Message Authentication Code

CS 5158/6058 Data Security and Privacy

Spring 2018

Instructor: Boyang Wang

Block Cipher: CBC Mode

- Cipher Block Chaining (CBC)
  - Probabilistic, is CPA-secure if F is a PRP
  - IV (initialization vector) chosen uniformly from $\{0,1\}^n$

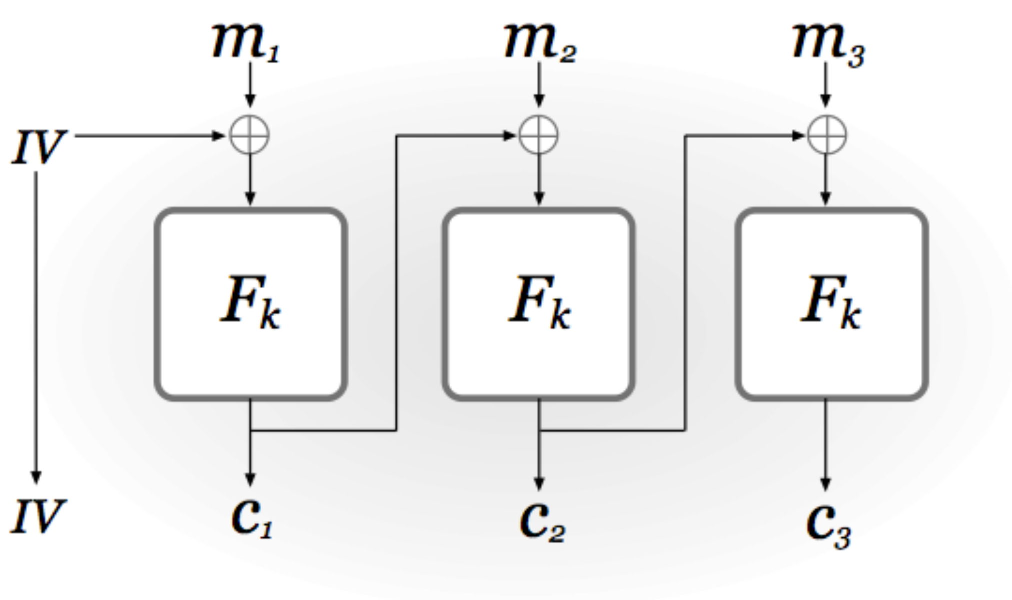


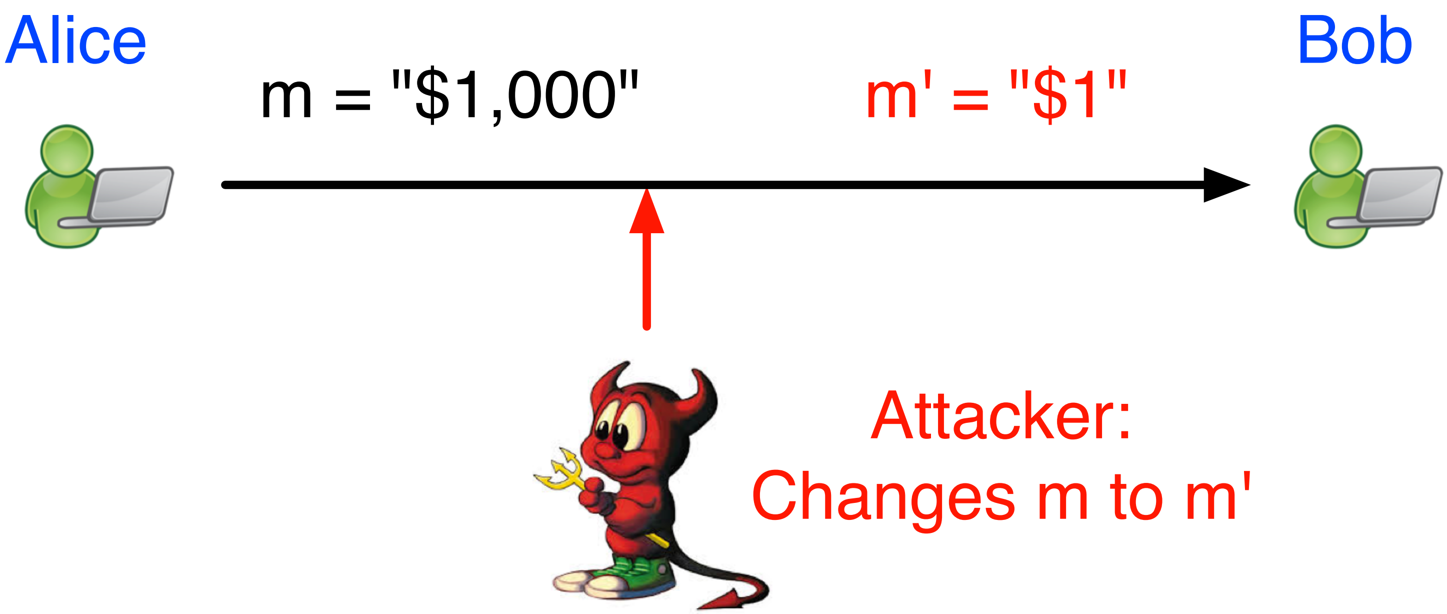
FIGURE 3.7: Cipher Block Chaining (CBC) mode.

$$c_0 = IV,$$
$$c_i = F_k(m_i \oplus c_{i-1})$$

$$c_0 = IV,$$
$$m_i = F_k^{-1}(c_i) \oplus c_{i-1}$$

Message Authentication

- Attacker modifies messages from Alice to Bob



- Bob needs to prove a message is correct (i.e., unchanged)

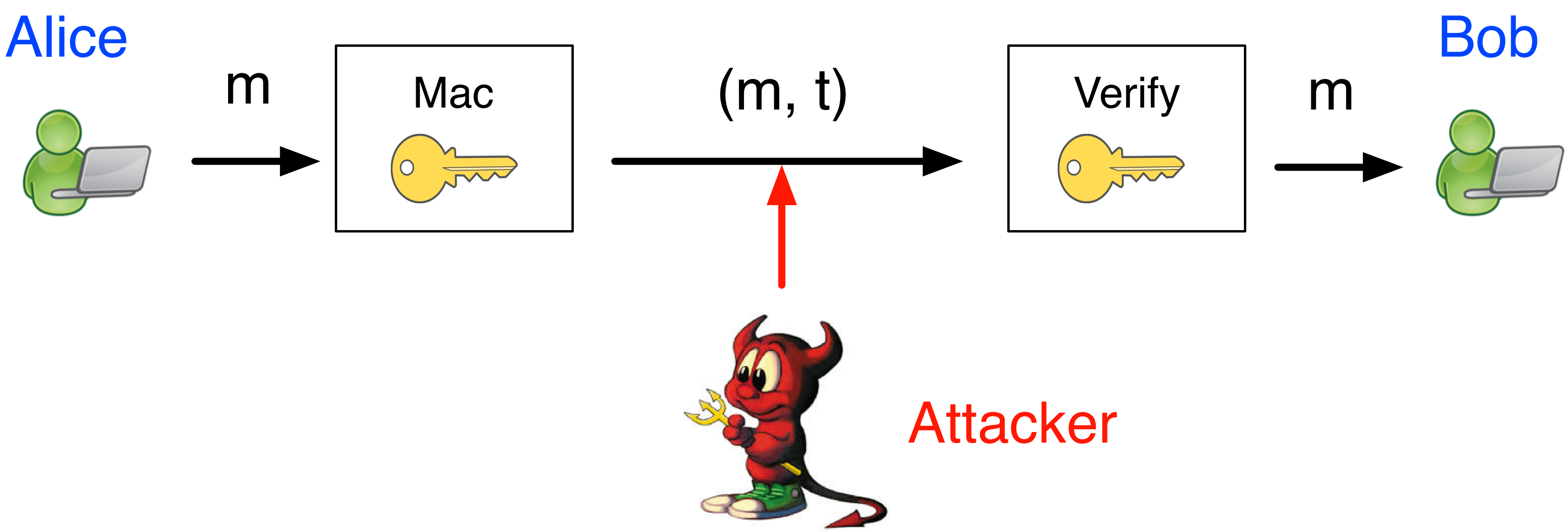
Message Authentication

- Attacker fakes messages from Alice to Bob

- Bob needs to prove a message is authentic, i.e., from Alice

Message Authentication Code

- Alice computes a tag for a message
- Bob can verify a message m using its tag t
  - If valid, accept a message
  - Otherwise, drop or ignore a message



Message Authentication Code

Message Authentication Code (MAC)

- KeyGen: takes a security parameter 1^n as input, outputs key k

- Mac: takes a key k and a message $m \in \{0,1\}^*$ as input, outputs a tag t

- Verify: takes a key k, a message m, and a tag t, outputs 1 if $\mathsf{Mac}_k(m) == t$, otherwise outputs 0.

Mac is deterministic (i.e., same m, same t)

Message Authentication Code

- Alice sends (m="$1000", t) to Bob
- Bob receives (m, t), verifies m by checking tag t
  - If m="$1000", then accept
  - If m= "$999", then drop it

- Basic requirements for <u>message authentication</u>
  1. Something (a secret) only Alice and Bob know
  2. Changing messages can be easily detected

Message Authentication

- How about simply use Encryption?
  1. Alice and Bob share a secure key
  2. An attacker cannot easily get a new meaningful message by a changing ciphertext.

- If use Encryption as Message Authentication
  - Alice only sends c to Bob, Bob decrypts c, if it is meaningful, then m is authentic; otherwise, it is not.

Message Authentication

- E.g., m = **`Tu`**, c = **`0x307aed45`**
  - Attacker changes ciphertext to c' = **`0x307aed46`**
  - Decryption of c' will not be {**`M`**, **`Tu`**, **`W`**, **`Th`**, **`F`**}

- However, data is not always meaningful
  - E.g., data is simply a binary string.
  - Alice sends c to Bob, attacker changes to c', the decryption of c' is m' a valid binary string, Bob will take m' as a valid message, but should be m.
  - Encryption cannot authenticate messages

x	00	01	10	11
k=00,F(x)	11	00	01	10
k=01,F(x)	10	11	00	01
k=10,F(x)	01	10	11	00
k=11,F(x)	00	01	10	11

- <u>Use C as a tag:</u>
- CBC Mode Decryption:
  - C = (IV, 100110)
  - k = 11, IV = 00
  - each block has 2 bits
  - M = 10 11 11

$$c_0 = IV, m_i = F_k^{-1}(c_i) \oplus c_{i-1}$$

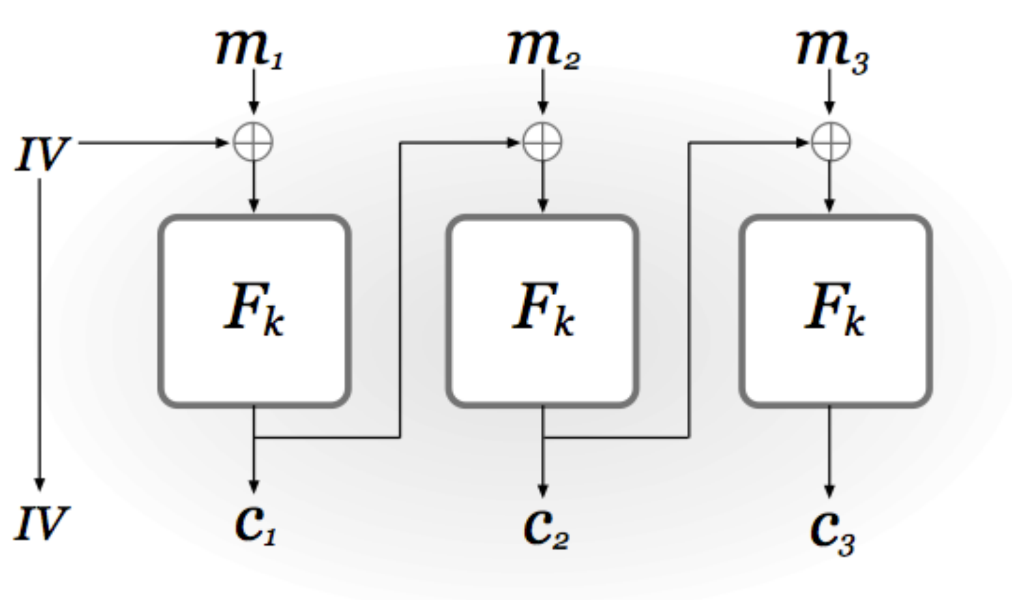


FIGURE 3.7: Cipher Block Chaining (CBC) mode.

x	00	01	10	11
k=00,F(x)	11	00	01	10
k=01,F(x)	10	11	00	01
k=10,F(x)	01	10	11	00
k=11,F(x)	00	01	10	11

$$c_0 = IV, m_i = F_k^{-1}(c_i) \oplus c_{i-1}$$

C = (00, 100110) & k = 11

- $F_k^{-1}(10) = 10$, $m_1 = F_k^{-1}(10) \oplus IV = 10 \oplus 00 = 10$
- $F_k^{-1}(01) = 01$, $m_2 = F_k^{-1}(01) \oplus c_1 = 01 \oplus 10 = 11$
- $F_k^{-1}(10) = 10$, $m_3 = F_k^{-1}(10) \oplus c_2 = 10 \oplus 01 = 11$
- M = 10 11 11

x	00	01	10	11
k=00,F(x)	11	00	01	10
k=01,F(x)	10	11	00	01
k=10,F(x)	01	10	11	00
k=11,F(x)	00	01	10	11

- Change one bit in C
- CBC Mode Decryption:
    - C’ = (IV, 100111)
    - k = 11, IV = 00
    - each block has 2 bits
    - M’ = 10 11 10

$$c_0 = IV, m_i = F_k^{-1}(c_i) \oplus c_{i-1}$$

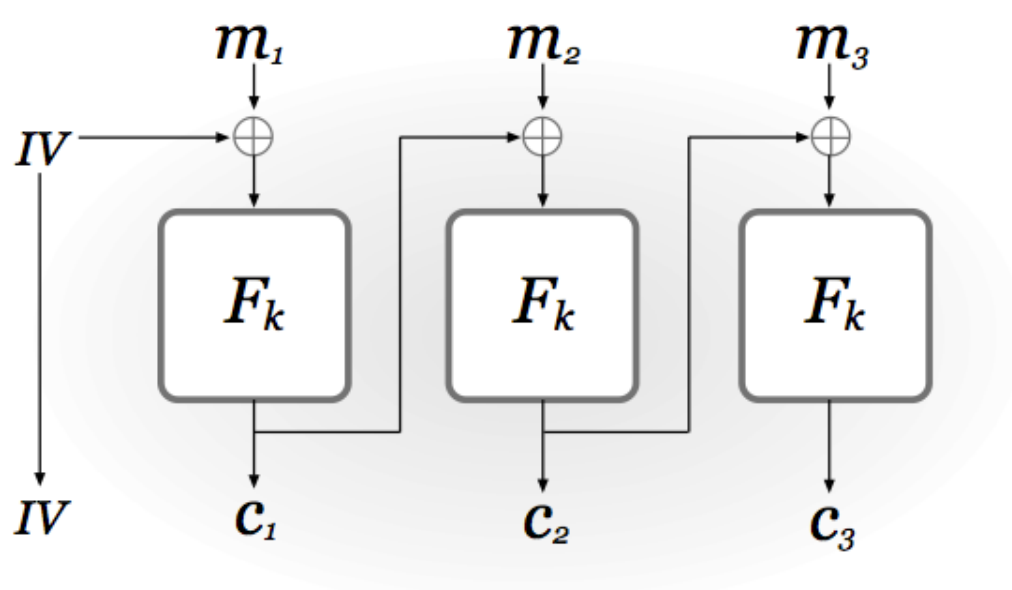


FIGURE 3.7: Cipher Block Chaining (CBC) mode.

x	00	01	10	11
k=00,F(x)	11	00	01	10
k=01,F(x)	10	11	00	01
k=10,F(x)	01	10	11	00
k=11,F(x)	00	01	10	11

$$c_0 = IV, m_i = F_k^{-1}(c_i) \oplus c_{i-1}$$

C' = (00, 100111) & k = 11

- $F_k^{-1}(10)$ = 10, $m_1 = F_k^{-1}(10) \oplus IV = 10 \oplus 00 = 10$
- $F_k^{-1}(01)$ = 01, $m_2 = F_k^{-1}(01) \oplus c_1 = 01 \oplus 10 = 11$
- $F_k^{-1}(10)$ = 10, $m_3 = F_k^{-1}(11) \oplus c_2 = 11 \oplus 01 = 10$
- M' = 10 11 10
- Bob takes M' = 101110, but should be M =101111

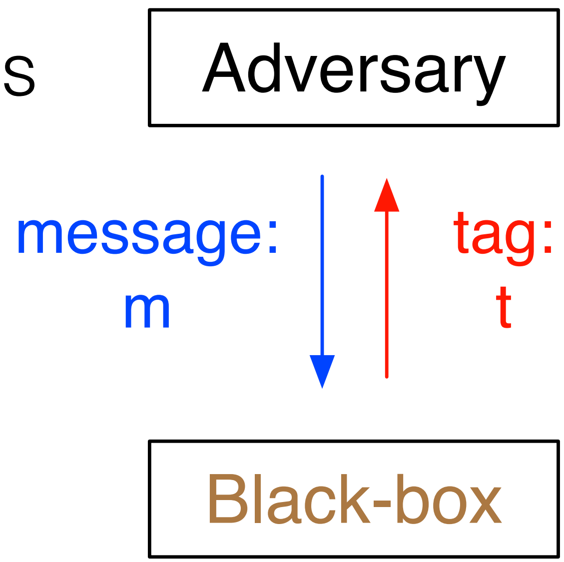
Assumptions on Adversary

- Assumptions on an <u>adversary</u>:
  - Knows messages (is not about privacy)
  - Knows Mac and Verify algorithm,
  - Can eavesdrop
    - Can collect previous message-tag pairs
  - Does not know key k

  - Has <u>limited</u> computational power
    - Run efficient (polynomial-time) algorithms

Security of MAC

- <u>Unforgeable</u> under chosen-message attacks
  - A PPT adversary has access to a <u>MAC oracle</u>
  - Submits a message m, obtains a valid tag t
  - Unlimited number of queries
  - Can have many message-tag pairs
  - E.g., $(m_1, t_1), (m_2, t_2), \ldots, (m_n, t_n)$



Security of MAC

- Unforgeable under chosen-message attacks
  - A PPT adversary has access to a MAC oracle
  - Adversary cannot generate a valid tag t' for a "new" message m'
    - Message m' that was not submitted to oracle
    - Verify(m', t')=1 happens with a negligible probability

Security of MAC

- Example: Alice sent 3 messages with tags to Bob
  - (`packers, 0x34dt`)
  - (`patriots,0xd5ac`)
  - (`eagles,  0xa70b`)

- Adversary learns above 3 message-tag pairs, sends
  - (`patriots,0xd5ac`) to Bob, Bob takes it
  - (`patriots,0xd5ab`) to Bob, Bob drops it
  - (`bengals,  0x1234`) to Bob, Bob drops it

Security of MAC

- Practice: Alice sent 3 messages with tags to Bob
    - (`packers, 0x34dt`)
    - (`patriots,0xd5ac`)
    - (`eagles,  0xa70b`)

- Adversary sends
    - (`patriot,0xd5ac`) to Bob, Bob ??
    - (`steelers,0x1234`) to Bob, Bob ??
    - (`packers,0xa70b`) to Bob, Bob ??

Security of MAC

A security game $\mathsf{MacForge}_{\mathcal{A},\Pi}(n)$:

1. A key k is generated by running $\mathsf{KeyGen}(1^n)$
2. Adversary $\mathcal{A}$ has access to a MAC oracle $\mathsf{Mac}_k(\cdot)$. Let $\mathcal{Q}$ denotes the set of all queries $\mathcal{A}$ submitted to the oracle. Eventually, $\mathcal{A}$ outputs (m', t'), where $m' \notin \mathcal{Q}$
3. $\mathcal{A}$ outputs 1 if $\mathsf{Verify}(m', t') = 1$, and outputs 0 otherwise

$$\Pr[\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathtt{negl}(n)$$

Security of MAC

- MAC does not prevent replay attacks
  - If (m,t) has been sent before, send (m,t) again
    - Alice —> (“Pay $1,000”, t) —> Bob
    - Attacker —> (“Pay $1,000”, t) —> Bob
    - Both valid, and $2,000 was paid in total.
    - But Bob only needs to pay $1,000
- Two possible solutions
  - Counters, but messages could drop
  - Time stamps, synchronize clocks may not be easy

A MAC from PRF

Build a fixed-length MAC for n-bit messages

- KeyGen: outputs a key k, where $k \xleftarrow{u} \{0,1\}^n$
- Mac: given a key k and a message $m \in \{0,1\}^n$, outputs a tag $t \leftarrow F_k(m)$, where F is a PRF
- Verify: given a key k, a message m and a tag $t \in \{0,1\}^n$, outputs 1 if $F_k(m) == t$, otherwise outputs 0

Guessing a valid tag for a new message is equivalent of guessing an output of PRF, which is negligible

x	00	01	10	11
k=00,F(x)	11	00	01	10
k=01,F(x)	10	11	00	01
k=10,F(x)	01	10	11	00
k=11,F(x)	00	01	10	11

$\mathsf{Mac} : t \leftarrow F_k(m)$

$\mathsf{Verify} : F_k(m) \stackrel{?}{=} t$

- <u>Practice:</u> given key k = 10
- Q1: m = 10, what is its tag t = ??
- Q2: given (m=11, t=01), is it valid?
- Q3: given (m=01, t=10), is it valid?
- Q4: given (m=10, t=10), is it valid?

x	00	01	10	11
k=00,F(x)	11	00	01	10
k=01,F(x)	10	11	00	01
k=10,F(x)	01	10	11	00
k=11,F(x)	00	01	10	11

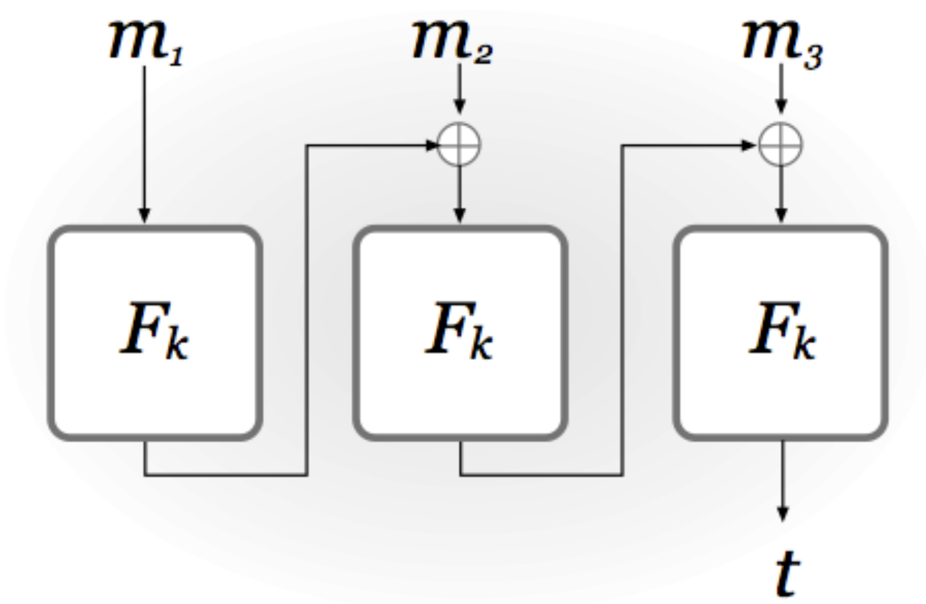
$$\text{Mac}: t \leftarrow F_k(m)$$

$$\text{Verify}: F_k(m) \stackrel{?}{=} t$$

- Practice: given key k = 10
- Q1: m = 10, tag t = 11
- Q2: given (m=11, t=01), not valid (t should be 00)
- Q3: given (m=01, t=10), valid
- Q4: given (m=10, t=10), not valid (t should be 11)

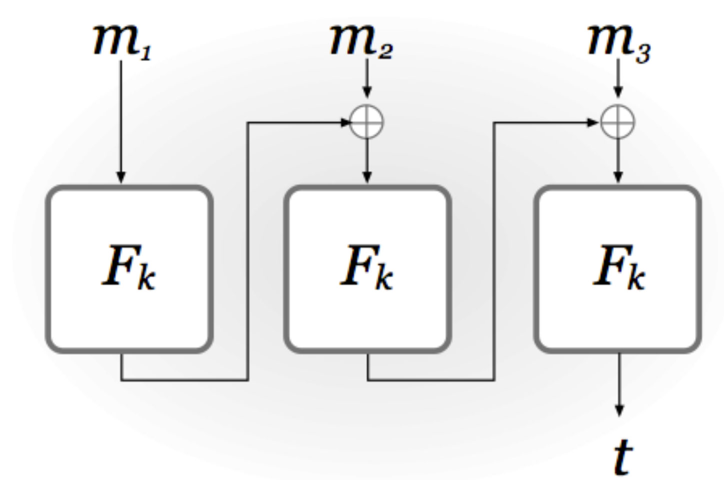
CBC-MAC

Build a fixed-length MAC for l*n-bit messages



- 1 message, multiple blocks, 1 tag
- Secure if F is a PRF

x	00	01	10	11
k=00,F(x)	11	00	01	10
k=01,F(x)	10	11	00	01
k=10,F(x)	01	10	11	00
k=11,F(x)	00	01	10	11



- Practice: given key k = 10
- Q1: m = 101110, tag t = ??
- Q2: (m=1101, t=01), is it valid?
- Q3: (m=1011, t=01), is it valid?

x	00	01	10	11
k=00,F(x)	11	00	01	10
k=01,F(x)	10	11	00	01
k=10,F(x)	01	10	11	00
k=11,F(x)	00	01	10	11

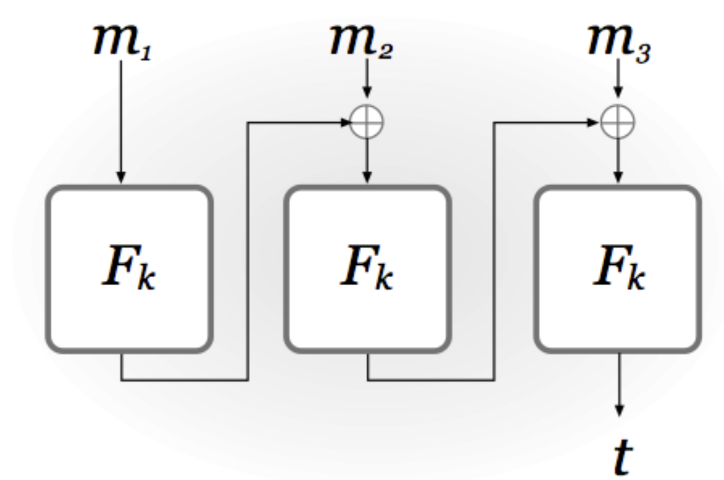
Given key k = 10

Q1: m = 101110, tag t = ??

$F_k(10)=11$, $11 \oplus m_2 = 00$

$F_k(00)=01$, $01 \oplus m_3 = 11$

$F_k(11)=00$, t = 00



x	00	01	10	11
k=00,F(x)	11	00	01	10
k=01,F(x)	10	11	00	01
k=10,F(x)	01	10	11	00
k=11,F(x)	00	01	10	11

Given key k = 10

Q2: (m=1101, t=01), is it valid?

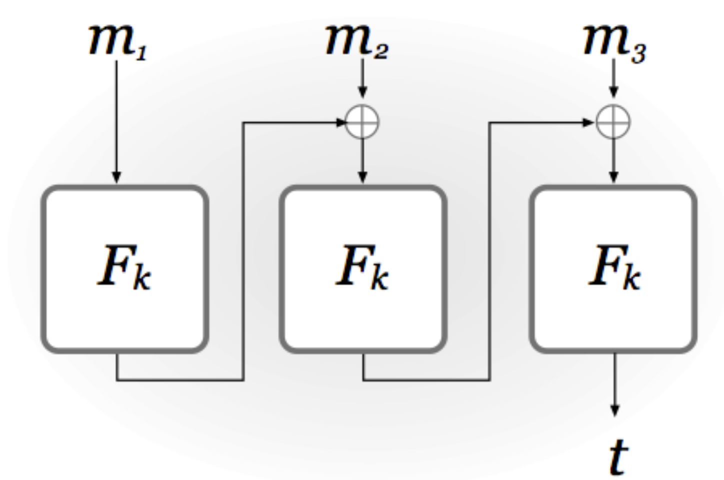
$F_k(11)=00$, $00 \oplus m_2 = 01$

$F_k(01)=10$, t=10!=01

Q3: (m=1011, t=01), is it valid?

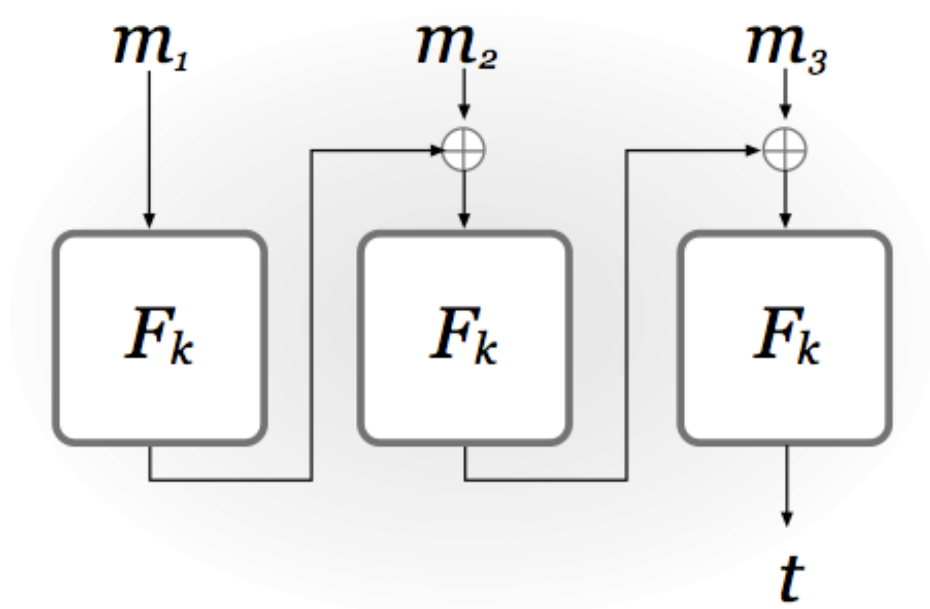
$F_k(10)=11$, $11 \oplus m_2 = 00$

$F_k(00)=01$, t=01==01



CBC-MAC

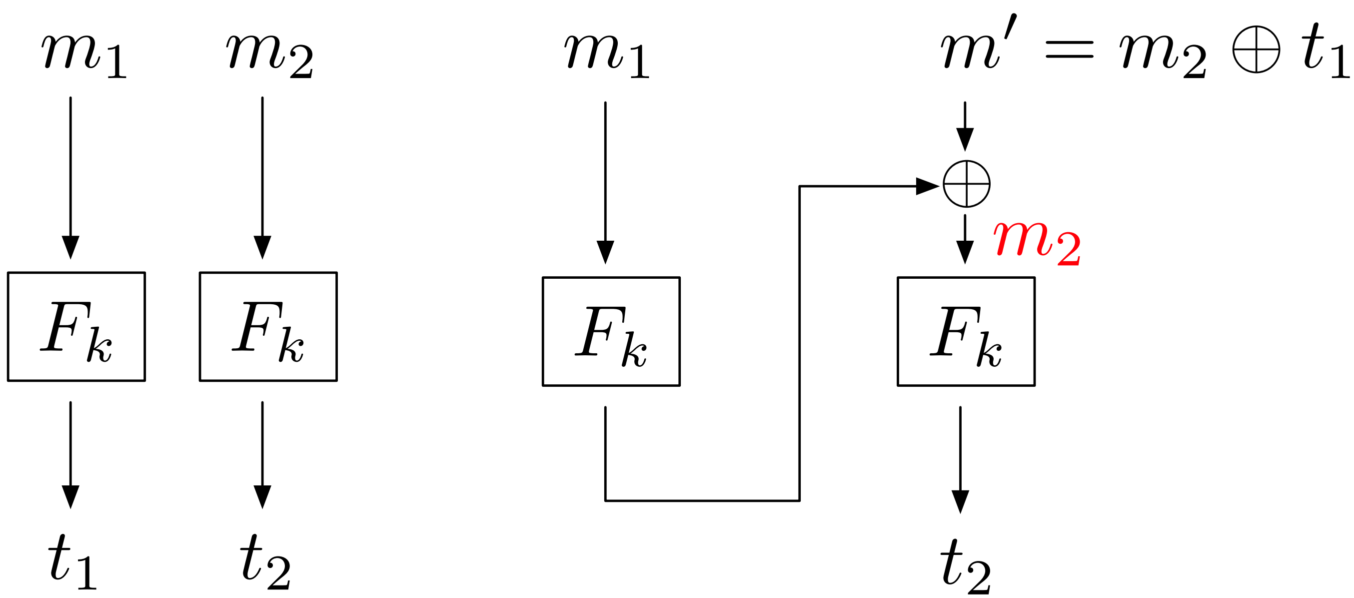
Build a fixed-length MAC for l*n-bit messages

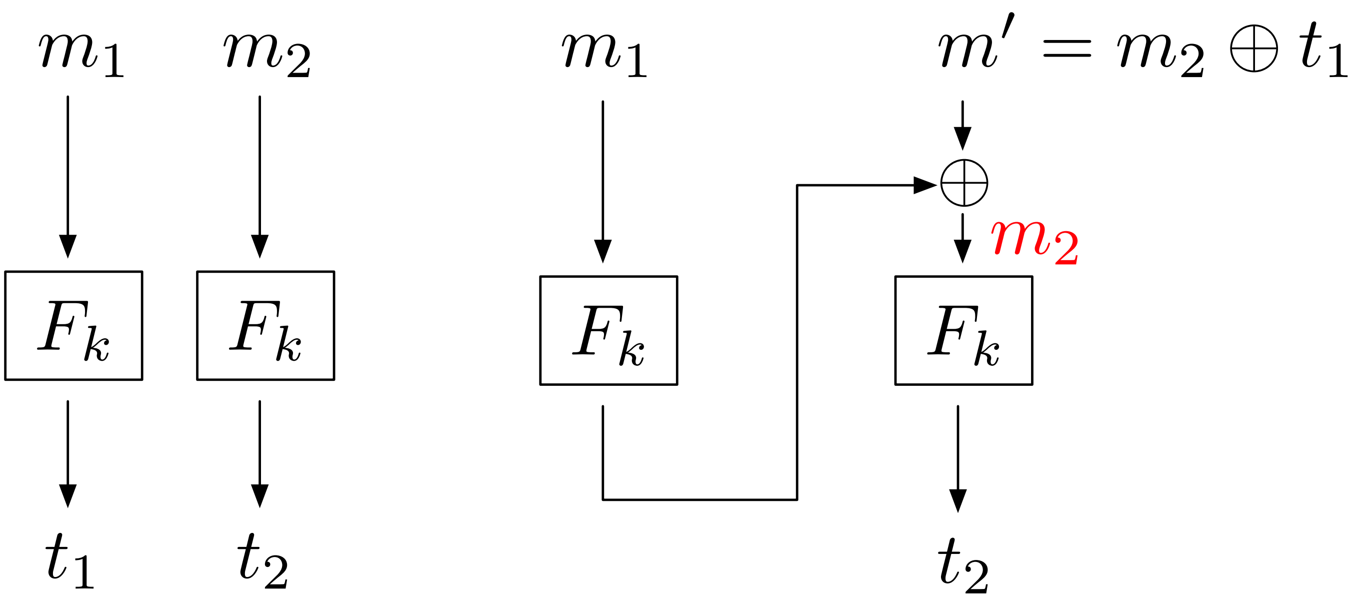


- Secure if F is a PRF, only for l*n-bit messages
  - Not secure for messages with arbitrary length

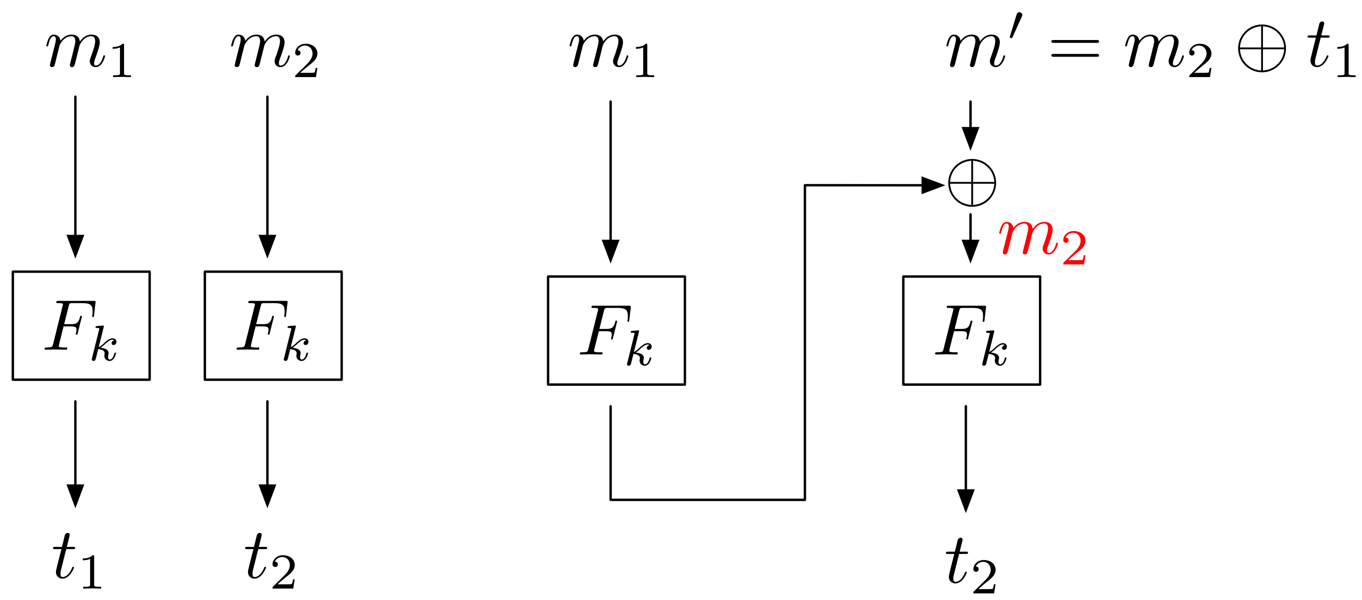
CBC-MAC

- l=1, adversary already has (m_1, t_1) and (m_2, t_2)
- Fakes a message $m = m_1 m'$, where $m' = m_2 \oplus t_1$
  - m is a "new" message, should not be valid
  - But (m, t_2) will pass the verification





- Example: Adversary knows $(m_1,t_1) = (10,11)$, $(m_2,t_2) = (01,10)$
  - compute $m' = m_2 \oplus t_1 = 01 \oplus 11 = 10$
  - fake "new" $m = m_1 m' = 1010$, send (m,t_2) to Bob
  - Bob will accept (m,t_2), since Verify(m,t_2) will output Yes



- Practice: Adversary knows $(m_1,t_1) = (1011,1110)$, $(m_2,t_2) = (0110,1000)$
  - fake a "new" message m, s.t. (m, t_1) will pass
  - computes $m'=m_1 \oplus t_2 = 1011 \oplus 1000 = 0011$
  - $m=m_2m'=01100011$
  - $(m, t_1) = (01100011, 1110)$

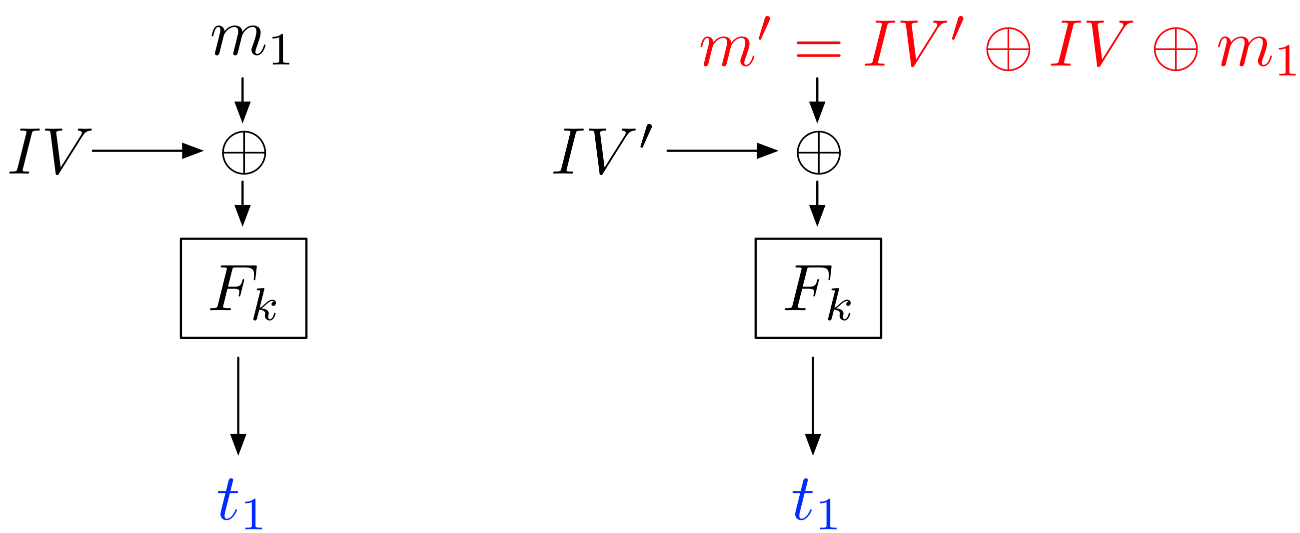
CBC-MAC

- CBC-MAC v.s. CBC-ENC

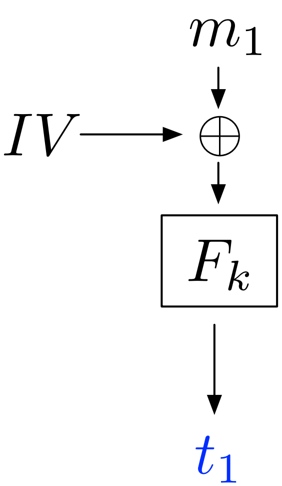
	CBC-MAC	CBC-ENC
Goal	Message Auth.	Encryption
Key	n bits	n bits
Message	l*n bits	l*n bits
IV	No IV	Random IV
Output	n bits	l*n bits

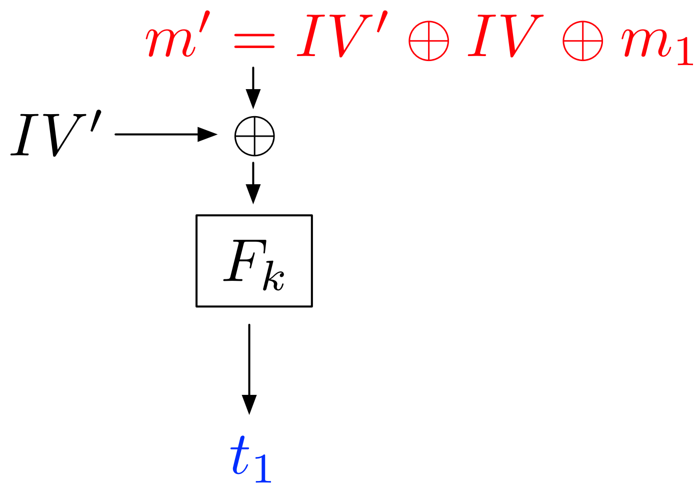
CBC-MAC

- Adversary only has (m_1, IV, t_1)
- Fakes a message m’=IV’⊕IV⊕m_1, where IV’ is “new”
  - m’ is a “new” message, should not be valid
  - But (m’, IV’, t_1) will pass the verification



- <u>Practice</u>: assume random IVs are used in CBC-MAC
  - Adversary knows (m, IV, t) is valid, where message m is 11010010, its IV is 11000011, tag is t.
  - Given another IV’ = 00001111, create a “new” message m’, s.t. (m’, IV’, t) will pass





$$\begin{array}{cc} & 11010010 \\ \oplus & 11000011 \\ \oplus & 00001111 \\ = & 00011110 \end{array}$$

Additional Reading

Chapter 4, *Introduction to Modern Cryptography, Drs. J. Katz and Y. Lindell, 2nd edition*