CS 4770: Cryptography

CS 6750: Cryptography and Communication Security

Alina Oprea
Associate Professor, CCIS
Northeastern University

February 12 2018

Announcements

- Schedule
  - Next week vacation on Monday (President's Day)
  - Class canceled on Thursday 02/22
  - Normal schedule on Monday 02/26
- Assignments
  - HW 2 due on Thu 02/15
  - Programming project Thu 02/15 – Mon 02/26
- Midterm exam
  - Thursday 03/01
  - Topics
    - Notions of security for encryption (PS, EAV, CPA, CCA)
    - Modes of operation for encryption (CBC, CTR)
    - PRG, PRF, PRP
    - MAC for integrity
    - Authenticated encryption

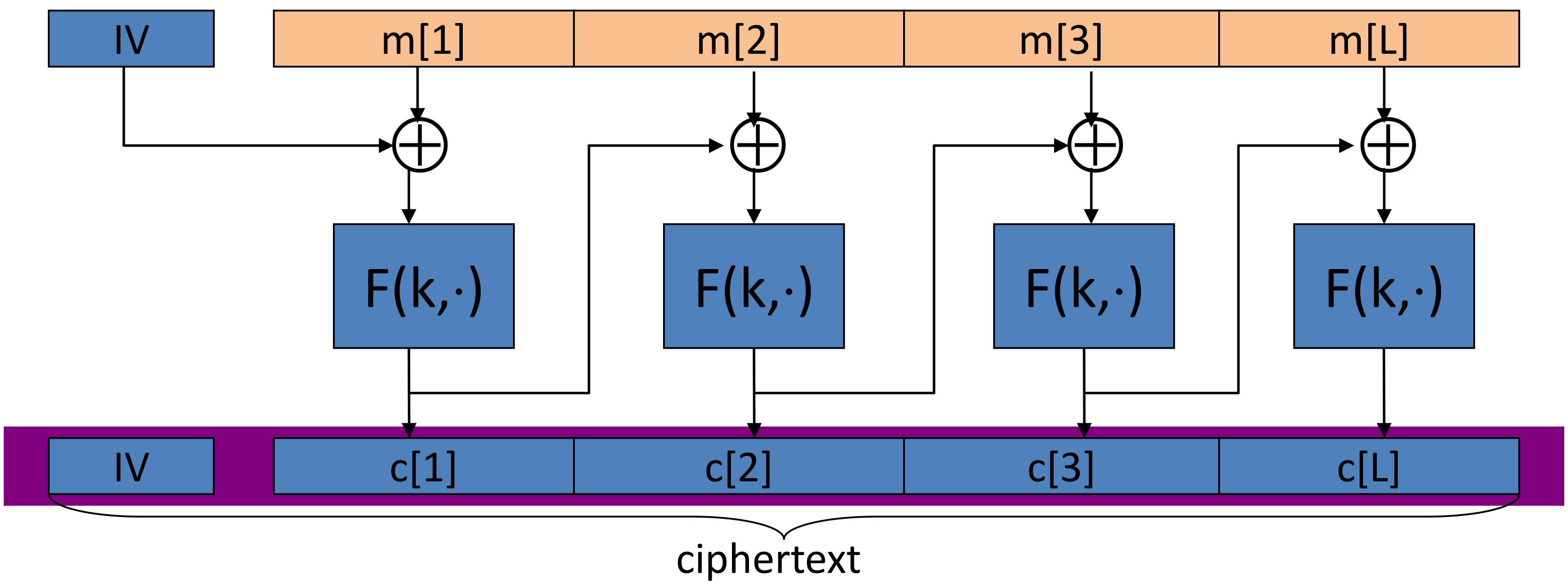
Recap

- To encrypt longer messages, use *CBC or CTR mode*
  - Both have CPA security
  - IV needs to be randomized
- CTR mode has some advantages
  - *Parallelizable*
  - *Better security*
- CBC encryption has padding vulnerabilities
- Authenticated encryption schemes are CCA secure
  - Will study them soon

CBC encryption

Let F be a PRP; F: K × $\{0,1\}^n \longrightarrow \{0,1\}^n$

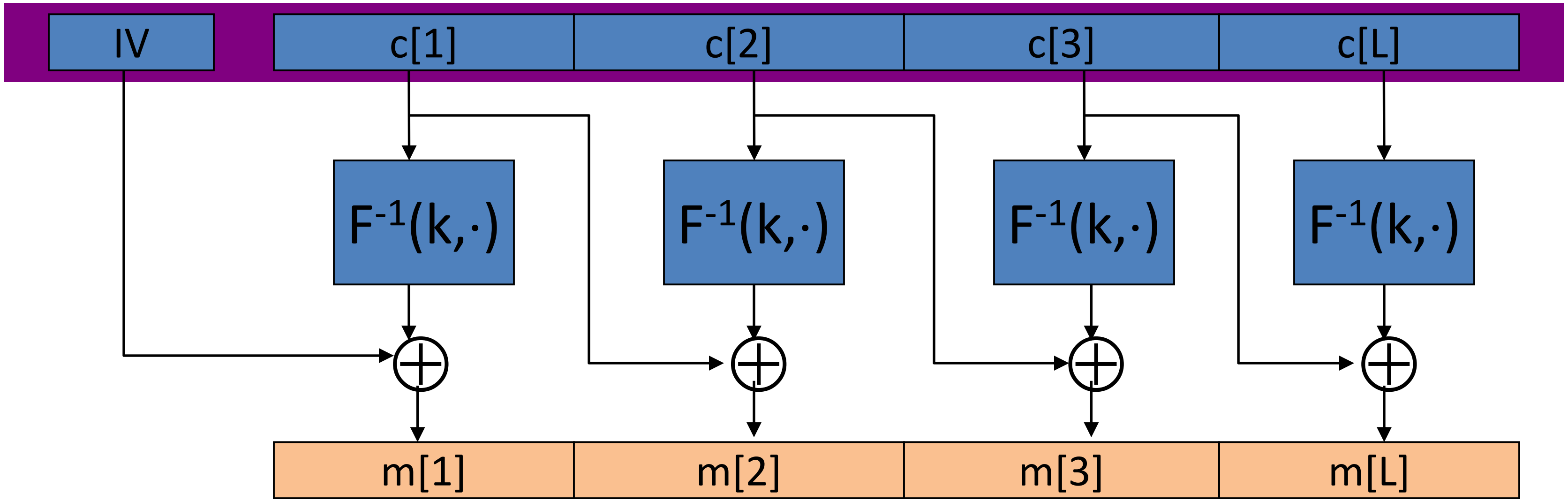
Enc_{CBC}(k,m): choose **random** IV$\in \{0,1\}^n$ and do:



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

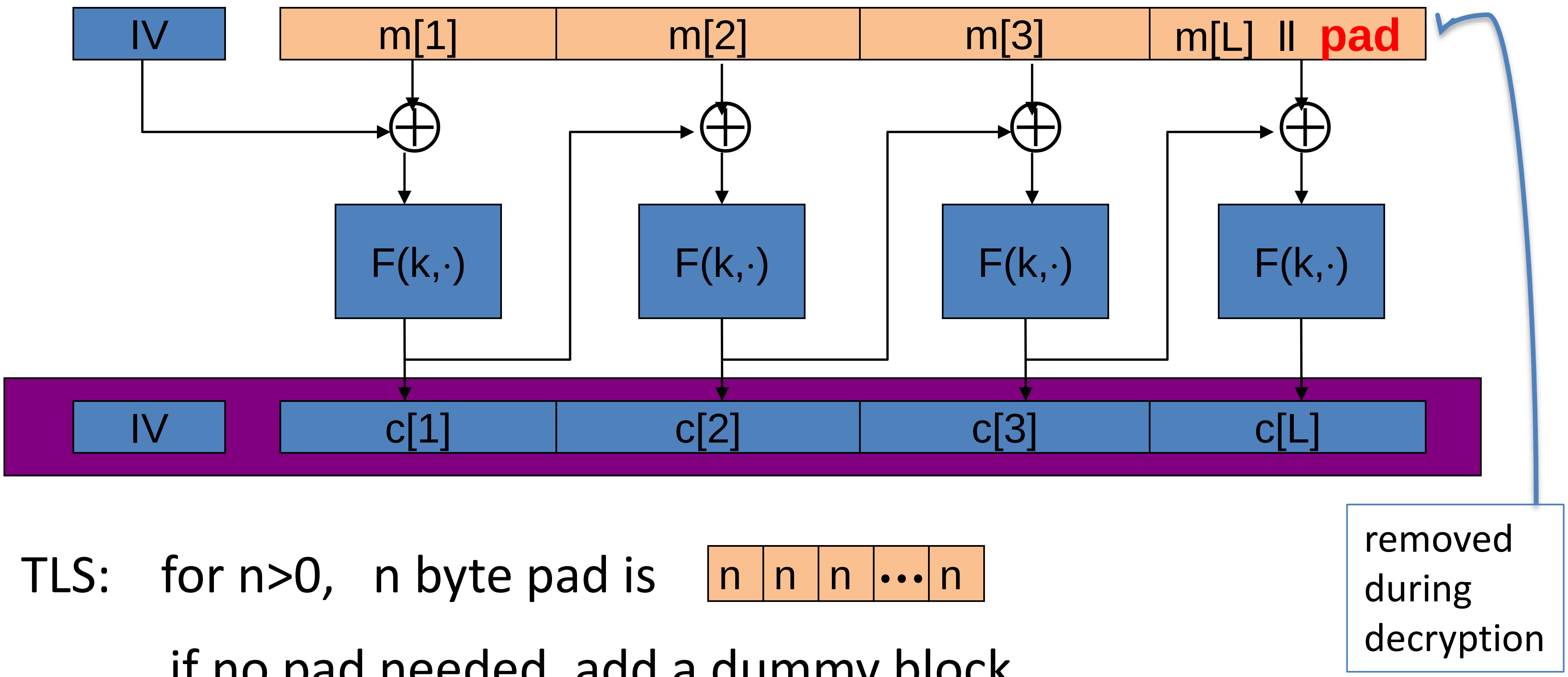
Decryption circuit

In symbols: c[1] = F_k(IV⊕m[1]) ⇒ m[1] =



$$m_i = F^{-1}{}_k(c_i) \oplus c_{i-1}$$

A CBC technicality: padding



TLS: for n>0, n byte pad is | n | n | n | ••• | n |

if no pad needed, add a dummy block

TLS bugs in older versions

IV for CBC is predictable: (chained IV)

- IV for next record is last ciphertext block of current record.
- Not CPA secure.

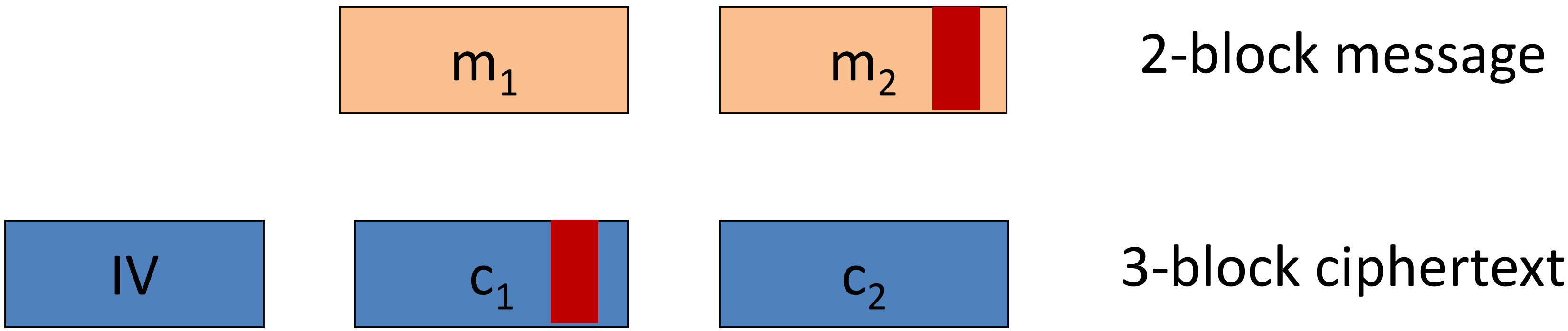
Padding oracle: during decryption

- If pad is invalid send decryption failed alert
- If mac is invalid send bad_record_mac alert

⇒ attacker learns information about plaintext

Lesson: when decryption fails, do not explain why

Padding oracle attack



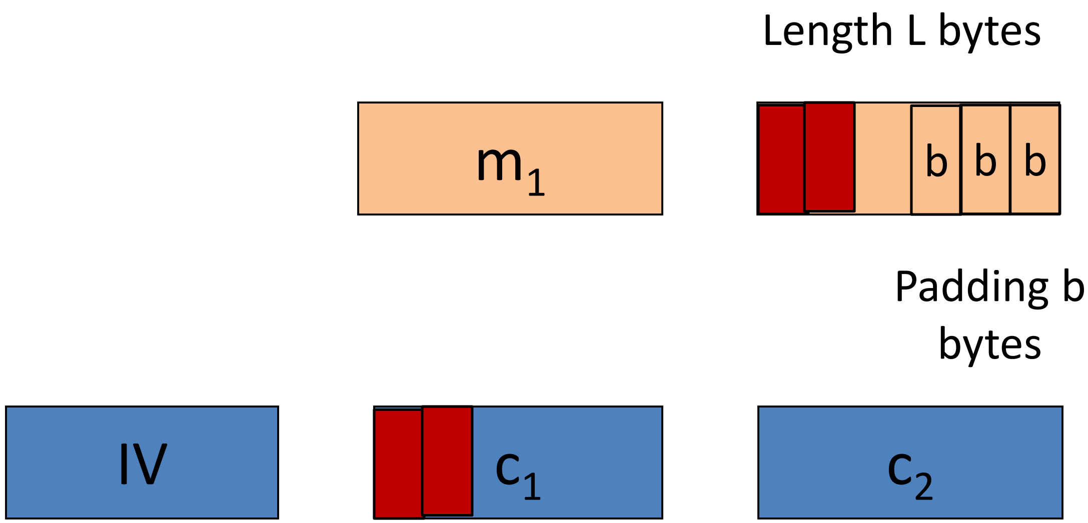
- Attacker can query ciphertexts to padding oracle
- Oracle responds with “bad padding” if message not correctly padded
- Goal: given ciphertext, find last block of message

$$c_2 = F_k(c_1 \oplus m_2)$$
$$m_2 = F_k^{-1}(c_2) \oplus c_1$$

$$c_1' = c_1 \oplus \Delta$$
$$m_2' = m_2 \oplus \Delta$$

Malleability

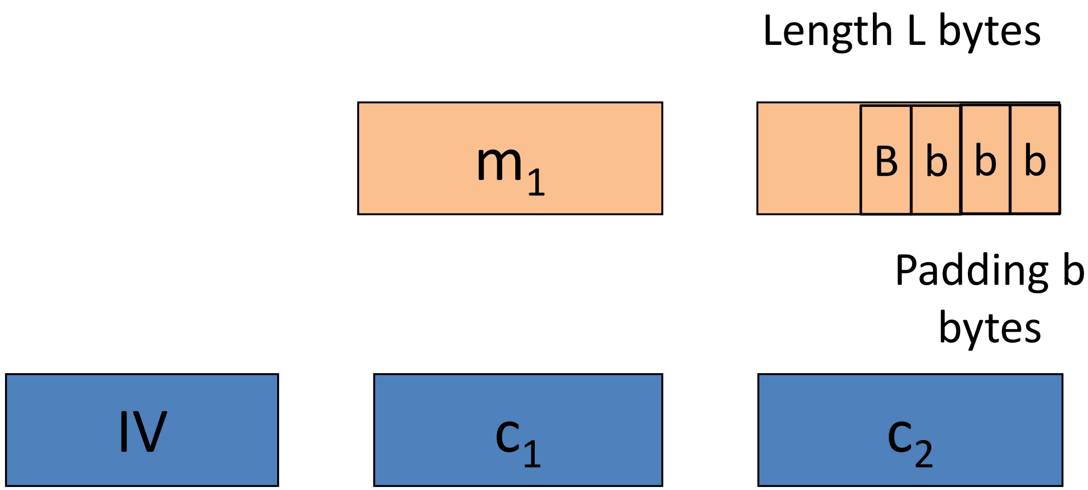
Find message length



Decryption fails if last b bytes do not have value b

- Modify first byte of c_1
- If decryption fails, then oracle checks all L bytes of m_2, thus b=L
- Else modify second byte of c_1
- If decryption fails, then b = L-1
- Continue until find b

Find message bytes



Decryption fails if last b bytes do not have value b

- Learn last byte B of m_2 (before padding)
  - Intuition: Induce a valid message of length b+1
  - For all i:
    - $\Delta_i = 0 \ldots 0\, i\big((b+1) \oplus b\big) \ldots \big((b+1) \oplus b\big)$
  - Query $c'_1 = c_1 + \Delta_i$ to padding oracle
  - But $m'_2 = m_2 + \Delta_i = 0 \ldots 0\, (B \oplus i)(b+1) \ldots (b+1)$
  - If $B \oplus i = b + 1$, decryption succeeds
- Exercise: extend it to recover all bytes from last block

Integrity

- Active adversaries
  - Can modify messages/ciphertexts in transit
- Protect message integrity
  - Message received by Bob is the original one sent by Alice
  - Message was not modified by adversary
- Scenarios
  - Secure communication on network
  - Protect files stored on disk

Message Authentication

Alice

m

k

(m, t=Tag$_k$(m))

Bob

verifies if t=Tag$_k$(m)

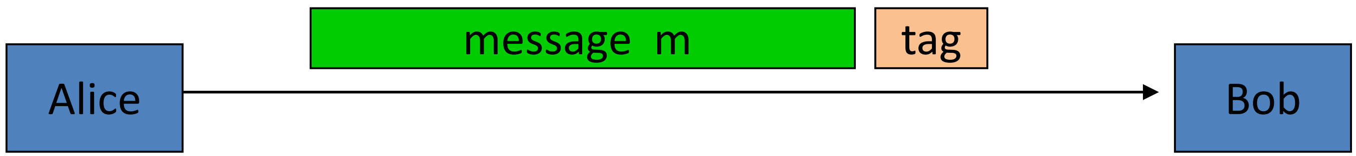
k

- Message was sent by Alice
- Message was not modified

Eve can see **(m, t=Tag$_k$(m))**

She should not be able to compute a valid tag **t'** on any other message **m'**.

Integrity requires a secret key



Generate tag:

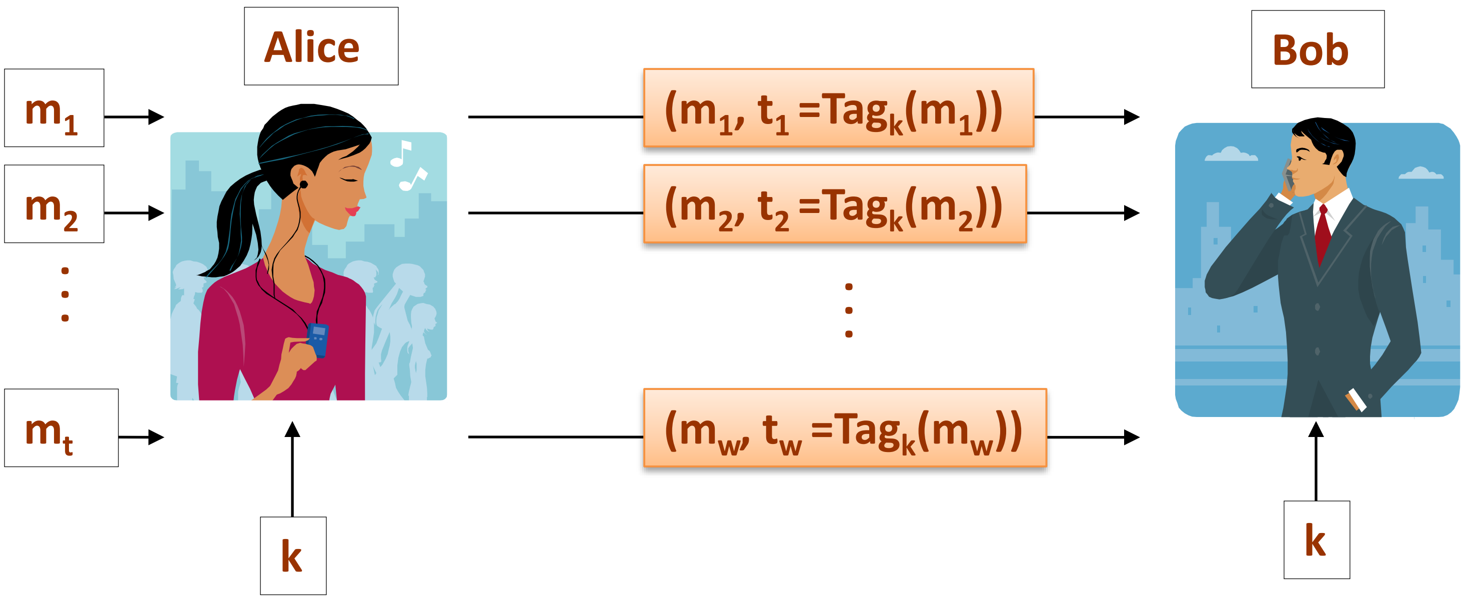
tag ← CRC(m)

Verify tag:

Ver(m, tag) $\stackrel{?}{=}$ 'yes'

- Attacker can easily modify message m and re-compute CRC.
- CRC designed to detect **<u>random</u>**, not malicious errors.

Message authentication – multiple messages



Eve should not be able to compute a valid tag **t’** on any other message **m’**.

A mathematical view

$\mathcal{K}$ – **key** space

$\mathcal{M}$ – **plaintext** space

$\mathcal{T}$ - set of **tags**

A **Message Authentication Code (MAC) scheme** is a pair **(Tag, Ver)**, where

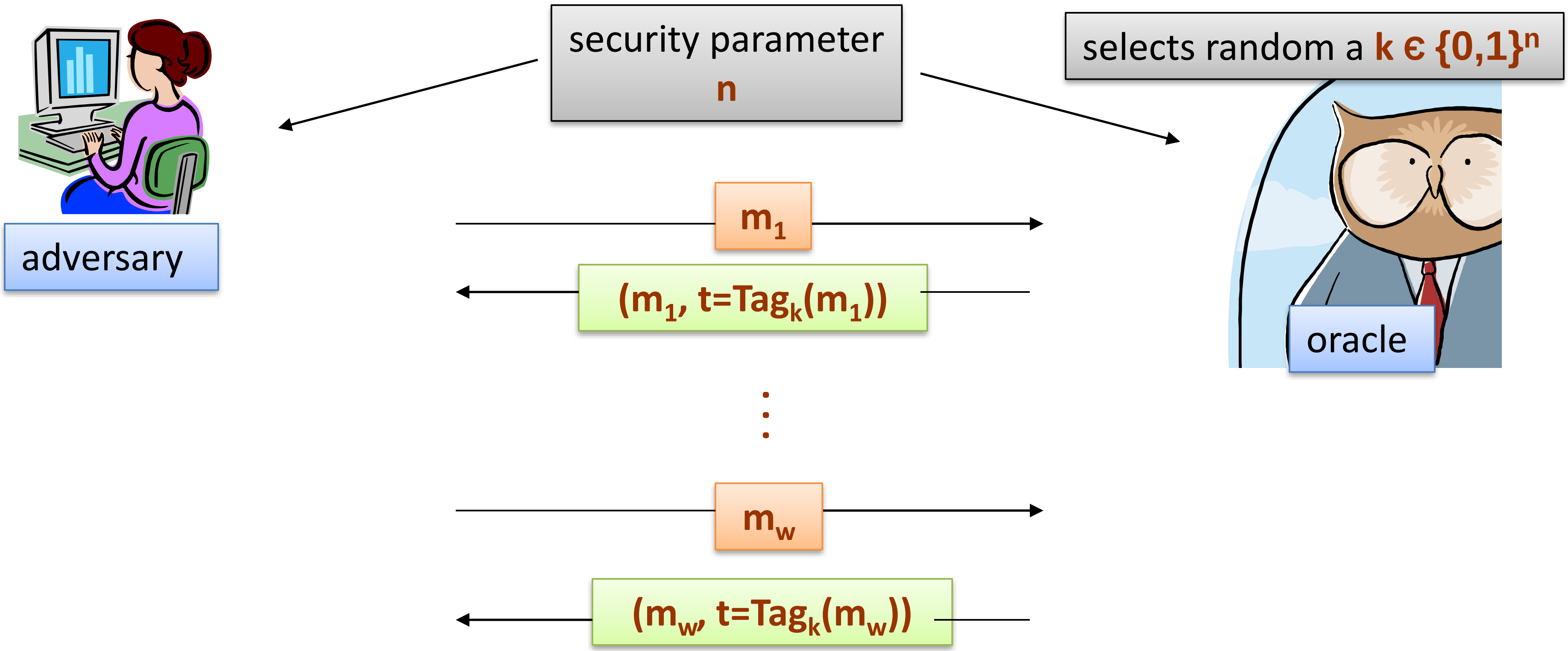
- **Tag :** $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$ is an **tagging** algorithm,
- **Ver:** $\mathcal{K} \times \mathcal{M} \times \mathcal{T} \rightarrow$ **{yes, no}** is a **verification** algorithm.

We will sometimes write **$Tag_k(m)$** and **$Ver_k(m,t)$** instead of **Tag(k,m)** and **Ver(k,m,t)**.

Correctness

it should always holds that:

$Ver_k(m, Tag_k(m))$ = yes.



We say that the adversary **wins the MAC game** if at the end **outputs** **(m’,t’)** such that

$Ver_k(m',t') = yes$

and

$m' \neq m_1,\ldots,m_w$

The security definition

We say that **(Tag,Ver)** is **secure** if

$\forall$ polynomial-time adversary **A**   **P[A** wins MAC Game**]** is negligible (in **n**)

Security experiment for MAC

- Experiment $\mathrm{Exp}_{\Pi,A}^{\mathrm{MAC}}(n)$:
  1. Choose $k \leftarrow Gen(n)$
  2. *m,t* $\leftarrow A^{Tag()}(n)$
  3. Output 1 if Ver(*m*,*t*) = 1 and *m* was not queried to the Tag() oracle
  4. Output 0 otherwise

(Gen,Tag,Ver) is **a secure (existential unforgeable)** MAC if:

For every **PPT** adversary A:

Pr[$\mathrm{Exp}_{\Pi,A}^{\mathrm{MAC}}(n)$ **= 1] is negligible in n**

MAC example

Let (**Tag,Ver**) be a MAC.

Suppose **Ver**(k,m) is always 5 bits long

Can this MAC be secure?

No, an attacker can simply guess the tag for messages

It depends on the details of the MAC

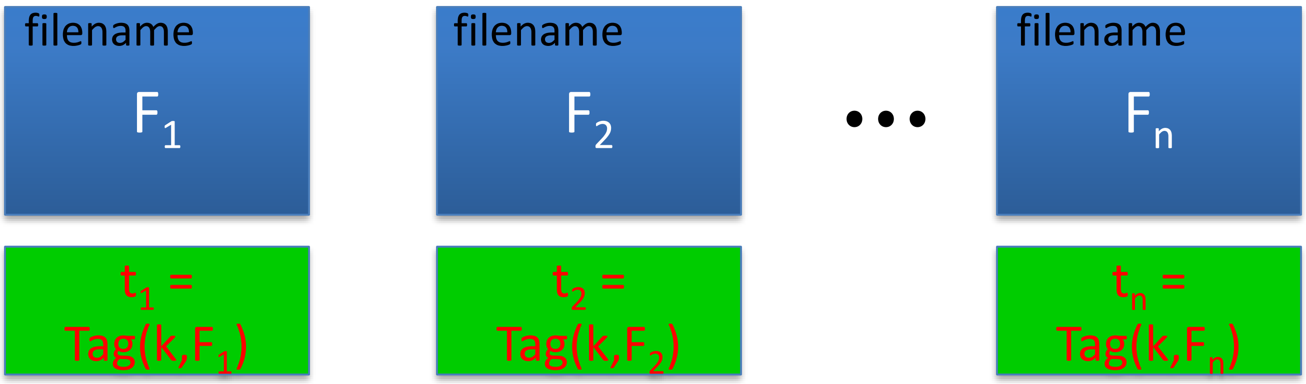
Yes, the attacker cannot generate a valid tag for any message

Encryption does not provide integrity!

- Stream ciphers
  - Enc(k, m) = m $\oplus$ G(k), G a secure PRG
  - Modify 1 bit in c implies one bit modification in the decrypted message
- Block ciphers
  - CTR: Enc is one-time pad with output of PRF function
  - Can modify the ciphertext and decrypt to a different message

Example: protecting system files

Suppose at install time the system computes:



k derived from user's password

Later a virus infects system and modifies system files

User reboots into clean OS and supplies his password

– Then: secure MAC $\Rightarrow$ all modified files will be detected

A simple construction from a block cipher

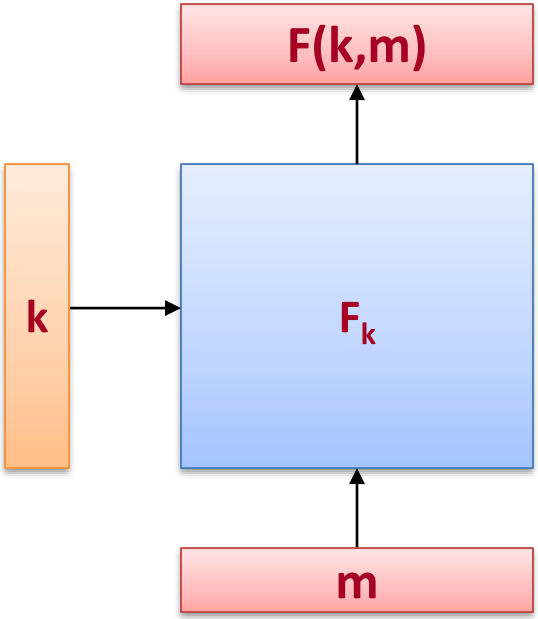
Let

$$F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$$

be a **PRF**.

A **MAC** scheme that works only for messages $m \in \{0,1\}^n$:

- **Tag(k,m) = F(k,m)**
- **Ver(k,m,t): Check t=F(k,m)**



Security

Theorem: If $\mathbf{F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n}$ is a secure PRF, then the PRF-MAC scheme is a secure MAC.

In particular, for every PPT MAC adversary A attacking the MAC there exists a PPT PRF adversary D attacking F s.t.:

$$\mathbf{Pr[}\mathrm{Exp}^{\mathrm{MAC}}_{\Pi,A}(n) \mathbf{= 1]} \leq \mathrm{Adv}^{\mathrm{PRF}}_{\mathrm{F},D} + 1/2^n$$

$$\mathrm{Adv}^{\mathrm{PR}F}_{\mathrm{E},D} = \left|\boldsymbol{Pr}\left[\boldsymbol{D}^{\boldsymbol{F}_{\boldsymbol{k}}(\cdot)}(\boldsymbol{n}) = \mathbf{1}\right] - \boldsymbol{Pr}[\boldsymbol{D}^{\boldsymbol{f}(\cdot)}(\boldsymbol{n})| = \mathbf{1}]\right|$$

How to MAC longer messages?

- AES: a MAC for 16-byte messages.

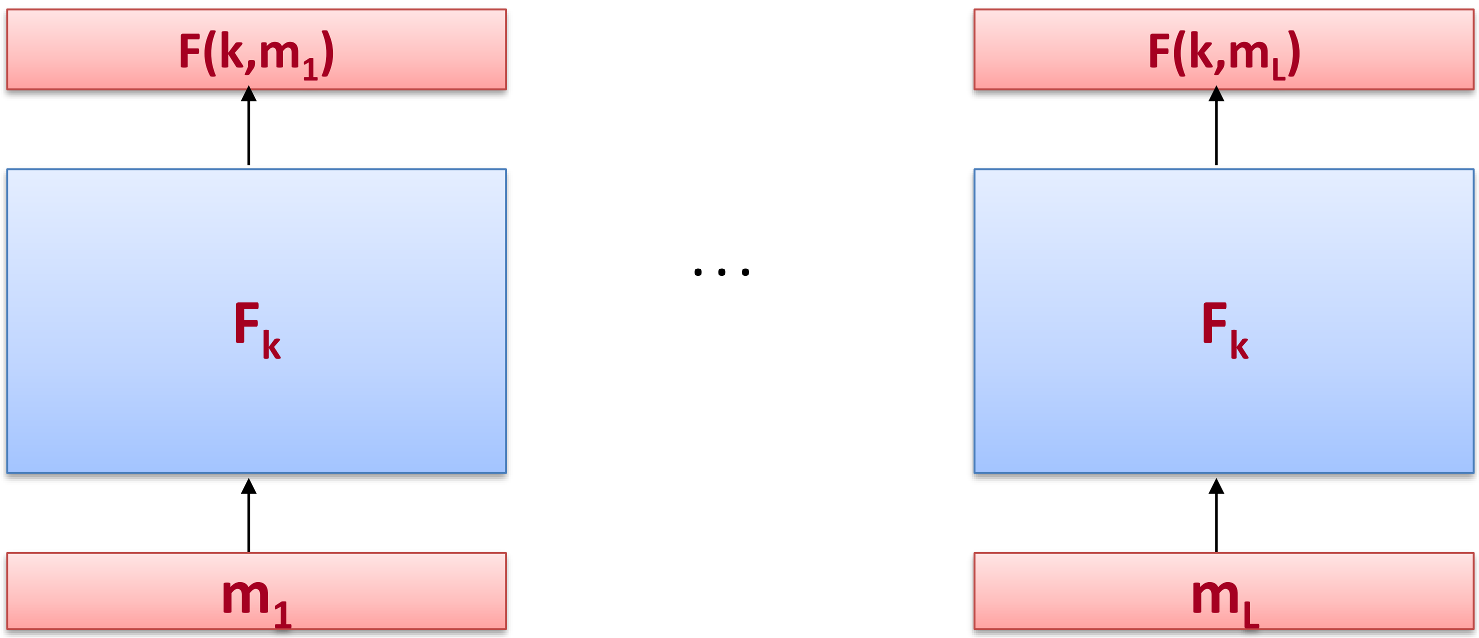
- Main question: how to convert Small-MAC into a Big-MAC ?

- Two main constructions used in practice:
  - **CBC-MAC** (banking – ANSI X9.9, X9.19, FIPS 186-3)
  - **HMAC** (Internet protocols: SSL, IPsec, SSH, …)

- Both convert a small-PRF into a big-PRF.

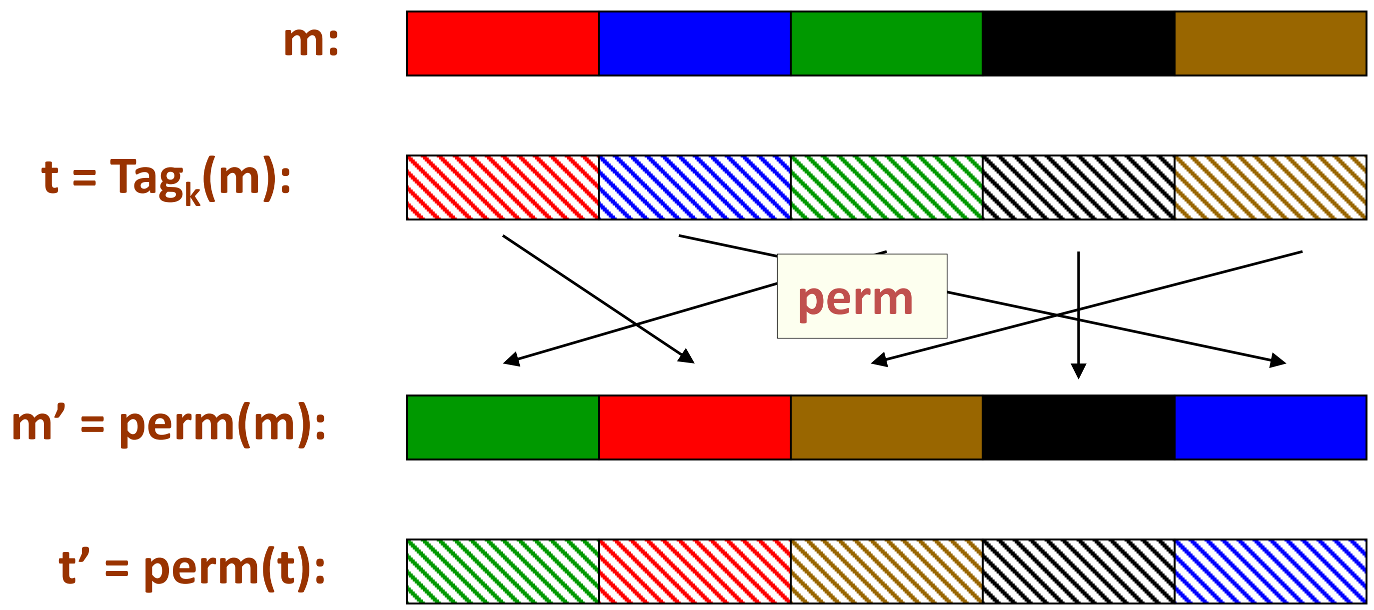
Longer messages: Idea 1

- Divide the message in blocks $m_1,\ldots,m_L$
- Authenticate each block separately



This doesn't work!

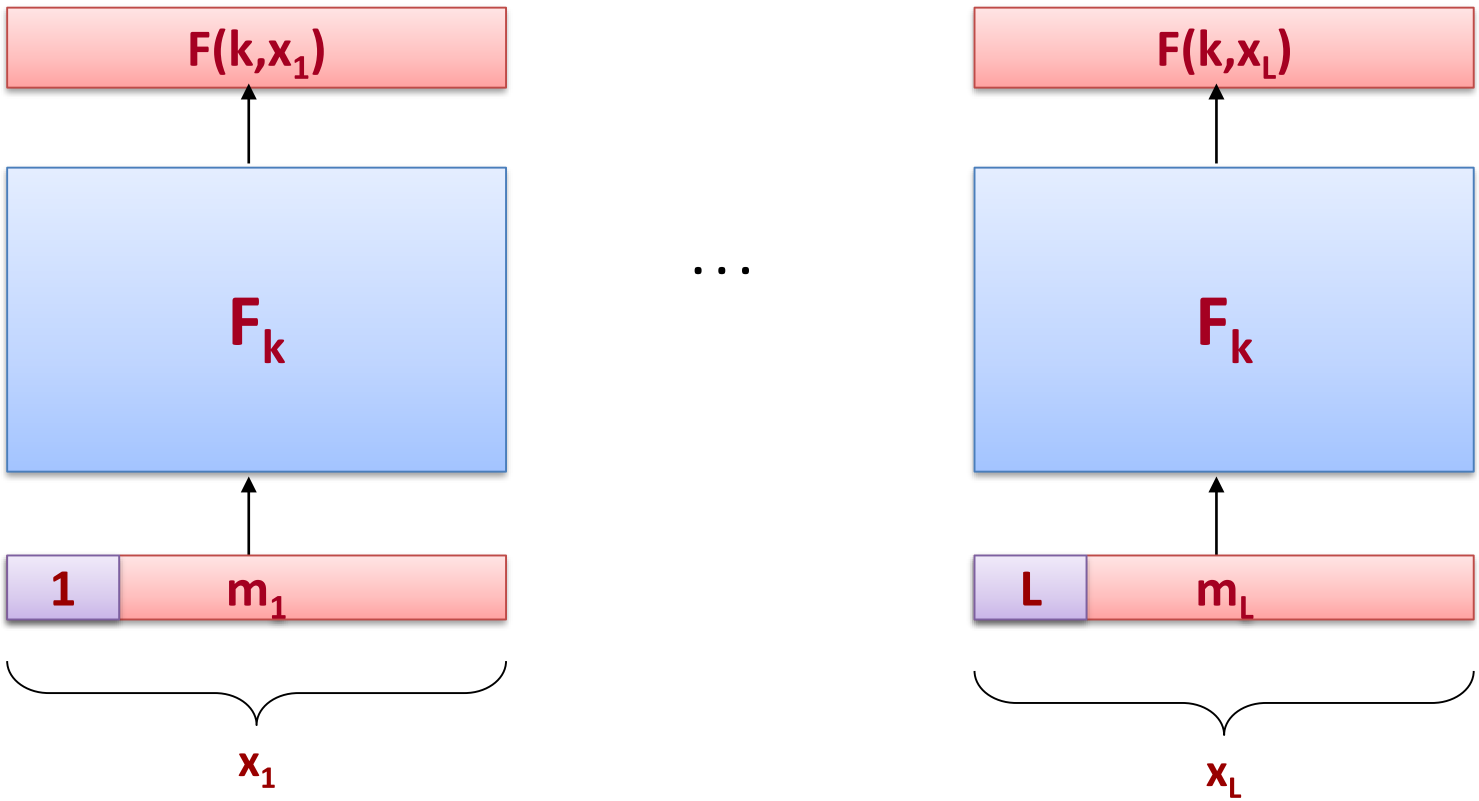
What goes wrong?



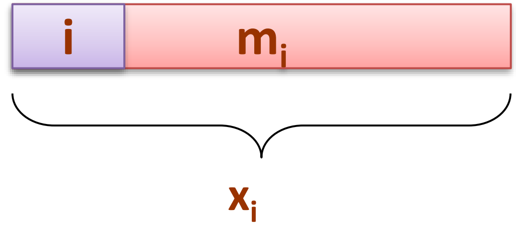
Then **t'** is a valid tag on **m'**.

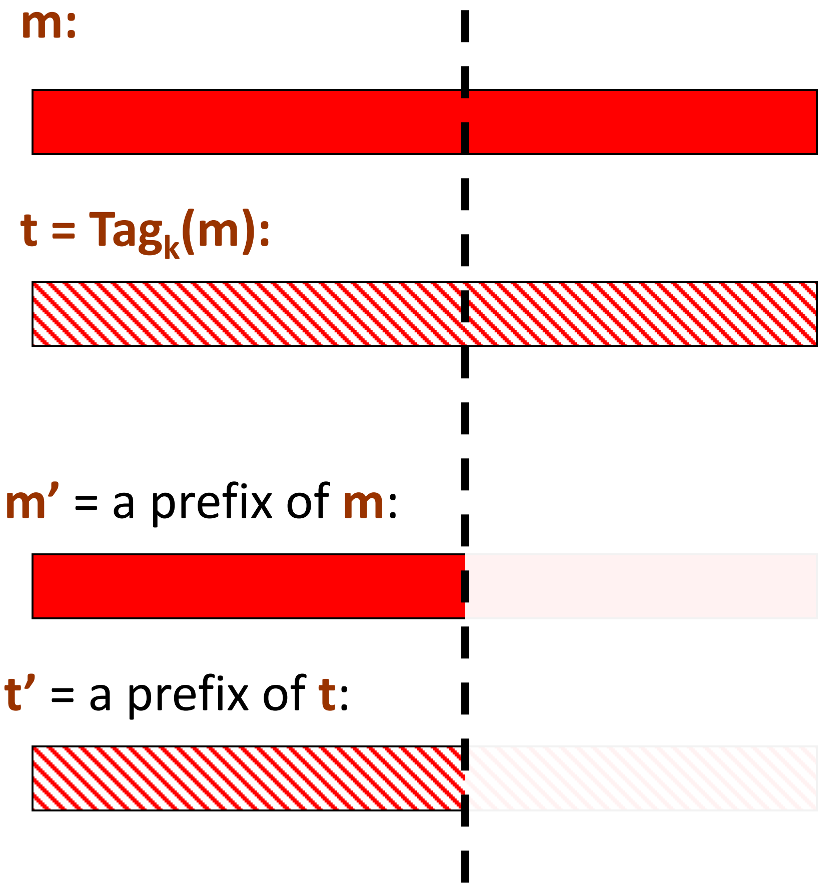
Longer messages: Idea 2

Add a counter to each block.



This doesn't work either!

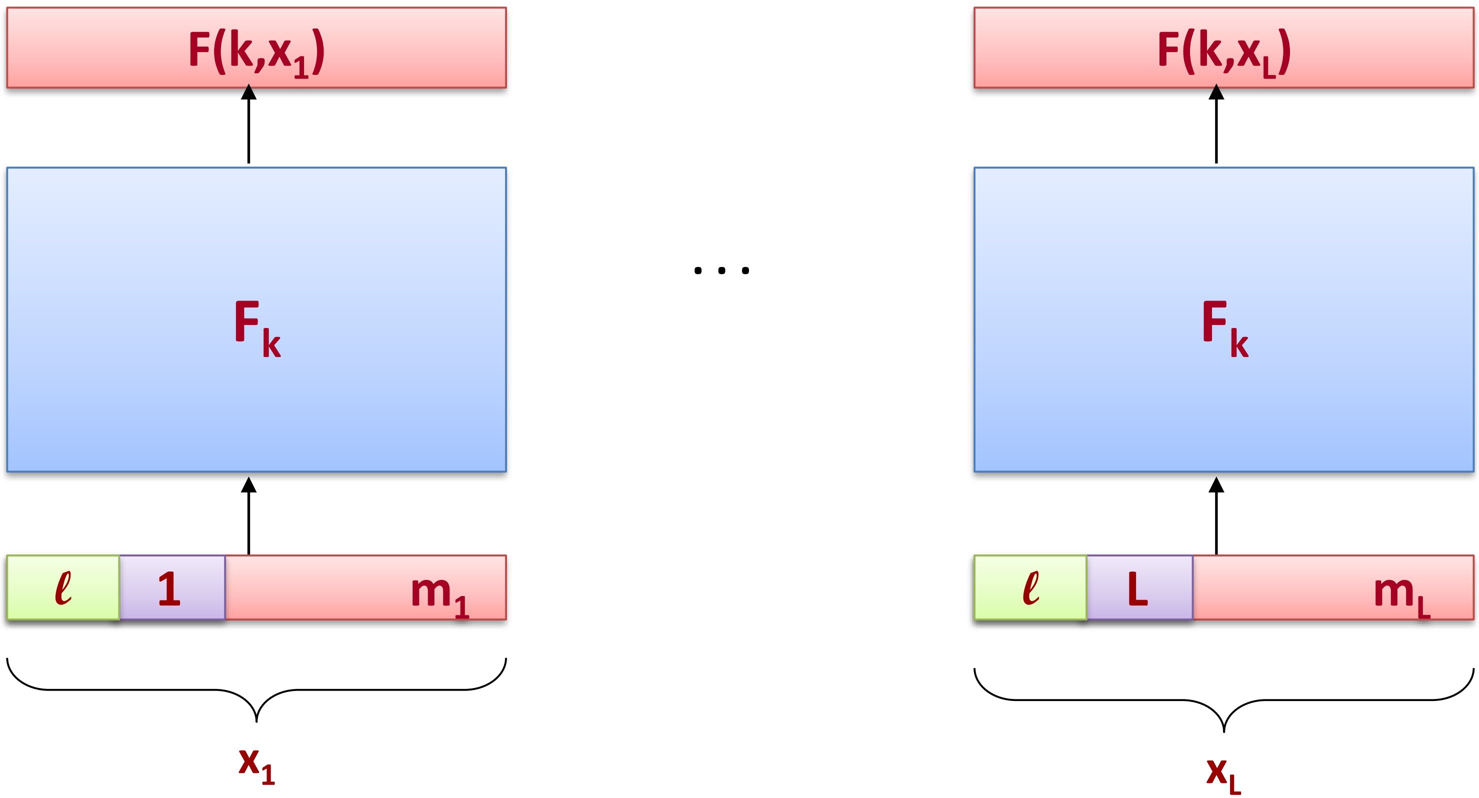




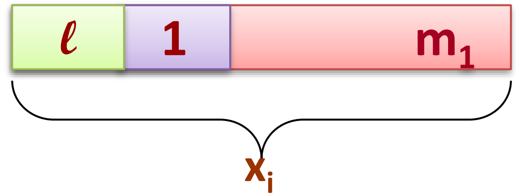
Then **t’** is a valid tag on **m’**.

Longer messages: Idea 3

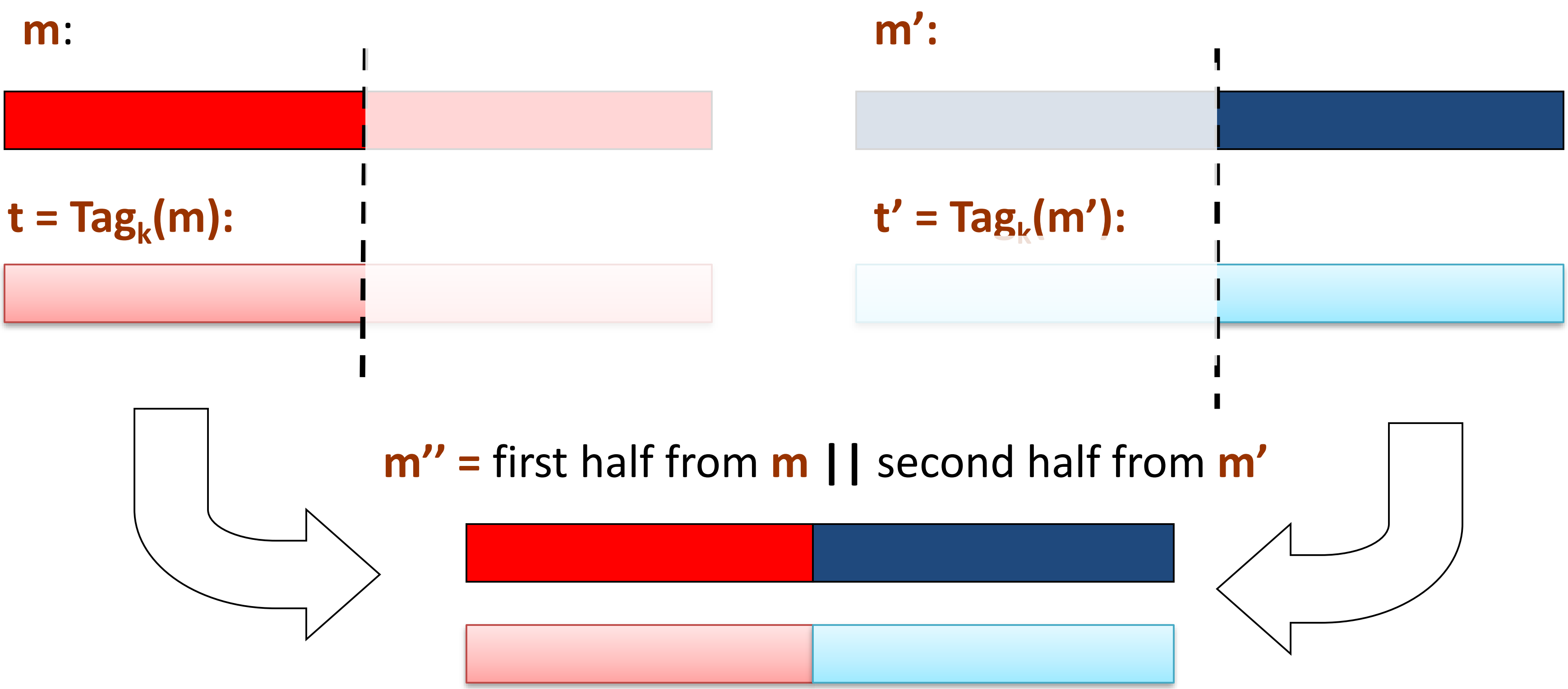
Add $\ell := |m|$ to each block



This doesn't work either!



What goes wrong?

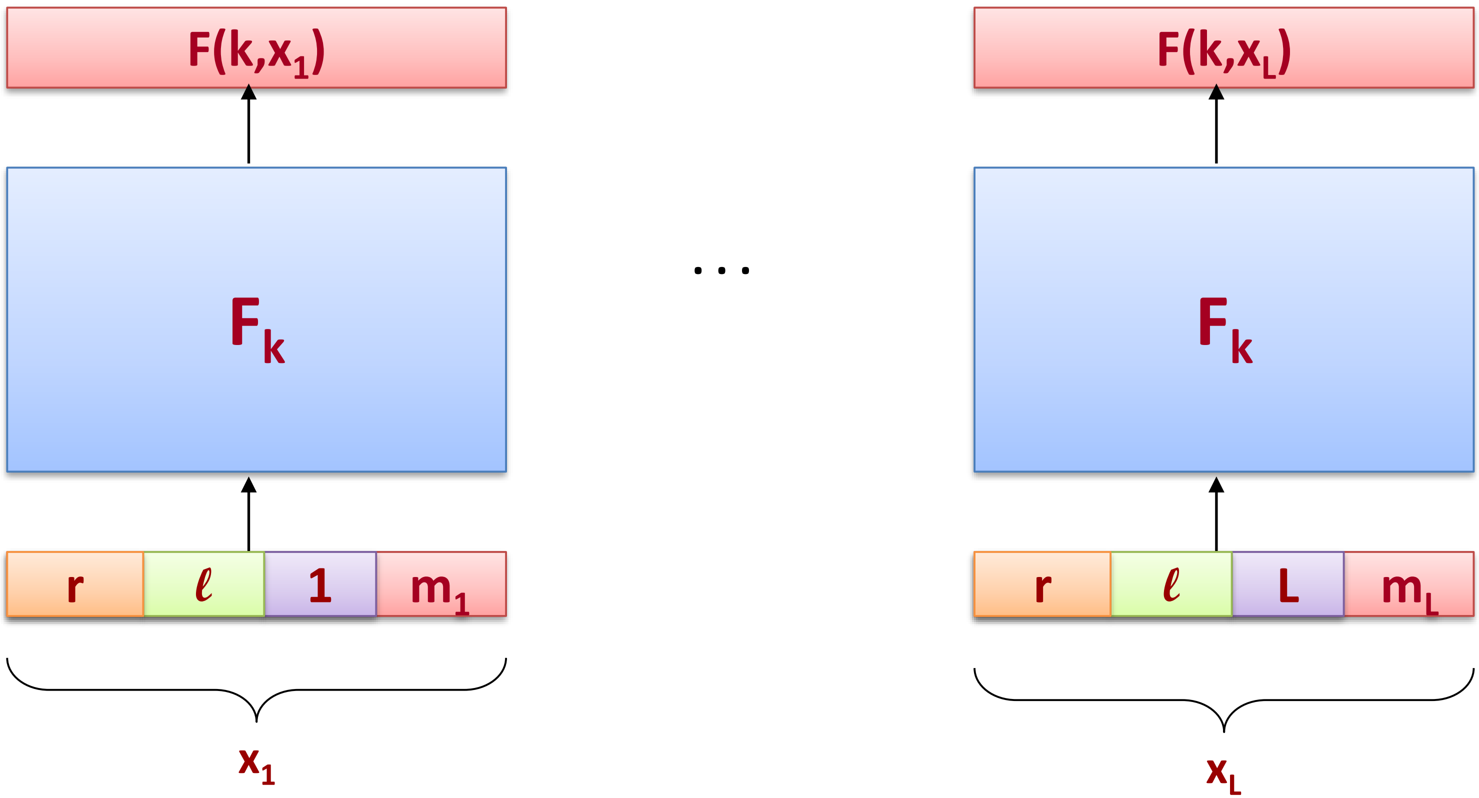


t'' = first half from **t** || second half from **t'**

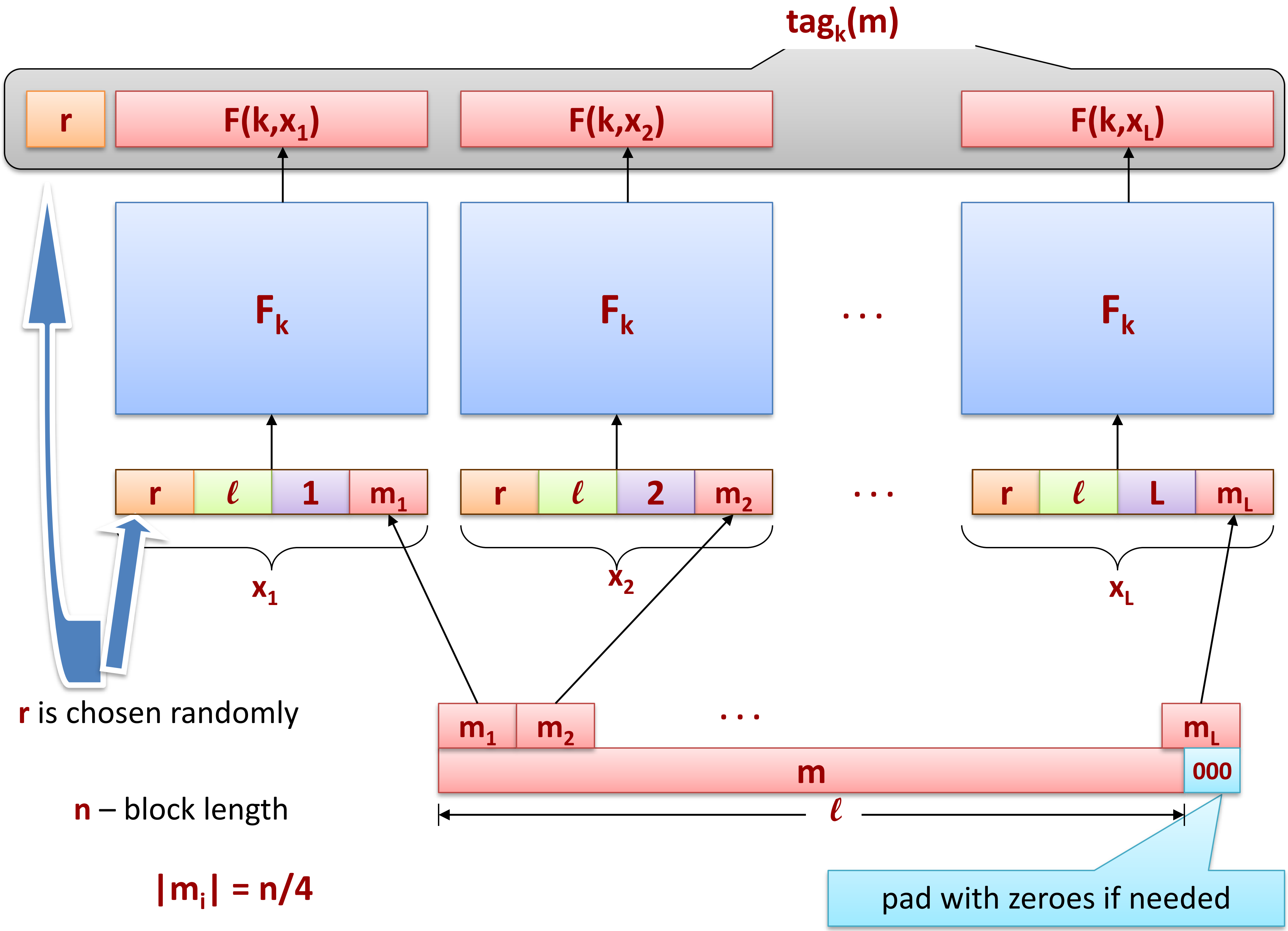
Then **t''** is a valid tag on **m''**.

Longer messages: Idea 4

Add a fresh random value to each block!



This works!



$tag_k(m)$
r
$F(k,x_1)$
$F(k,x_2)$
$F(k,x_L)$
F_k
F_k
F_k
. . .
r
ℓ
1
m_1
r
ℓ
2
m_2
. . .
r
ℓ
L
m_L
x_1
x_2
x_L
m_1
m_2
. . .
m_L
m
000
ℓ
pad with zeroes if needed
r is chosen randomly
n – block length
$|m_i| = n/4$

This construction can be proven secure

Theorem

Assuming that

$F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a **pseudorandom function**

the construction from the previous slide is secure.

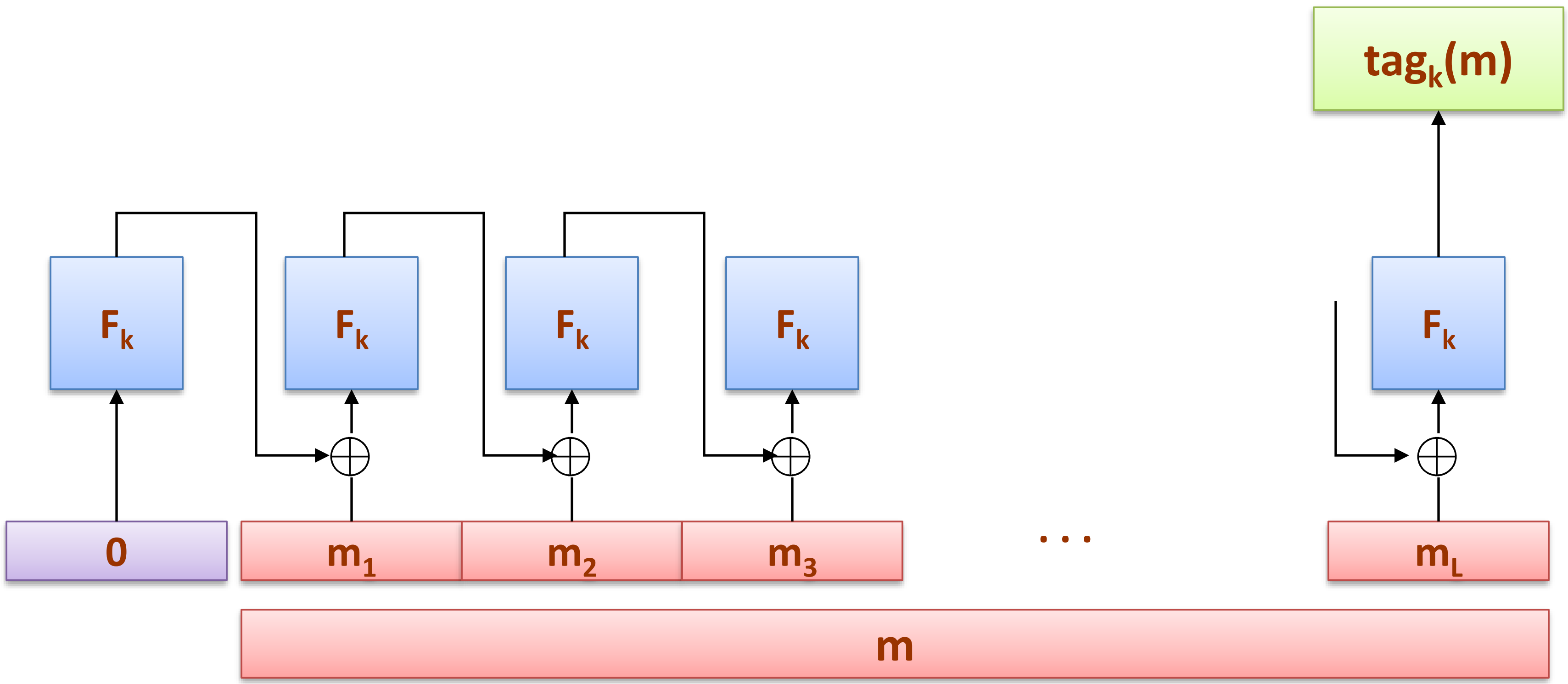
Problem:

The tag is **4 times longer** than the message...

We can do much better!

CBC-MAC

$F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ - a PRF



Theorem

Assuming that $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a **pseudorandom function** and messages of fixed length are tagged, then CBC-MAC construction is secure.

CBC-MAC vs CBC-Enc

- Different security properties
  - CBC-Enc is CPA secure encryption
  - CBC-MAC is secure MAC
- Initialization
  - CBC-Enc uses random IV
  - CBC-MAC uses first block fixed at 0
- Output
  - CBC-Enc outputs all intermediate blocks (to decrypt)
  - CBC-MAC outputs only last block

Key insights

- Integrity vs confidentiality
  - Complementary properties
  - Both are needed in practice
- Message Authentication Codes (MAC)
  - Secret key needed for integrity
  - Security definition
  - Encryption not sufficient for integrity
- Constructions
  - MACs on single block (e.g., 128-bit) can be built from PRFs
  - CBC-MAC for integrity on longer messages

Acknowledgement

Some of the slides and slide contents are taken from
http://www.crypto.edu.pl/Dziembowski/teaching
and fall under the following:

We have also used slides from Prof. Dan Boneh online cryptography course at Stanford University:

http://crypto.stanford.edu/~dabo/courses/OnlineCrypto/