cipher Block Chaining (CBC) Mode

- Blocks are "chained" together, next input depends upon previous output
- An initialization vector (IV) is required to initialize CBC mode

Encryption	Decryption
$C_0 = IV,$	$IV = C_0$
$C_1 = Enc(C_0 \oplus P_1, K),$	$P_1 = C_0 \oplus Dec(C_1, K),$
…	…
$C_i = Enc(C_{i-1} \oplus P_i, K)$	$P_i = C_{i-1} \oplus Dec(C_i, K)$ for $i = 1, 2, \ldots$

- Analogous to classic codebook with additive
- IV is usually randomly generated at encryption time and **sent** (or stored) as the **first** "ciphertext" block (IV is random, but ***not secret***), it should be "nonce" = "number used only once"

CBC Mode



CBC Mode

Block cipher $\mathcal{E}: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

Alg $\mathrm{Enc}(K, M)$

---

1. $\mathrm{IV} \xleftarrow{\$} \{0,1\}^n$
2. $C_0 \leftarrow \mathrm{IV}$
3. **for** $i = 1, \ldots, m$ **do**
4. $\quad C_i \leftarrow \mathcal{E}_K(M_i \oplus C_{i-1})$
5. **return** $C_0 || C_1 || \cdots || C_m$

Alg $\mathrm{Dec}(K, C)$

---

1. **for** $i = 1, \ldots, m$ **do**
2. $\quad M_i \leftarrow \mathcal{E}_K^{-1}(C_i) \oplus C_{i-1}$
3. **return** $M_1 || M_2 || \cdots || M_m$

Note: || refers to string concatenation

CBC Mode

- Suppose we encrypt $M^{(i)}, M^{(j)} \in \{Yes, No\}$
- Trudy sees $C^{(i)} = C^{(i)}_0 C^{(i)}_1$ and $C^{(j)} = C^{(j)}_0 C^{(j)}_1$
- Suppose Trudy knows $M^{(i)} = Yes$
- Can Trudy determine whether $M^{(j)} = Yes$ or $M^{(j)} = No$?
  - ✧ No, Trudy cannot determine $M^{(j)}$
- Identical plaintext blocks yield **different** ciphertext blocks — this is good!
- So, CBC is better than ECB. But is it perfectly secure?
  - ✧ No, CBC is **not** perfectly indistinguishable
  - ✧ Observation:

    if $C_i = C_j$, then $Enc(M_i \oplus C_{i-1}) = Enc(M_j \oplus C_{j-1})$

    thus, $M_i \oplus C_{i-1} = M_j \oplus C_{j-1}$

    thus, $M_i \oplus M_j = C_{i-1} \oplus C_{j-1}$

CBC Mode

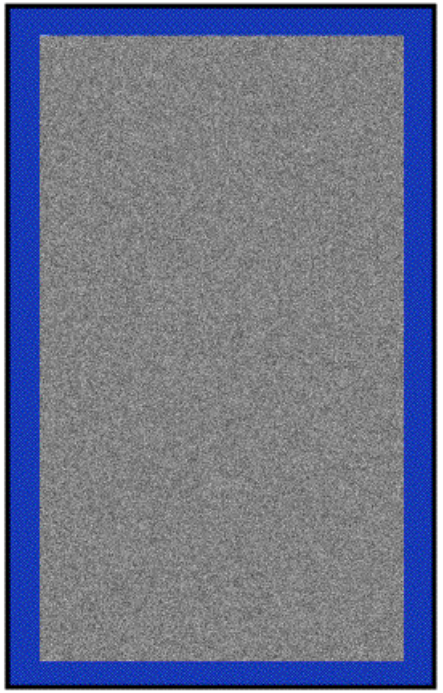
- If C_1 is garbled to, say, $G \neq C_1$ then

$$P_1 \neq C_0 \oplus \text{Dec}(G, K), P_2 \neq G \oplus \text{Dec}(C_2, K)$$

- But $P_3 = C_2 \oplus \text{Dec}(C_3, K)$, $P_4 = C_3 \oplus \text{Dec}(C_4, K)$, …
- Automatically recovers from errors!
  - Each plaintext block only depends on two consecutive ciphertext blocks, so errors do not propagate beyond two blocks

Alice Likes CBC Mode

- Alice's uncompressed image, and the same image encrypted in CBC mode



- Why does this happen?
  - Because Enc is non-deterministic, same plaintext yields **different** ciphertext!

CBC Properties

- Probabilistic encryption: repeated text gets mapped to different encrypted data
  - Can be proven to be "secure" assuming that the block cipher has desirable properties and that **random IV's** are used
- A ciphertext block depends on all preceding plaintext blocks; reorder affects decryption
- Errors in one block propagate to two blocks
  - One bit error in C_j affects all bits in M_j and one bit in M_{j+1}
- Sequential encryption, cannot use parallel hardware
- Usage: chooses random IV and protects the integrity of IV

Example

- Suppose that we use a block cipher to encrypt according to the rule

  $C_1 = IV \oplus Enc(P_1, K)$, $C_2 = C_1 \oplus Enc(P_2, K)$, $C_3 = C_2 \oplus Enc(P_3, K)$, …

  a) What is the corresponding decryption rule?

  b) Give two security disadvantages of this mode as compared to CBC mode.

- Solution:

  a) $P_1 = Dec(C_1 \oplus IV, K)$, and $P_i = Dec(C_i \oplus C_{i-1}, K)$ for i = 2, 3, …

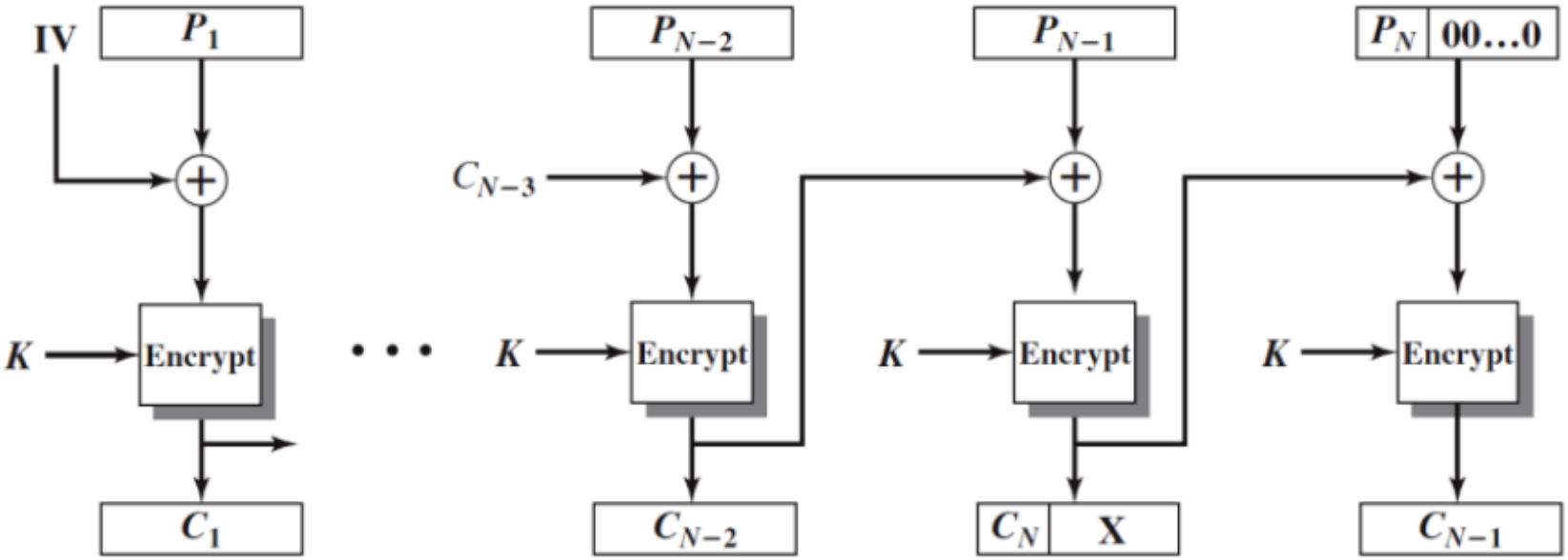
  b) If $P_i = P_j$ for some i and j $\Rightarrow$ $Enc(P_i, K) = Enc(P_j, K)$ $\Rightarrow$ $C_i \oplus C_{i-1} = C_j \oplus C_{j-1}$ $\Rightarrow$ Same problems as with ECB (i.e., cut-and-paste attack, revealing the original plaintext contents from ciphertext, …)

Example

Bob wishes to encrypt some plaintext and store the resulting ciphertext on his hard drive. Specifically, he wants the ciphertext to be the *same length* as the original plaintext. For this purpose, he employed the **ciphertext stealing (CTS)** mode, the implementation of which is shown in the figure in the next page. Initially, the plaintext is divided into independent blocks of length $\boldsymbol{S}$ bits, giving the plaintext blocks $\boldsymbol{P_1}$, $\boldsymbol{P_2}$, …, $\boldsymbol{P_N}$. Assume that the last block of plaintext (i.e., $\boldsymbol{P_N}$) is $\boldsymbol{L}$ bits long, where $\boldsymbol{L < S}$. The encryption sequence is as follows:

1) Encrypt the first $(\boldsymbol{N-2})$ blocks using the traditional cipher block chaining (**CBC**) technique.
2) XOR $\boldsymbol{P_{N-1}}$ with the previous ciphertext block $\boldsymbol{C_{N-2}}$ to create $\boldsymbol{Y_{N-1}}$.
3) Encrypt $\boldsymbol{Y_{N-1}}$ to create $\boldsymbol{E_{N-1}}$.
4) Select the first $\boldsymbol{L}$ bits of $\boldsymbol{E_{N-1}}$ to create $\boldsymbol{C_N}$.
5) Pad $\boldsymbol{P_N}$ with $(\boldsymbol{S-L})$ zeros at the end and exclusive-OR with $\boldsymbol{E_{N-1}}$ to create $\boldsymbol{Y_N}$.
6) Encrypt $\boldsymbol{Y_N}$ to create $\boldsymbol{C_{N-1}}$.

Example



a) Describe how to decrypt the ciphertext $(\boldsymbol{C_1}, \ldots, \boldsymbol{C_{N-1}}, \boldsymbol{C_N})$, that is, show the decryption sequence.

b) If a single-bit error occurs in the storage of ciphertext $\boldsymbol{C_i}$, which plaintext blocks, if any, will be correctly restored by the decryption algorithm? Explain your answer.

Example

- Solution – (a):

1) Decrypt the first $(N-2)$ ciphertext blocks $(C_1, \ldots, C_{N-2})$ using the traditional cipher block chaining (CBC) technique ,
   $P_i = C_{i-1} \oplus Dec(K, C_i), \forall\ i \in [1, N-2], C_0 = IV$
2) Decrypt C_{N-1} to create Y_N , $Y_N = Dec(K, C_{N-1})$
3) Select the first L bits of Y_N and exclusive-OR with C_N to calculate P_N ,
   $P_N = Y_N[1:L] \oplus C_N$
4) Concatenate C_N with the last $(S-L)$ bits of Y_N to create E_{N-1} ,
   $E_{N-1} = C_N \,||\, Y_N[L+1:S]$
5) Decrypt E_{N-1} to create Y_{N-1} , $Y_{N-1} = Dec(K, E_{N-1})$
6) XOR Y_{N-1} with the previous ciphertext block C_{N-2} to calculate P_{N-1} ,
   $P_{N-1} = Y_{N-1} \oplus C_{N-2}$

Example

- Solution – (b):

If C_i is garbled to, say, $G \neq C_i$ then

$P_i \neq C_{i-1} \oplus Dec(K, G)$, $P_{i+1} \neq G \oplus Dec(K, C_{i+1})$

But $P_{i-1} = C_{i-2} \oplus Dec(K, C_{i-1})$, $P_{i-2} = C_{i-3} \oplus Dec(K, C_{i-2})$ … and

$$P_{i+2} = C_{i+1} \oplus Dec(K, C_{i+2}) \text{ , } P_{i+3} = C_{i+2} \oplus Dec(K, C_{i+3}) \ldots$$

In other words, each plaintext block only depends on two consecutive ciphertext blocks, so errors do not propagate beyond two blocks.

Thus, all plaintext blocks except plaintext blocks P_i and P_{i+1} will be correctly restored by the decryption algorithm.

Presentation Outline

- More on Block Cipher
- **Block Cipher Modes**
  - ECB
  - CBC
  - **CTR**
- Integrity
- Authenticated Encryption

Counter (CTR) Mode

- Use the block cipher as a keystream generator
  - Another way to construct PRNG using DES
  - Can also be used for random access, with a significant limitation …
- Counter (CTR) mode employs an initialization vector (IV), or a number used only once (nonce), referred to as ctr

Encryption	Decryption
$C_0 = ctr,$	$ctr = C_0,$
$C_1 = P_1 \oplus Enc(ctr + 1, K),$	$P_1 = C_1 \oplus Enc(ctr + 1, K),$
…	…
$C_i = P_i \oplus Enc(ctr + i, K)$	$P_i = C_i \oplus Enc(ctr + i, K)$ for $i = 1, 2, \ldots$

  - CTR mode allows parallelization of encryption. Thus, suited for high-speed implementations
  - Sender and receiver share: ctr (does **not** need to be secret) and the secret key k

Counter (CTR) Mode

Block cipher $\mathcal{E}: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

In nonce-based encryption, the ctr will not be chosen at random. Instead, it will be an input to the Enc algorithm, i.e., Enc(K, M, Nonce). Thus, Enc becomes a deterministic function.

Note: $(\boldsymbol{ctr} + \boldsymbol{i})$ denotes the n-bit string formed by converting $\boldsymbol{ctr}$ to an integer, calculating $(\boldsymbol{ctr}_{int} + \boldsymbol{i})$ modulo $\boldsymbol{2^n}$, and then converting the result back to an n-bit string

Note: || refers to string concatenation

Alg $\mathrm{Enc}(K, M)$

---

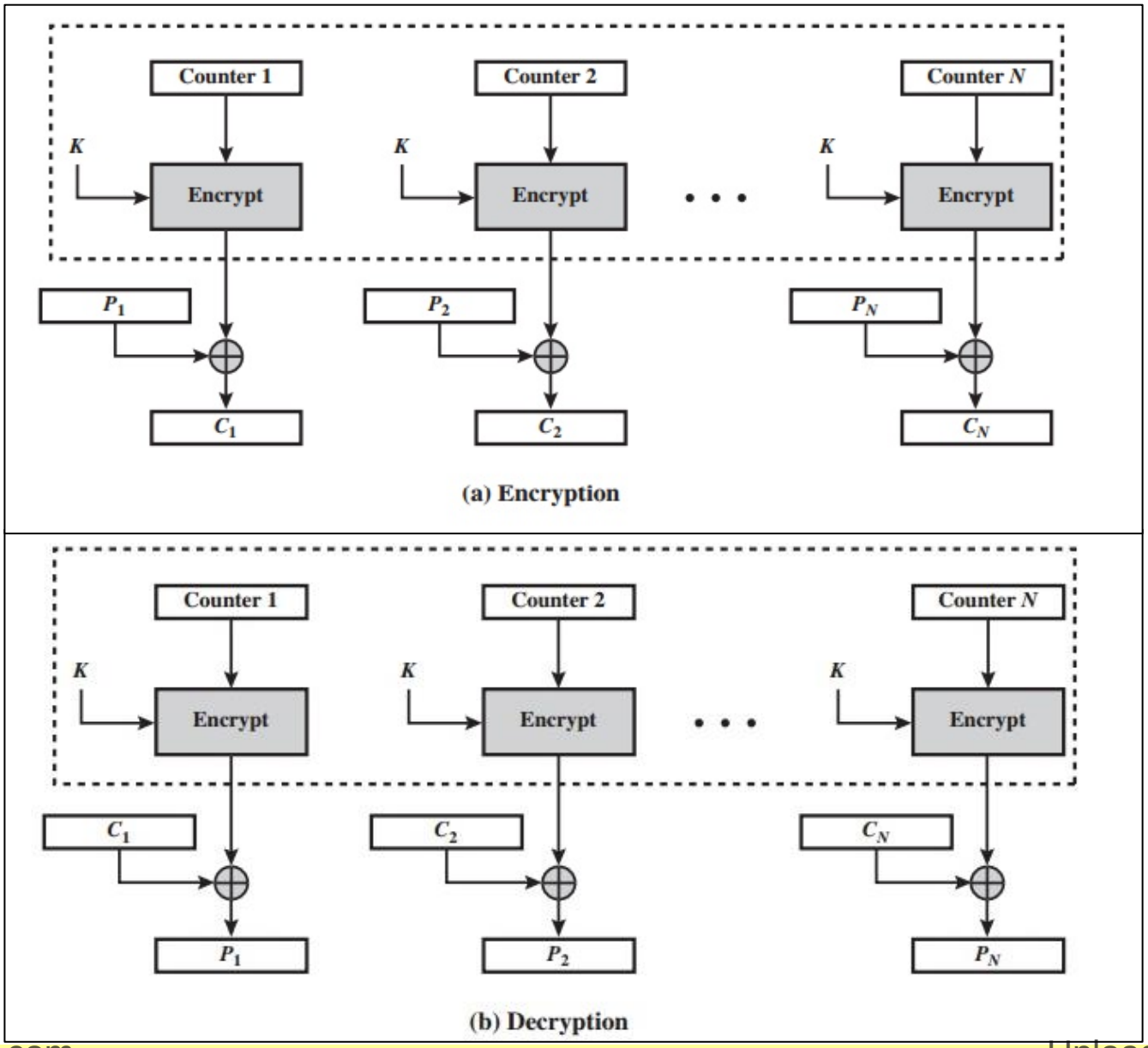
1. $\mathrm{ctr} \xleftarrow{\$} \{0,1\}^n$
2. $C_0 \leftarrow \mathrm{ctr}$
3. **for** $i = 1, \ldots, m$ **do**
4. $\quad C_i \leftarrow \mathcal{E}_K(\mathrm{ctr} + i) \oplus M_i$
5. **return** $C_0 || C_1 || \cdots || C_m$

Alg $\mathrm{Dec}(K, C)$

---

1. $\mathrm{ctr} \leftarrow C_0$
2. **for** $i = 1, \ldots, m$ **do**
3. $\quad M_i \leftarrow \mathcal{E}_K(\mathrm{ctr} + i) \oplus C_i$
4. **return** $M_1 || M_2 || \cdots || M_m$

Counter (CTR) Mode



CTR Properties

- Software and hardware efficiency
  - Different blocks can be encrypted/decrypted in parallel
- Preprocessing
  - The encryption part can be done offline and when the message is known, just do the XOR
- Random access
  - Decryption of a block can be done in random order, very useful for hard-disk encryption
- Messages of arbitrary length
  - Ciphertext is the same length with the plaintext
- Simplicity
  - Can use a family of functions Enc that is not required to be a block cipher as the CTR mode doesn't use Enc^{-1}

CTR Security

- Birthday attack:
  - Suppose **1-block** messages $M^{(0)}$ and $M^{(1)}$ are encrypted
  - Then, with a CTR symmetric encryption scheme:

    $C^{(0)}_0 C^{(0)}_1 = Enc^{CTR}(K, M^{(0)})$ and $C^{(1)}_0 C^{(1)}_1 = Enc^{CTR}(K, M^{(1)})$
  - If $C^{(0)}_0 = C^{(1)}_0$, referred to as **event collision**, then:
    - $C^{(0)}_1 = C^{(1)}_1$ if and only if $M^{(0)} = M^{(1)}$
    - $C^{(0)}_1 \oplus C^{(1)}_1 = M^{(0)} \oplus M^{(1)}$
  - So, if adversary is lucky, he can detect message equality and violate the indistinguishability under chosen plaintext attack (IND-CPA)
- The CTR mode **can be broken** (in the IND-CPA sense) using the birthday attack in about $\mathbf{2^{(n/2)}}$ **queries**, where **n** is the block length of the underlying block cipher
  - The attack does not depend on the key length of the block cipher

Block Cipher Modes

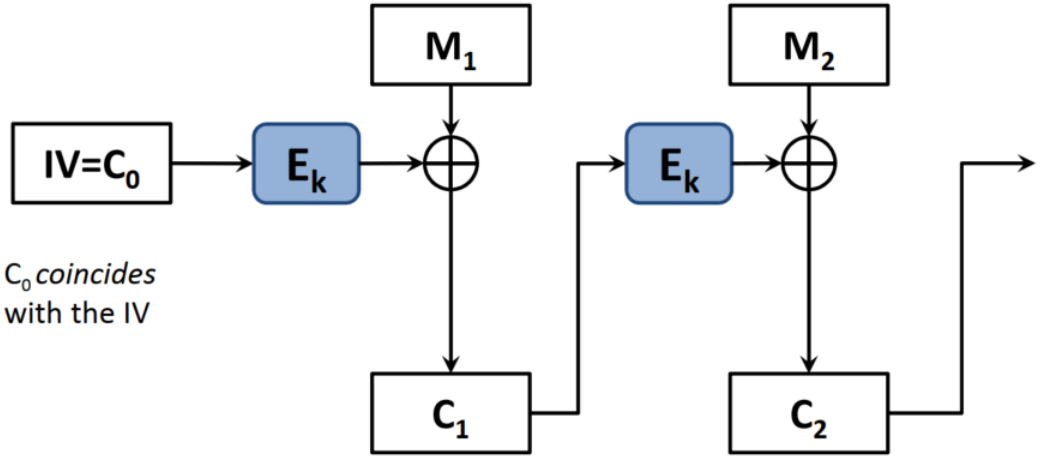
- Other Block Cipher Modes
  - CFB
  - OFB

Cipher Feedback (CFB) Mode

- The message is XORed with the feedback of encrypting the previous block
- **Encryption**

$$C_i = M_i \oplus \text{Enc}(C_{i-1}, K) \ ,$$

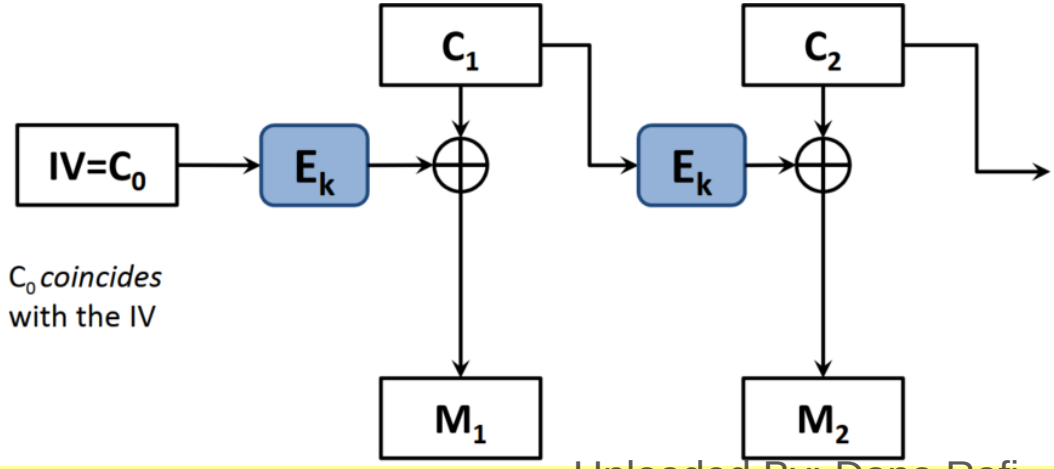
with $C_0 = IV$



- **Decryption**

$$M_i = C_i \oplus \text{Enc}(C_{i-1}, K) \ ,$$

with $C_0 = IV$

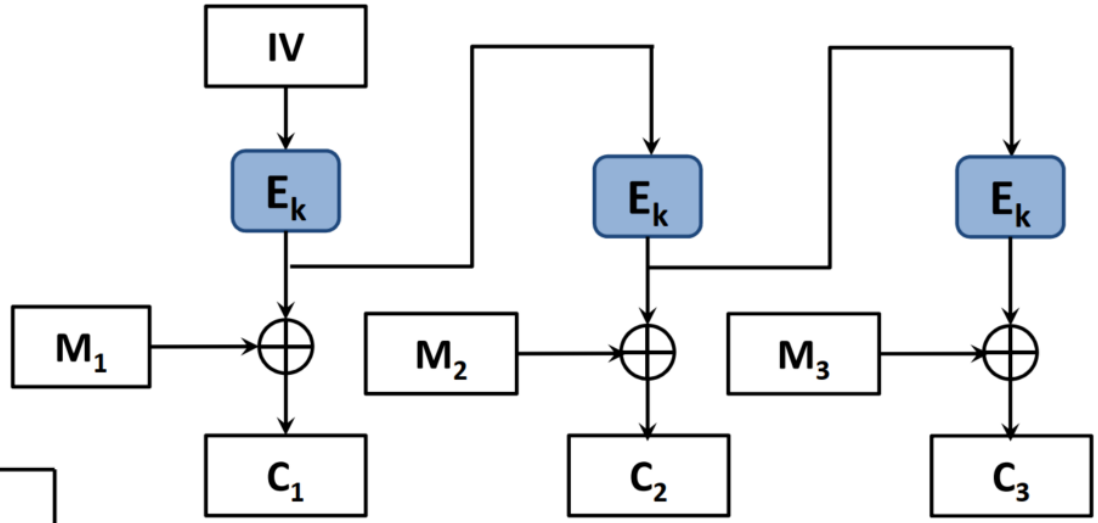


CFB Properties

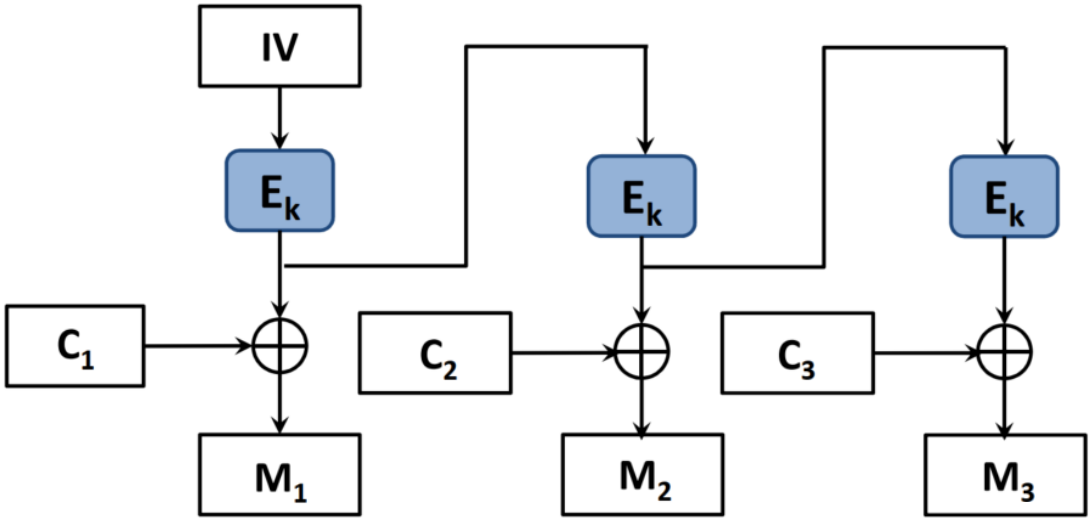
- Probabilistic encryption
- Sequential encryption
  - A ciphertext block depends on all preceding plaintext blocks; reorder affects decryption
- Errors propagate for several blocks after the error, but the mode is self-synchronizing (like CBC)
- Decreased throughput
  - Can vary the number of bits feedback, trading off throughput for ease of use

Output Feedback (OFB) Mode

- Constructs a Pseudo Random Number Generator using DES Enc function



Encryption



Decryption

OFB Properties

- Probabilistic encryption
- Sequential encryption, but pre-processing possible
- Error propagation limited
- Subject to limitations of stream ciphers

Presentation Outline

- Block Cipher Modes
  - ECB
  - CBC
  - CTR
- **Integrity**
- Authenticated Encryption

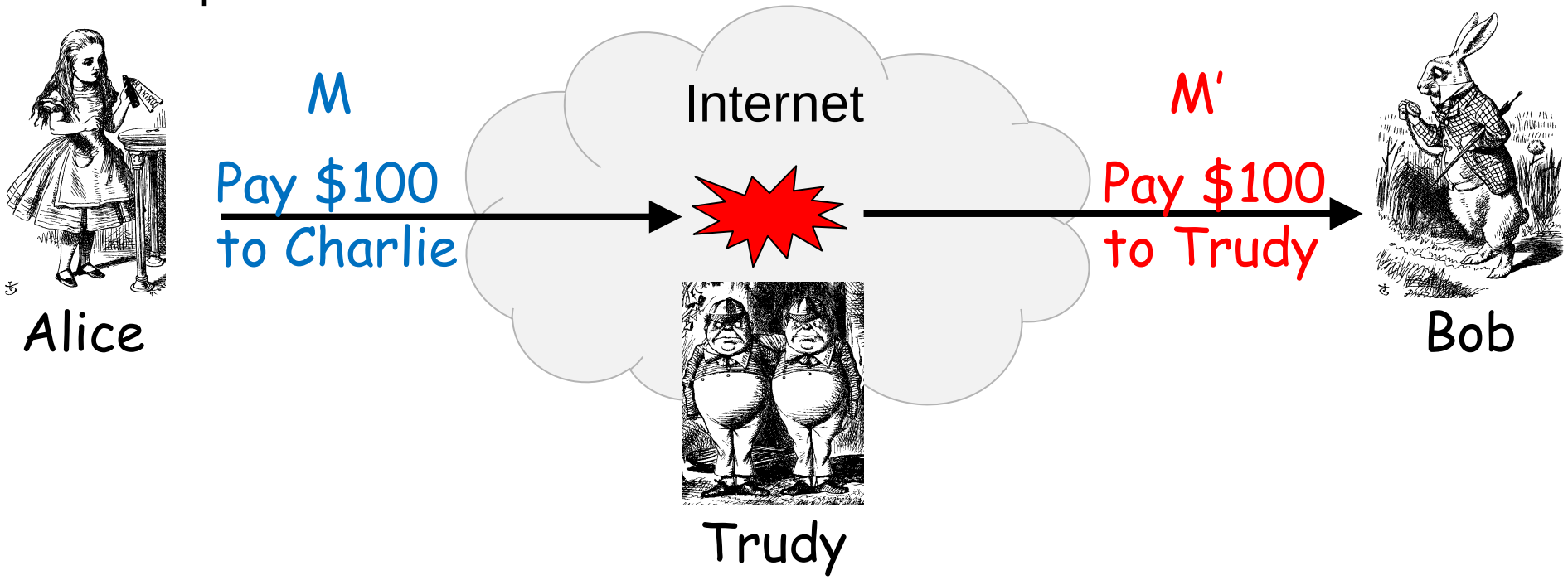
Basic Goals of Cryptography

	Message Privacy	Message Integrity / Authentication
Symmetric Keys	Symmetric Encryption (private-key encryption)	Message Authentication Codes (MAC)
Asymmetric Keys	Asymmetric Encryption (public-key encryption)	Digital Signatures

Data Integrity

- **Integrity** — detect unauthorized *writing* (i.e., modification of data)

  Protecting:
  - ✧ OS system files against tempering
  - ✧ Browser cookies stored by web servers
  - ✧ Control signals in network management
- Example: Inter-bank fund transfers

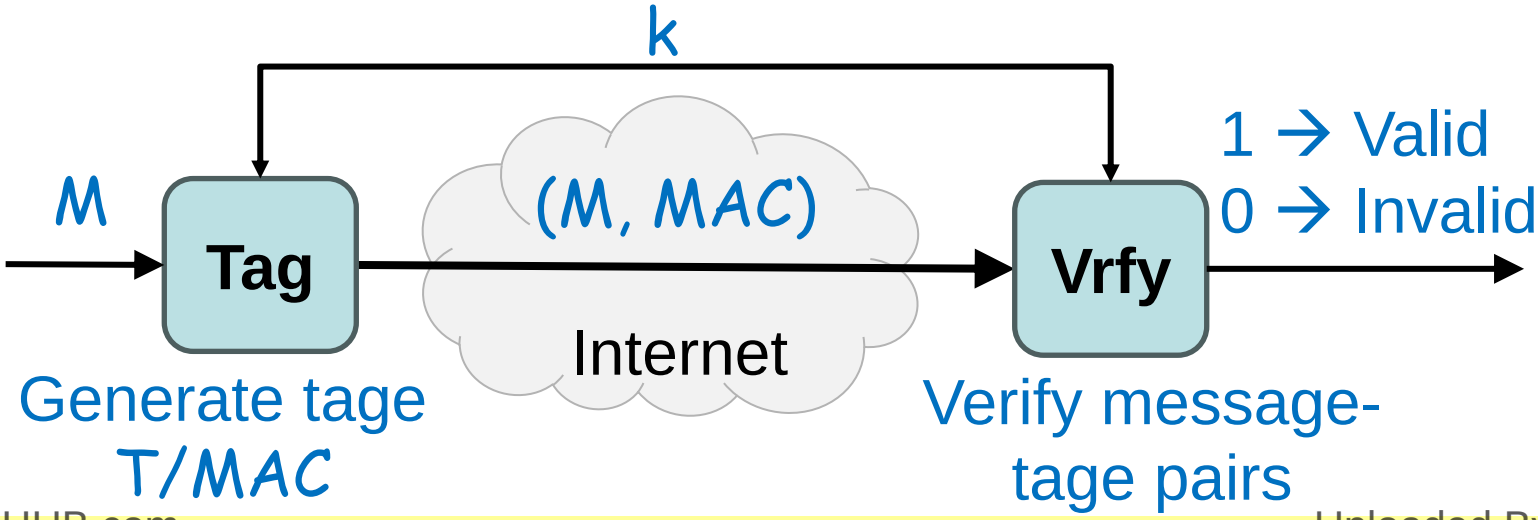


  - ✧ Confidentiality may be nice, integrity is critical

Message Authentication Code (MAC)

- Encryption provides **confidentiality** (prevents unauthorized disclosure)
- Encryption alone does **not** provide integrity
  - One-time pad, ECB cut-and-paste, etc.
- Message Authentication Codes (*MACs*)
  - Also called the "**cryptographic checksums**"
  - Used for data **integrity** (Integrity **not** the same as confidentiality)

Alice

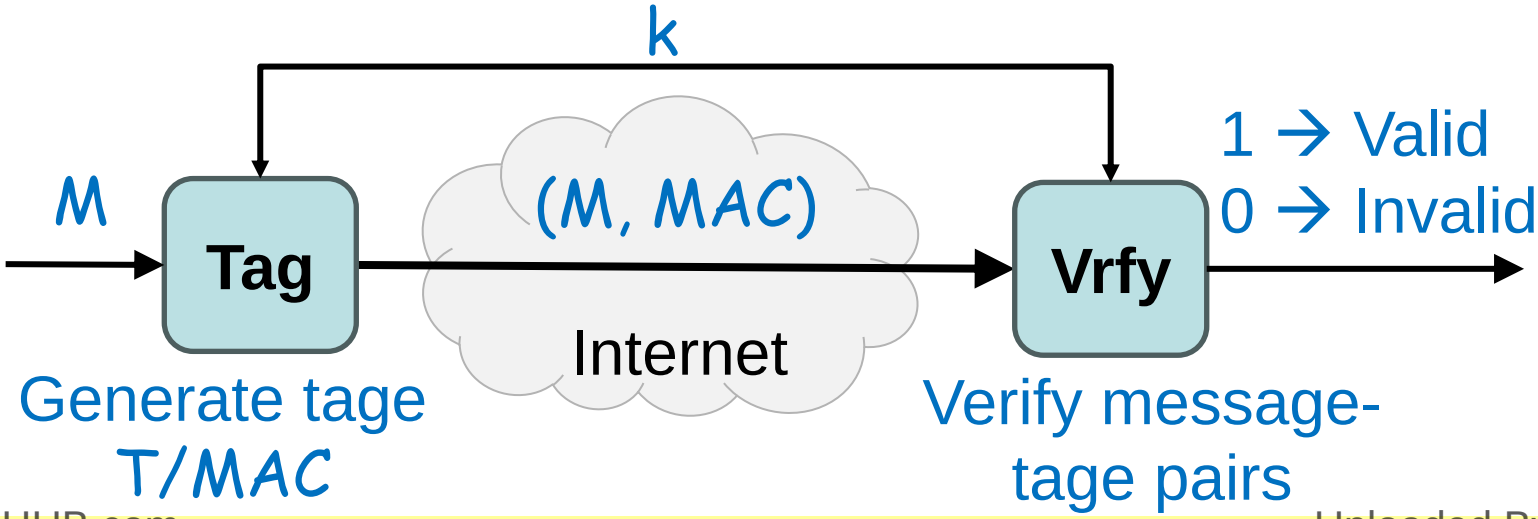


Bob

Properties of MACs

- Arbitrary input length
- Fixed output length
- Message Authentication
  - Bob is certain that Alice sent the message
- Integrity
  - Manipulations in transit will be detected by Bob
- Non-repudiation is NOT given (Non-non-repudiation)
  - Offers no protection if Alice & Bob try to cheat each other

Alice

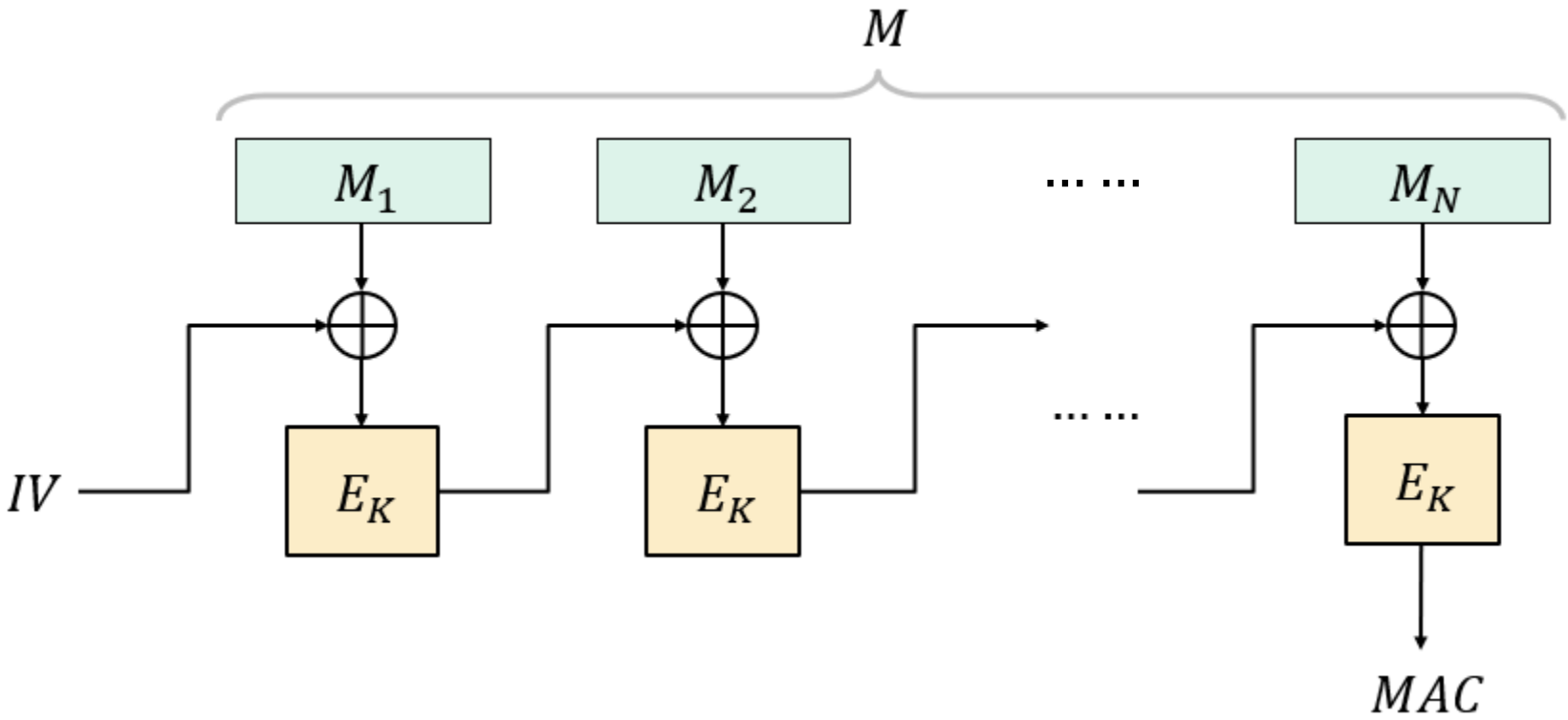


Bob

MAC Computation/Verification

- *MAC* is computed as **CBC residue**
  - That is, compute CBC encryption, saving only final ciphertext block
- *MAC* computation (assuming N blocks)

$C_0 = IV \quad , \quad C_1 = Enc(C_0 \oplus M_1, K) \quad , \ldots , \quad C_N = Enc(C_{N-1} \oplus M_N, K) = \mathbf{MAC}$



MAC Computation/Verification

- *MAC* sent with IV and plaintext
  - Remember that our goal here is integrity, but not confidentiality
- Correctness requirement: Vrfy(K, M, Tag(K, M)) = 1 (Valid)
- Receiver does same computation and verifies that result agrees with *MAC* (Note: receiver must know the key K)
  - If the computed "*MAC*" matches the received *MAC*, then we are satisfied with the integrity of the data
  - If the computed "*MAC*" does not match the received *MAC*, then we know that something is amiss
- Computing a *MAC* based on CBC encryption is not the only way to provide data integrity, another options
  - Hashed MAC (HMAC) and digital signature (will be discussed later)

Does a MAC work?

- Suppose Alice has 4 plaintext blocks
- Alice computes

  $C_0 = IV$, $C_1 = Enc(C_0 \oplus P_1, K)$, $C_2 = Enc(C_1 \oplus P_2, K)$,
  $C_3 = Enc(C_2 \oplus P_3, K)$, $C_4 = Enc(C_3 \oplus P_4, K) = MAC$

- Alice sends IV, P_1, P_2, P_3, P_4 and ***MAC*** to Bob
- Suppose Trudy ***changes*** P_2 to X
- Bob computes

  $C_0 = IV$, $C_1 = Enc(C_0 \oplus P_1, K)$, $C_2 = Enc(C_1 \oplus X, K)$,
  $C_3 = Enc(C_2 \oplus P_3, K)$, $C_4 = Enc(C_3 \oplus P_4, K) = MAC \neq MAC$

- That is, error propagates into ***MAC***
- Trudy can't make ***MAC*** == ***MAC*** without K

Does a MAC work? (cont.)

- Suppose Alice has 4 plaintext blocks
- Alice computes

  $C_0 = IV$, $C_1 = Enc(C_0 \oplus P_1, K)$, $C_2 = Enc(C_1 \oplus P_2, K)$,
  $C_3 = Enc(C_2 \oplus P_3, K)$, $C_4 = Enc(C_3 \oplus P_4, K) = MAC$

- Alice sends IV, P_1, P_2, P_3, P_4 and ***MAC*** to Bob
- Suppose Trudy ***swaps*** P_2 and P_3
- Bob computes

  $C_0 = IV$, $C_1 = Enc(C_0 \oplus P_1, K)$, $C_2 = Enc(C_1 \oplus P_3, K)$,
  $C_3 = Enc(C_2 \oplus P_2, K)$, $C_4 = Enc(C_3 \oplus P_4, K) = MAC \neq MAC$

- That is, error propagates into ***MAC***
- Trudy can't make ***MAC*** == ***MAC*** without K

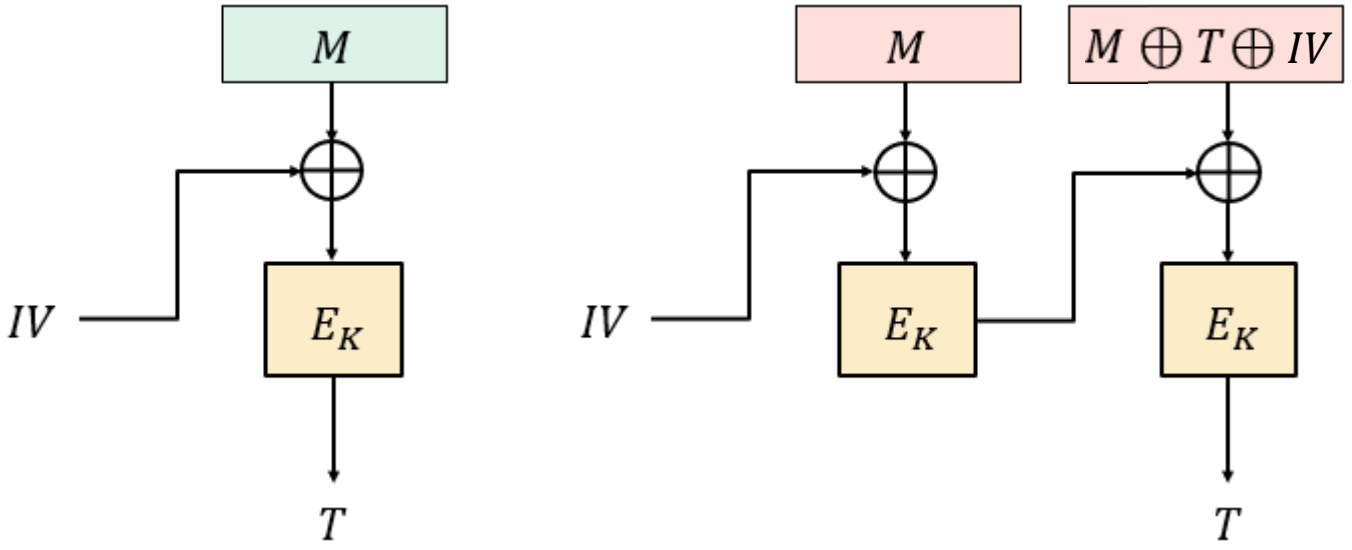
Example

- Suppose that you know a *MAC* value X and the key K that was used to compute the *MAC*, but you do not know the original message
  a) Show that you can construct a message M that also has its *MAC* equal to X. Note that we are assuming that you know the key K and the same key is used for both *MAC* computations.
  b) How much of the message M are you free to choose?

- Solution:
  a) Since we know K and *MAC*, then by using CBC mode we can set P_n to be any desired message block, and choose $C_{n-1} = Dec(MAC, K) \oplus P_n$. We can set P_i to be any desired message block, and choose $C_{i-1} = Dec(C_i, K) \oplus P_i$, where $1 < i \leq n - 1$. Finally, we can set P_1 to be any desired message block, and choose $IV = Dec(C_1, K) \oplus P_1$.
  b) Given that there are no restriction on IV, then the compute message can be freely chosen.

CBC-MAC Security

- ***CBC-MAC*** is UF-CMA (unforgeability against chosen-message attacks) secure ***MAC*** for messages of a **single** fixed-length



$$\text{CBC.Tag}(K, M||M \oplus T \oplus IV)$$
$$= E_K\big(E_K(IV \oplus M) \oplus (M \oplus T \oplus IV)\big)$$
$$= E_K\left(T \oplus (M \oplus T \oplus IV)\right)$$
$$= E_K\ (M \oplus IV)$$
$$= T$$

$$\mathbf{Adv}^{\text{uf-cma}}_{\text{CBC-MAC}}(A) = \Pr[A \text{ wins}] = 1$$

Adversary is doing well

$\text{Exp}^{\text{uf-cma}}_{\Sigma}(A)$

1. $K \xleftarrow{\$} \Sigma.\text{KeyGen}$
2. $S \leftarrow [\,];\ \ won \leftarrow 0$
3. $A_K^{\text{Tag}_K(\cdot),\ \text{VF}_K(\cdot,\cdot)}$
4. **return** won

$\text{Tag}_K(M)$

---

1. $T \leftarrow \Sigma.\text{Tag}(K, M)$
2. $S.\text{add}\big((M, T)\big)$
3. **return** T

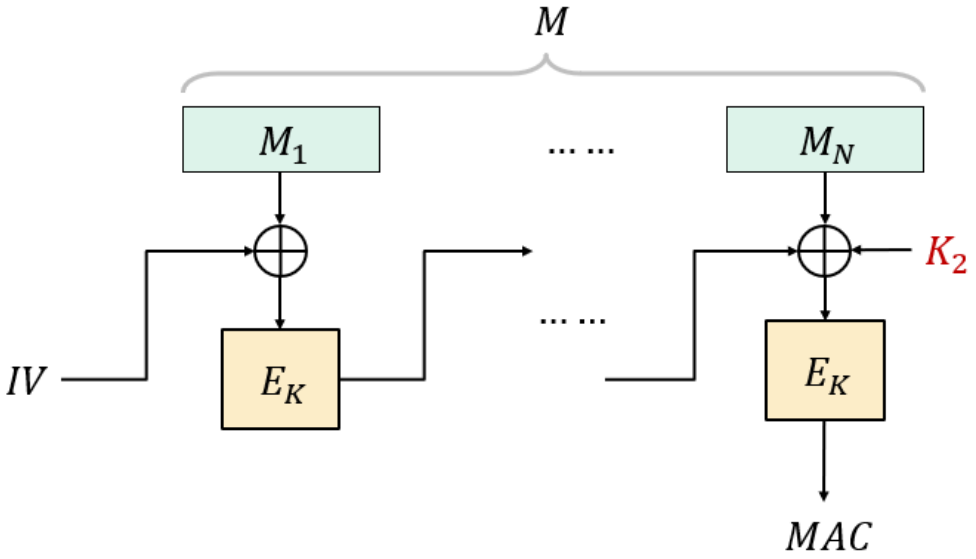
$\text{VF}_K(M, T)$

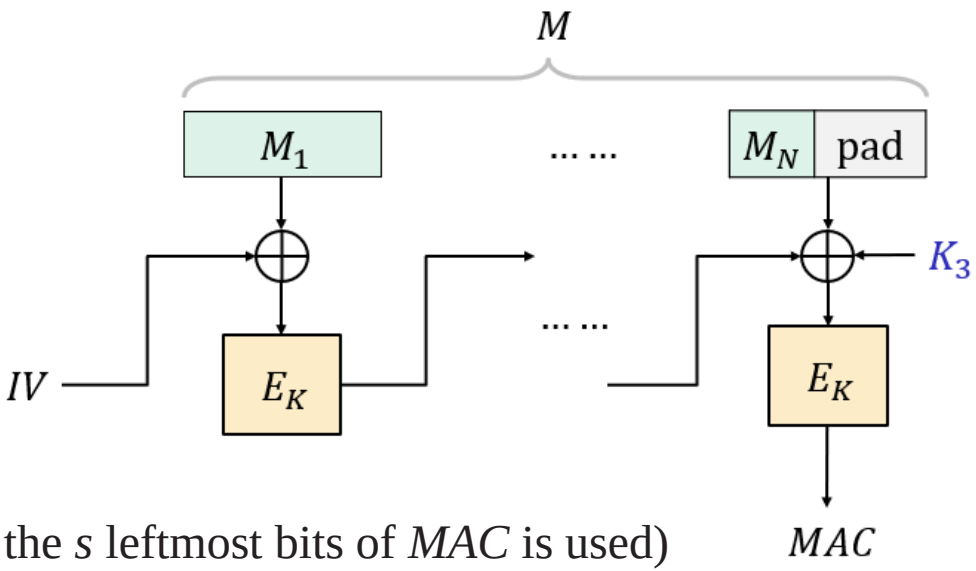
---

1. $d \leftarrow \Sigma.\text{Vrfy}(K, M, T)$
2. **if** $d = 1$ and $(M, T) \notin S$ **then:**
3. $\quad won \leftarrow 1$
4. **return** d

Cipher-based MAC (CMAC)

- For variable-length messages, the **CMAC** (a.k.a. One-key CBC-MAC (**OMAC**)) is used to upgrade **CBC-MAC**
  - The CMAC derives two keys (K_2, K_3) from the master key (K)
  - K_2 is used when the message is a multiplicative of the block size, while K_3 is used when the padding is needed (The final block is padded to the right with a 1 and as many 0s as necessary)
  - $K_2 = 2 * E_K(0^n)$ and $K_3 = 4 * E_K(0^n)$, $*$ is the multiplication in GF(2^n)



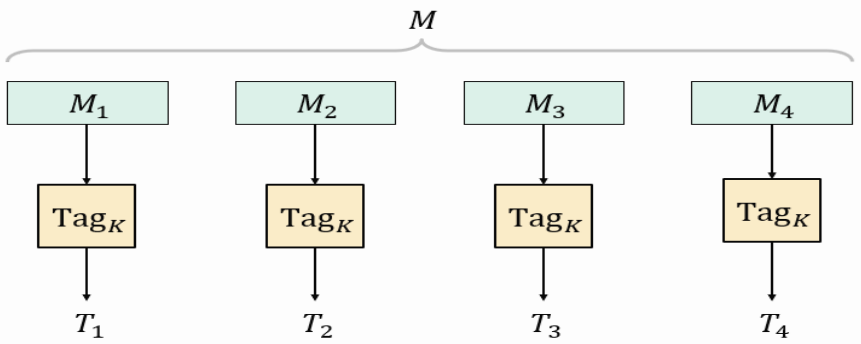


(or, the s leftmost bits of MAC is used)

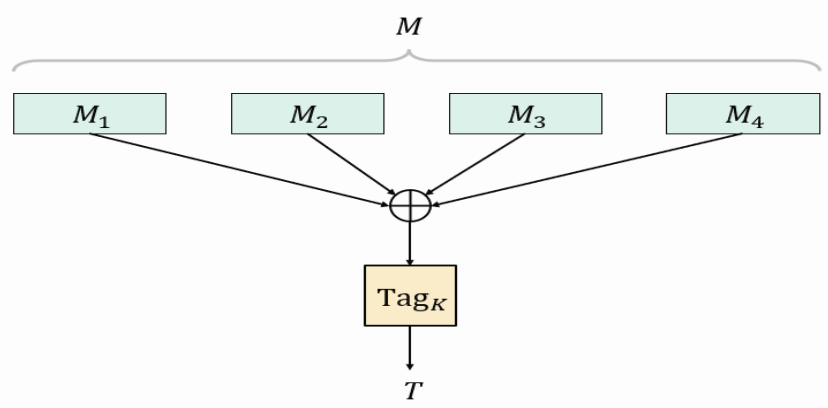
Example

❖ Can we use any of the following methods to compute the **MAC**?

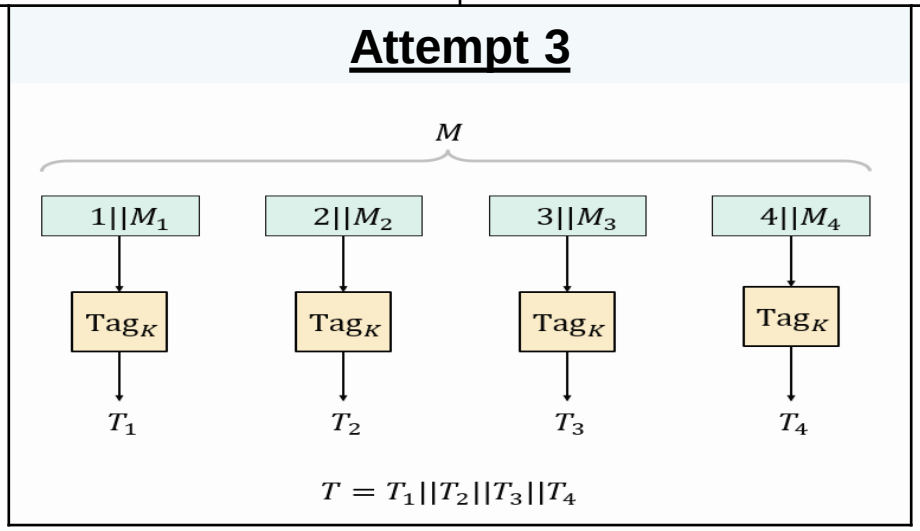
Attempt 1



Attempt 2



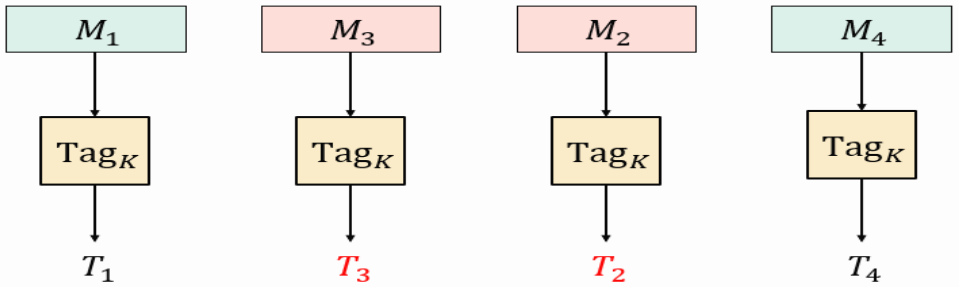
Attempt 3



Example

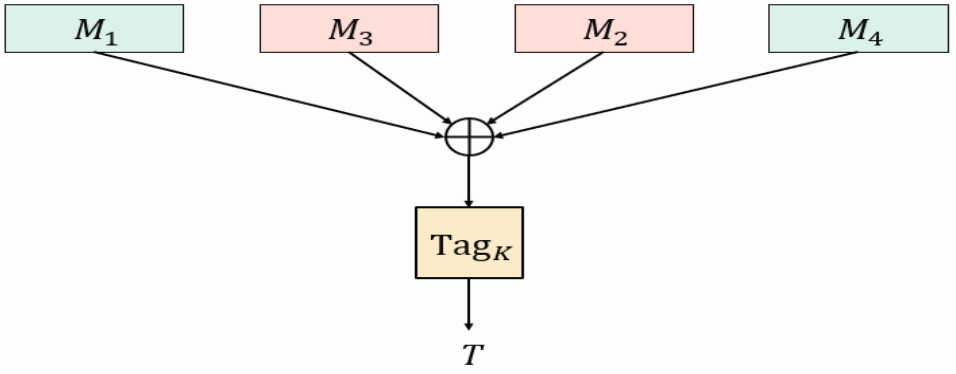
✧ No, there is a possible attack on each of the given attempts

Attempt 1 Attack



$$T = T_1||T_3||T_2||T_4$$

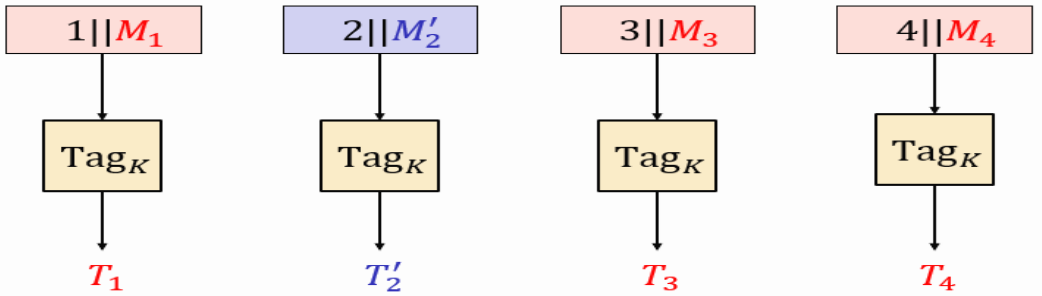
Attempt 2 Attack



Attempt 3 Attack

$$M = M_1||M_2||M_3||M_4 \xrightarrow{\text{Tag}_K} T_1||T_2||T_3||T_4$$

$$M' = M'_1||M'_2||M'_3||M'_4 \xrightarrow{\text{Tag}_K} T'_1||T'_2||T'_3||T'_4$$



$$T = T_1||T'_2||T_3||T_4$$

Presentation Outline

- Block Cipher Modes
  - ECB
  - CBC
  - CTR
- Integrity
- **Authenticated Encryption**

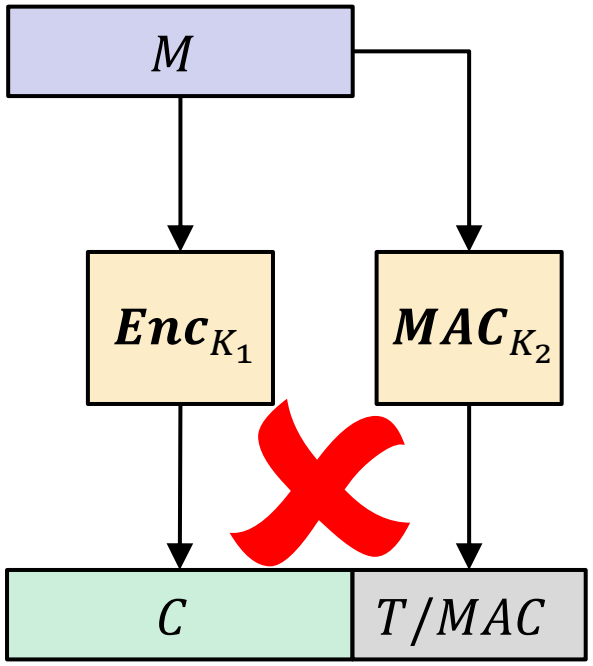
Basic Goals of Cryptography

	Message Privacy	Message Integrity / Authentication
Symmetric Keys	Symmetric Encryption (private-key encryption)	Message Authentication Codes (MAC)
Asymmetric Keys	Asymmetric Encryption (public-key encryption)	Digital Signatures

Confidentiality, Integrity, and Authenticity

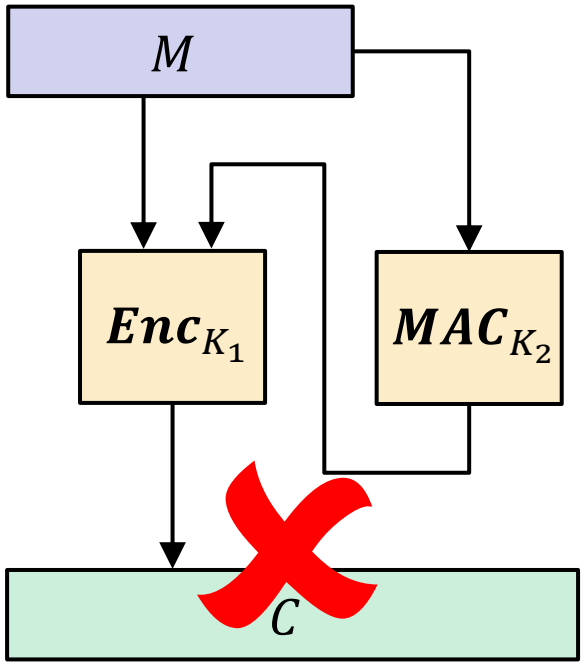
- Authenticated encryption: privacy and integrity in one primitive
- Encrypt with one key, and then, MAC with another key

Encrypt-and-MAC (E&M)



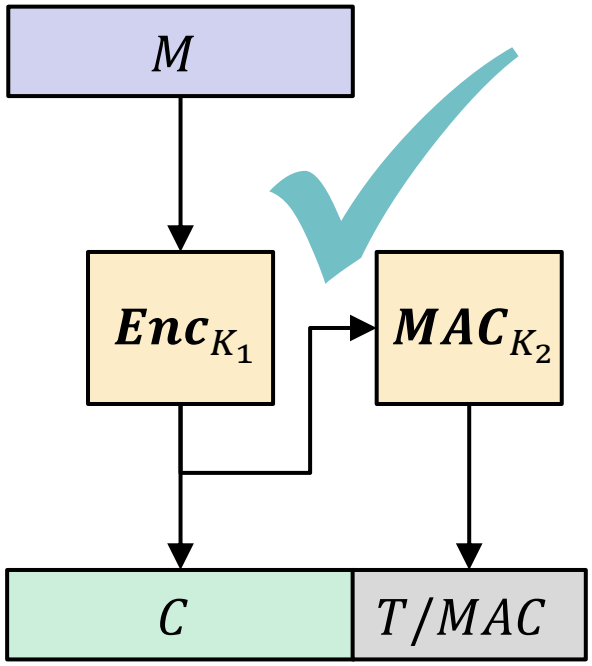
Used in TLS 1.2

MAC-then-Encrypt (MtE)



Used in SSH

Encrypt-then-MAC (EtM)



MAC must cover all of C (including IVs)
Used in IPsec

Confidentiality, Integrity, and Authenticity

- *MAC* is checked first
  - The ciphertext is discarded if the *MAC* is invalid
- Why not use the same key?
  - Reusing a single key introduces unnecessary risks, breaking cryptographic best practices and potentially leading to vulnerabilities
- Using different keys to encrypt and compute *MAC* works, even if keys are related
  - For instance, they can be derived from a single key K via $K_e = F_K(0)$ and $K_m = F_K(1)$, where F is a PRF such as a block cipher, the CBC-MAC, or HMAC
  - But, twice as much work as encryption alone
  - Slow; needs 2 block cipher calls per message block
  - Can do a little better — about 1.5 "encryptions"
    - Such as AES-GCM (Galois/Counter Mode, based on EtM) and AES-CCM (Counter with CBC-MAC, based on E&M), modern AE schemes
- Confidentiality and integrity with same work as one encryption is a research topic

Uses for Symmetric Crypto

- Confidentiality
  - Transmitting data over insecure channel
  - Secure storage on insecure media
- Integrity (*MAC*)
- Authentication protocols (later…)
- Anything you can do with a hash function (upcoming chapter…)

Slides Original Source

- Jonathan Katz and Yehuda Lindell, “Introduction to Modern Cryptography,” Third Edition, 2021
- M. Stamp, “Information Security: Principles and Practice,” John Wiley
- B. Forouzan, “Cryptography and Network Security,” McGraw-Hill
- C. Paar and J. Pelzl, “Understanding Cryptography – A Textbook for Students and Practitioners,” Springer (www.crypto-textbook.com)