COS433/Math 473: Cryptography

Mark Zhandry

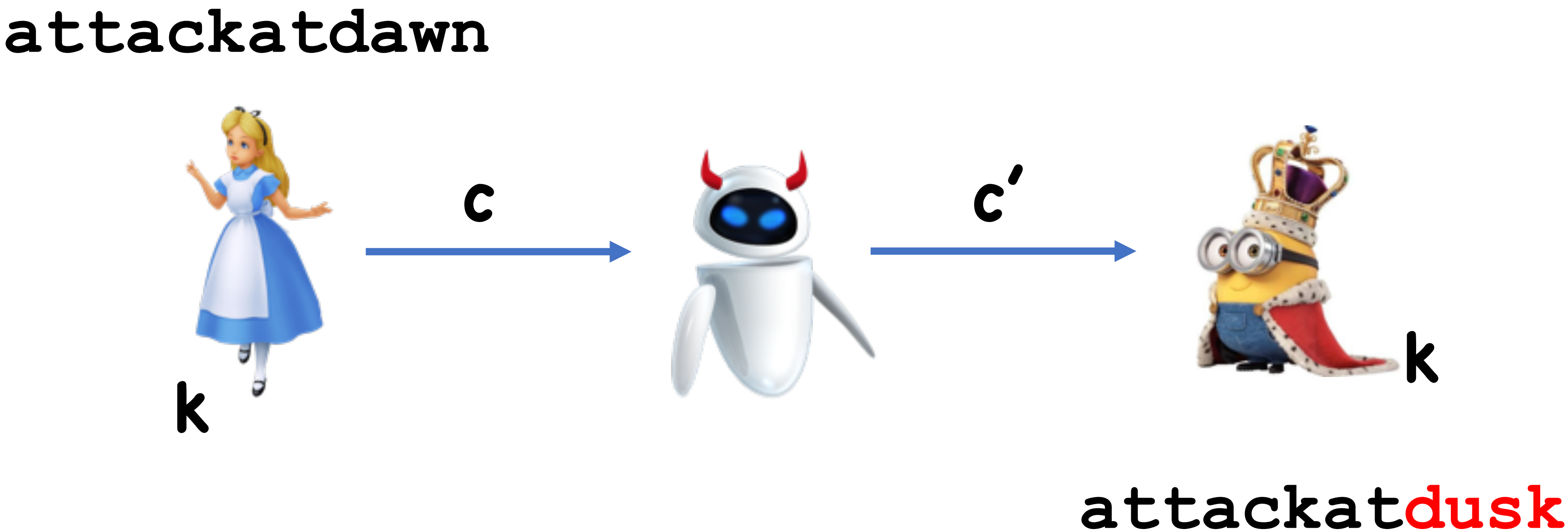
Princeton University

Spring 2020

Previously on COS 433…

Message Integrity

Limitations of CPA security



How?

Message Integrity

We cannot stop adversary from changing the message in route to Bob

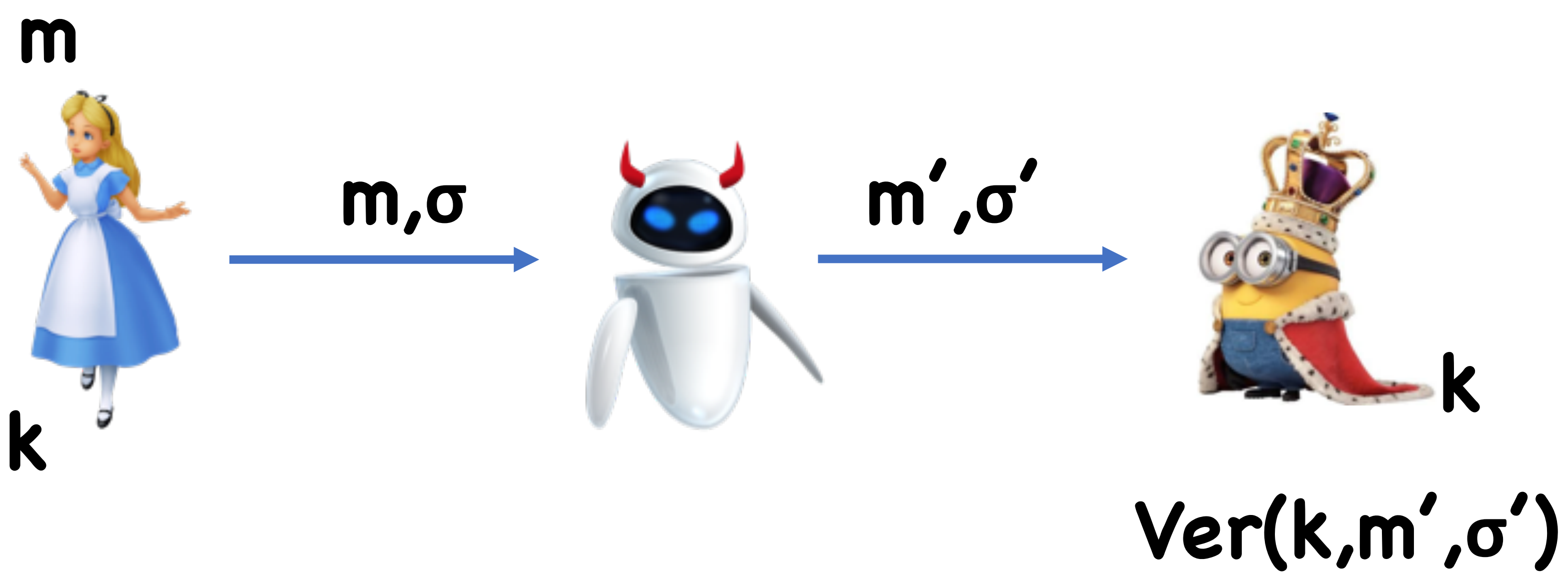
However, we can hope to have Bob perform some check on the message he receives to ensure it was sent by Alice and not modified

- If check fails, Bob rejects the message

For now, we won't care about message secrecy

- We will add it back in later

Message Authentication



Goal: If Eve changed **m**, Bob should reject

Message Authentication Codes

Syntax:

- Key space $\mathbf{K_\lambda}$
- Message space $\mathbf{M_\lambda}$
- Tag space $\mathbf{T_\lambda}$
- **MAC(k,m)** $\rightarrow$ σ
- **Ver(k,m,σ)** $\rightarrow$ **0/1**

Correctness:

- $\forall$**m,k, Ver(k,m, MAC(k,m)) = 1**

1-time Security For MACs

$m \in M_\lambda$

σ

(m^*,σ^*)

$k \leftarrow K_\lambda$

$\sigma \leftarrow MAC(k,m)$

Output 1 iff:

- $m^* \neq m$
- $Ver(k,m^*,\sigma^*) = 1$

1CMA-Adv( , λ) = Pr[ outputs 1]

Definition: (MAC,Ver) is 1-time statistically secure under a chosen message attack (**statistically 1CMA-secure)** if, for all , ∃ negligible **ε** such that:

1CMA-Adv( **, λ) ≤ ε(λ)**

A Simple 1-time MAC

Suppose $\mathbf{H}_\lambda$ is a family of pairwise independent functions from $\mathbf{M}_\lambda$ to $\mathbf{T}_\lambda$

For any $\mathbf{m_0 \neq m_1} \in \mathbf{M}_\lambda$, $\sigma_0, \sigma_1 \in \mathbf{T}_\lambda$

$$\Pr_{h \leftarrow H_\lambda}[\ h(m_0)=\sigma_0 \wedge h(m_1)=\sigma_1] = 1/|T_\lambda|^2$$

$\mathbf{K = H_\lambda}$

$\mathbf{MAC(h, m) = h(m)}$

$\mathbf{Ver(h,m,\sigma) = (h(m) == \sigma)}$

Theorem: If $|T_\lambda|$ is super-polynomial, then **(MAC,Ver)** is 1-time secure

Intuition: after seeing one message/tag pair, adversary learns nothing about tag on any other message

So to have security, just need $|T_\lambda|$ to be large

Ex: $T_\lambda = \{0,1\}^{128}$

Constructing Pairwise Independent Functions

$\mathbf{T_\lambda} = \mathbb{F}$ (finite field of size $\approx 2^\lambda$)

- Example: $\mathbb{Z}_p$ **for some prime p**

Easy case: let $\mathbf{M_\lambda} = \mathbb{F}$

- $\mathbf{H_\lambda} = \{h(x) = a\,x + b : a,b \in \mathbb{F}\}$

Slightly harder case: Embed $\mathbf{M_\lambda} \subseteq \mathbb{F}^n$

- $\mathbf{H_\lambda} = \{h(x) = \langle a,x \rangle + b : a \in \mathbb{F}^n,\ b \in \mathbb{F}\}$

Today

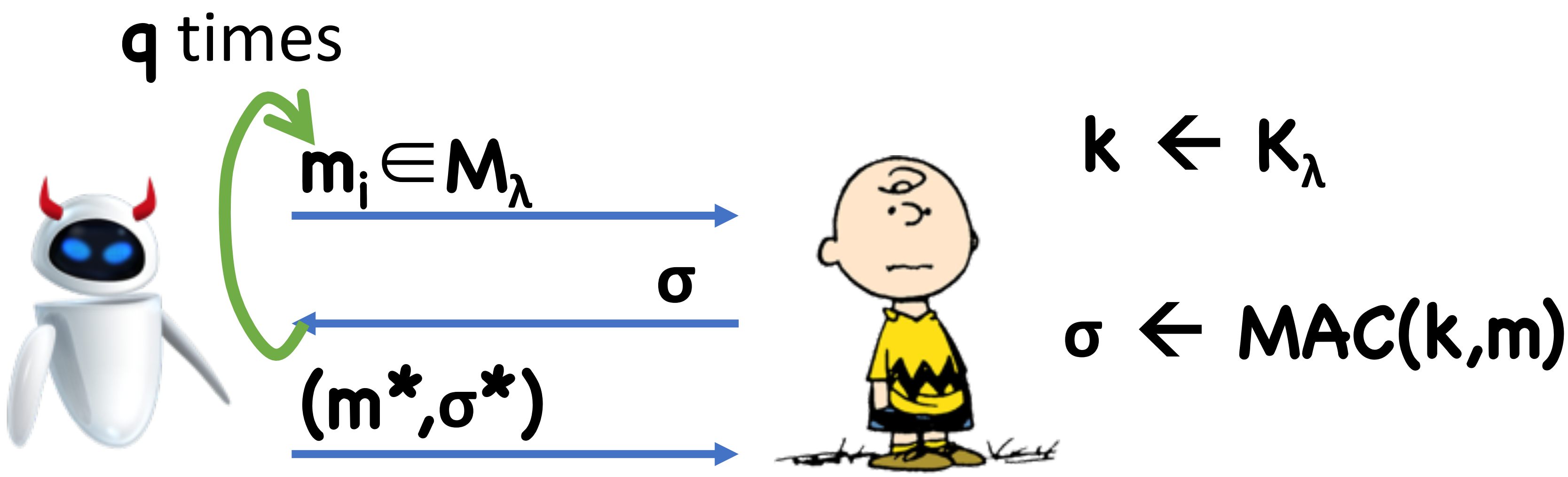
Message integrity, continued

Multiple Use MACs?

Just like with OTP, if use 1-time twice, no security

Why?

$\mathbf{q}$-Time MACs



Output 1 iff:

- $m^* \notin \{m_1,\ldots,m_q\}$
- $Ver(k,m^*,\sigma^*) = 1$

qCMA-Adv( , λ) = Pr[ outputs 1]

Definition: (MAC,Ver) is **q**-time statistically secure under a chosen message attack (**statistically qCMA-secure)** if, for all  making at most **q** queries, $\exists$ negligible **ε** such that:

CMA-Adv( **, λ) ≤ ε(λ)**

Constructing **q**-time MACs

Ideas?

Limitations?

Impossibility of Large **q**

Theorem: Any **qCMA-secure** MAC must have

$$\mathbf{q \leq \log |K_\lambda|}$$

Proof Idea

Idea:

- By making $q \gg \log |K_\lambda|$ queries, you *should* be able to uniquely determine key
- Once key is determined, can forge any message

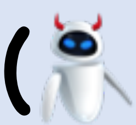
Problem:

- What if certain bits of the key are ignored
- Intuition: ignoring bits of key shouldn't help
- With care, proof can be formalized

Computational Security

Definition: (MAC,Ver) is computationally secure under a chosen message attack (**CMA-secure)** if, for all  running in polynomial time (and making a polynomial number of queries), $\exists$ negligible ε such that

$$\textbf{CMA-Adv}(\text{[adversary]}, \lambda) \leq \varepsilon(\lambda)$$



Constructing MACs

Use a PRF

$F: K_\lambda \times M_\lambda \rightarrow T_\lambda$

$MAC(k,m) = F(k,m)$
$Ver(k,m,\sigma) = (F(k,m) == \sigma)$

Theorem: If $\mathbf{F}$ is a secure PRF and $|\mathbf{T}_\lambda|$ is super-polynomial, then **(MAC,Ver)** is CMA secure

Security Proof

Assume toward contradiction polynomial time 

Hybrids!

Security Proof

Hybrid 0

$m_i \in M_\lambda$

σ_i

(m^*, σ^*)

$k \leftarrow K_\lambda$

$\sigma \leftarrow F(k, m_i)$

Output 1 iff:

- $m^* \notin \{m_1, \ldots\}$
- $F(k, m^*) = \sigma^*$

CMA Experiment

Security Proof

<u>Hybrid 1</u>

$m_i \in M_\lambda$

σ_i

(m^*, σ^*)

$H \leftarrow Funcs(M_\lambda, T_\lambda)$

$\sigma \leftarrow H(m_i)$

Output 1 iff:

- $m^* \notin \{m_1, \ldots\}$
- $H(m^*) = \sigma^*$

Security Proof

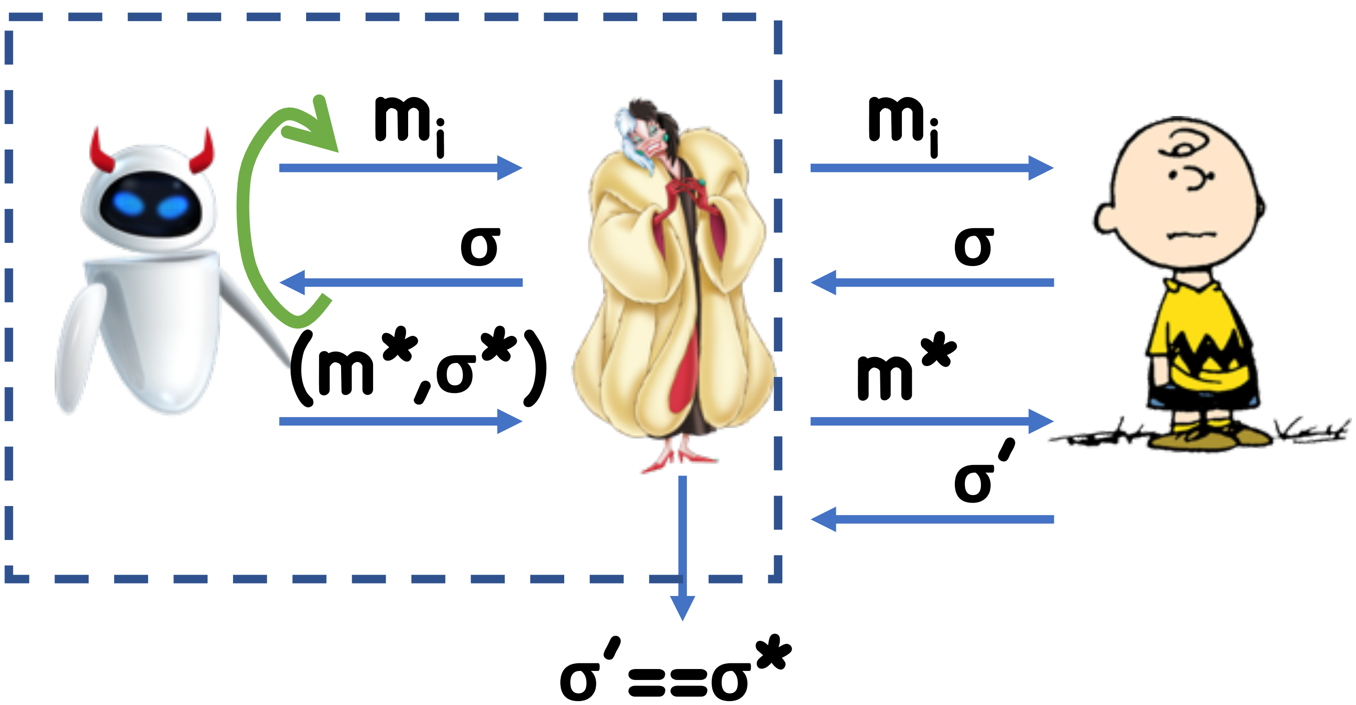
Claim: in Hybrid 1, output 1 with probability $\mathbf{1/|T_\lambda|}$

-  sees values of $\mathbf{H}$ on points $\mathbf{m_i}$
- Value on $\mathbf{m^*}$ independent of  's view
- Therefore, probability $\boldsymbol{\sigma^*=H(m^*) = 1/|T_\lambda|}$

Security Proof

Claim: $|\Pr[1 \leftarrow \text{Hyb1}] - \Pr[1 \leftarrow \text{Hyb2}]| \leq \varepsilon(\lambda)$

Suppose not, construct PRF adversary 



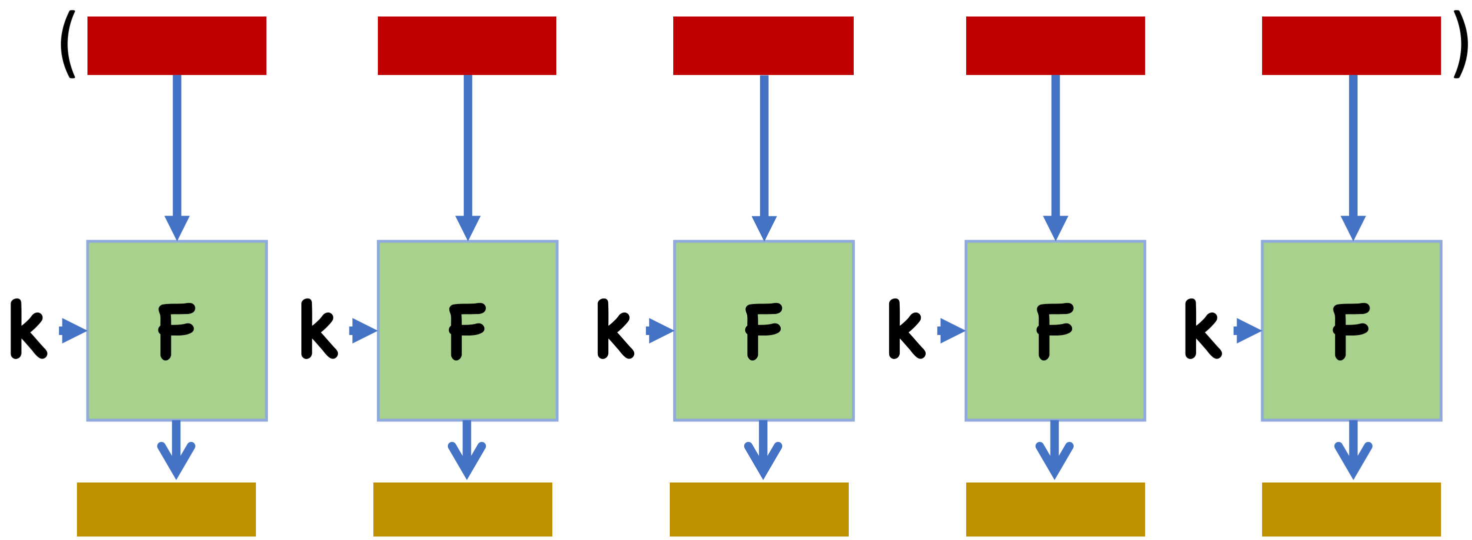
MACs/PRFs for Larger Domains

We saw that block ciphers are good PRFs

However, the input length is generally fixed
- For example, AES maximum block length is 128 bits

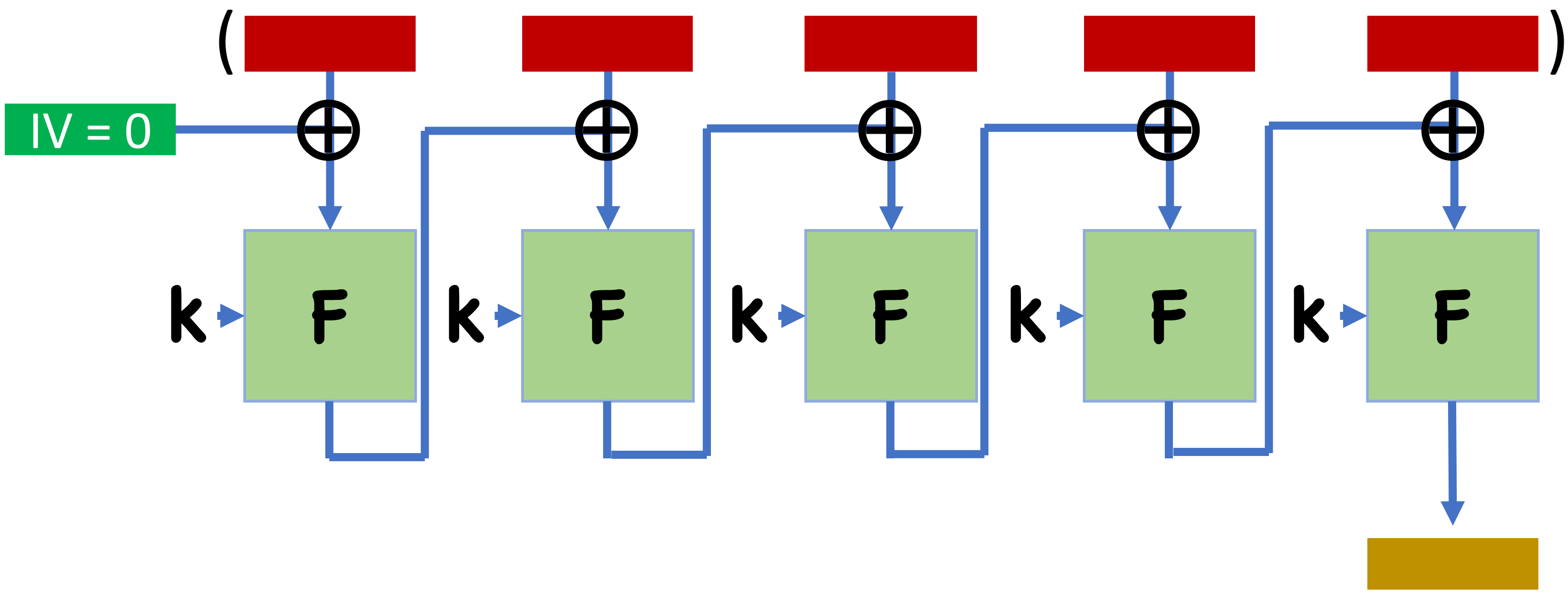
How do we handle larger messages?

Block-wise Authentication?



Why is this insecure?

CBC-MAC



Theorem: CBC-MAC is a secure PRF for **fixed-length** messages

Timing Attacks on MACs

How do you implement check $\mathbf{F(k,m)==\sigma}$?

String comparison often optimized for performance

Compare(A,B):

- **For i = 1,…,A.length**
    - **If A[i] != B[i], abort and return False;**
- **Return True;**

Time depends on number of initial bytes that match

Timing Attacks on MACs

To forge a message **m**:

For each candidate first byte $\boldsymbol{\sigma_0}$:

- Query server on $\boldsymbol{(m, \sigma)}$ where first byte of $\boldsymbol{\sigma}$ is $\boldsymbol{\sigma_0}$
- See how long it takes to reject

First byte is $\boldsymbol{\sigma_0}$ that causes the longest response

- If wrong, server rejects when comparing first byte
- If right, server rejects when comparing second

Timing Attacks on MACs

To forge a message **m**:

Now we have first byte $\boldsymbol{\sigma_0}$

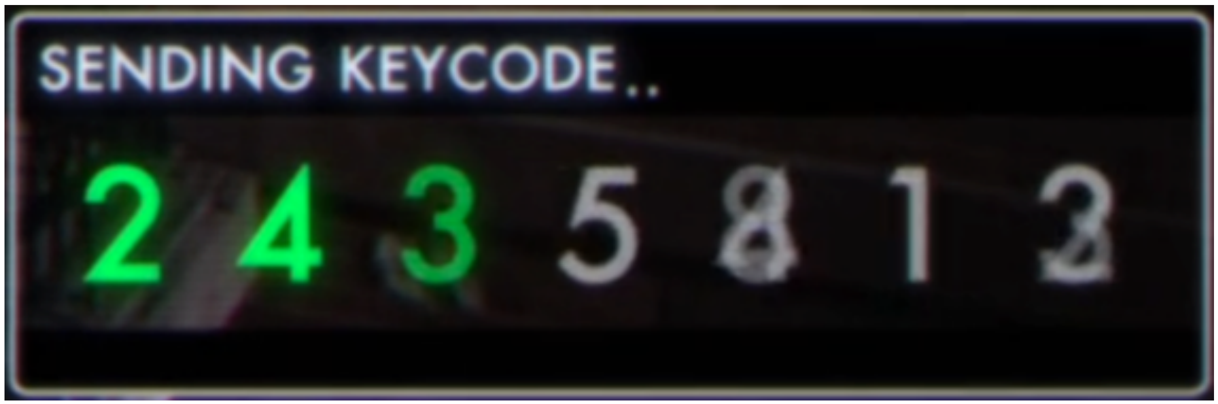
For each candidate second byte $\boldsymbol{\sigma_1}$:

- Query server on $\mathbf{(m,\ \boldsymbol{\sigma})}$ where first two bytes of $\boldsymbol{\sigma}$ are $\boldsymbol{\sigma_0},\boldsymbol{\sigma_1}$
- See how long it takes to reject

Second byte is $\boldsymbol{\sigma_1}$ that causes the longest response

…

Holiwudd Criptoe!



Most likely not what was meant by Hollywood, but conceivable

Thwarting Timing Attacks

Possibility:

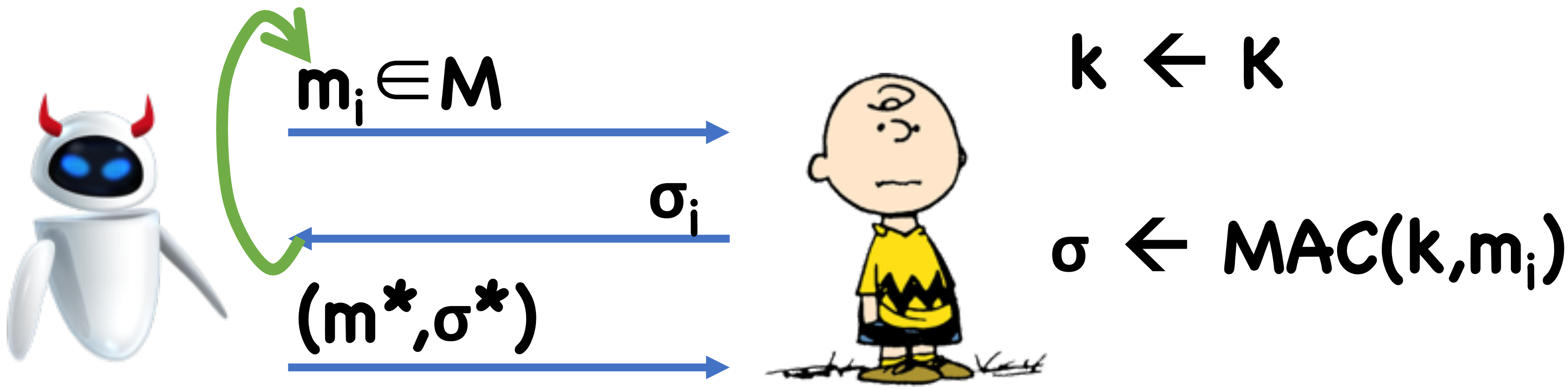
- Use a string comparison that is guaranteed to take constant time
- Unfortunately, this is hard in practice, as optimized compilers could still try to shortcut the comparison

Possibility:

- Choose random block cipher key **k'**
- Compare by testing **F(k',A) == F(k', B)**
- Timing of "**==**" independent of how many bytes **A** and **B** share

Alternate security notions

Strongly Secure MACs



Output 1 iff:

- $(m^*,\sigma^*) \notin \{(m_1,\sigma_1),\ldots\}$
- $Ver(k,m^*,\sigma^*) = 1$

SCMA-Adv() = Pr[ outputs 1]

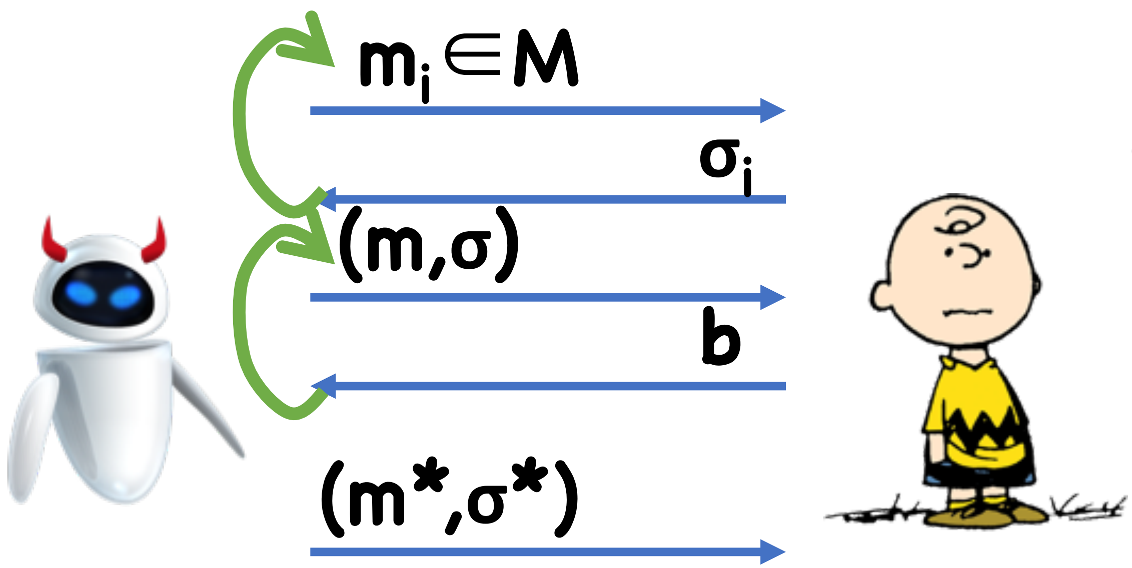
Strongly Secure MACs

Useful when you don't want to allow the adversary to change *any* part of the communication

If there is only a single valid tag for each message (such as in the PRF-based MAC), then (weak) security also implies strong security

In general, though, strong security is stronger than weak security

Adding Verification Queries



$k \leftarrow K$

$\sigma_i \leftarrow MAC(k,m_i)$

$b \leftarrow Ver(k,m,\sigma)$

Output 1 iff:

- $m^* \notin \{m_1,\ldots\}$
- $Ver(k,m^*,\sigma^*) = 1$

CMA'-Adv() = Pr[ outputs 1]

Theorem: (MAC,Ver) is strongly CMA secure if and only if it is strongly CMA' secure

Improving efficiency

Limitations of CBC-MAC

Many block cipher evaluations

Sequential

Carter Wegman MAC

$\mathbf{k' = (k,h)}$ where:

- $\mathbf{k}$ is a PRF key for $\mathbf{F: K \times R \rightarrow Y}$
- $\mathbf{h}$ is sampled from a pairwise independent function family

MAC(k',m):

- Choose a random $\mathbf{r \leftarrow R}$
- Set $\boldsymbol{\sigma} = \mathbf{(r,\ F(k,r) \oplus h(m))}$

Theorem: If $\mathbf{F}$ is secure and $|\mathbf{T}|,|\mathbf{R}|$ are super-polynomial, then the Carter Wegman MAC is strongly CMA secure

Efficiency of CW MAC

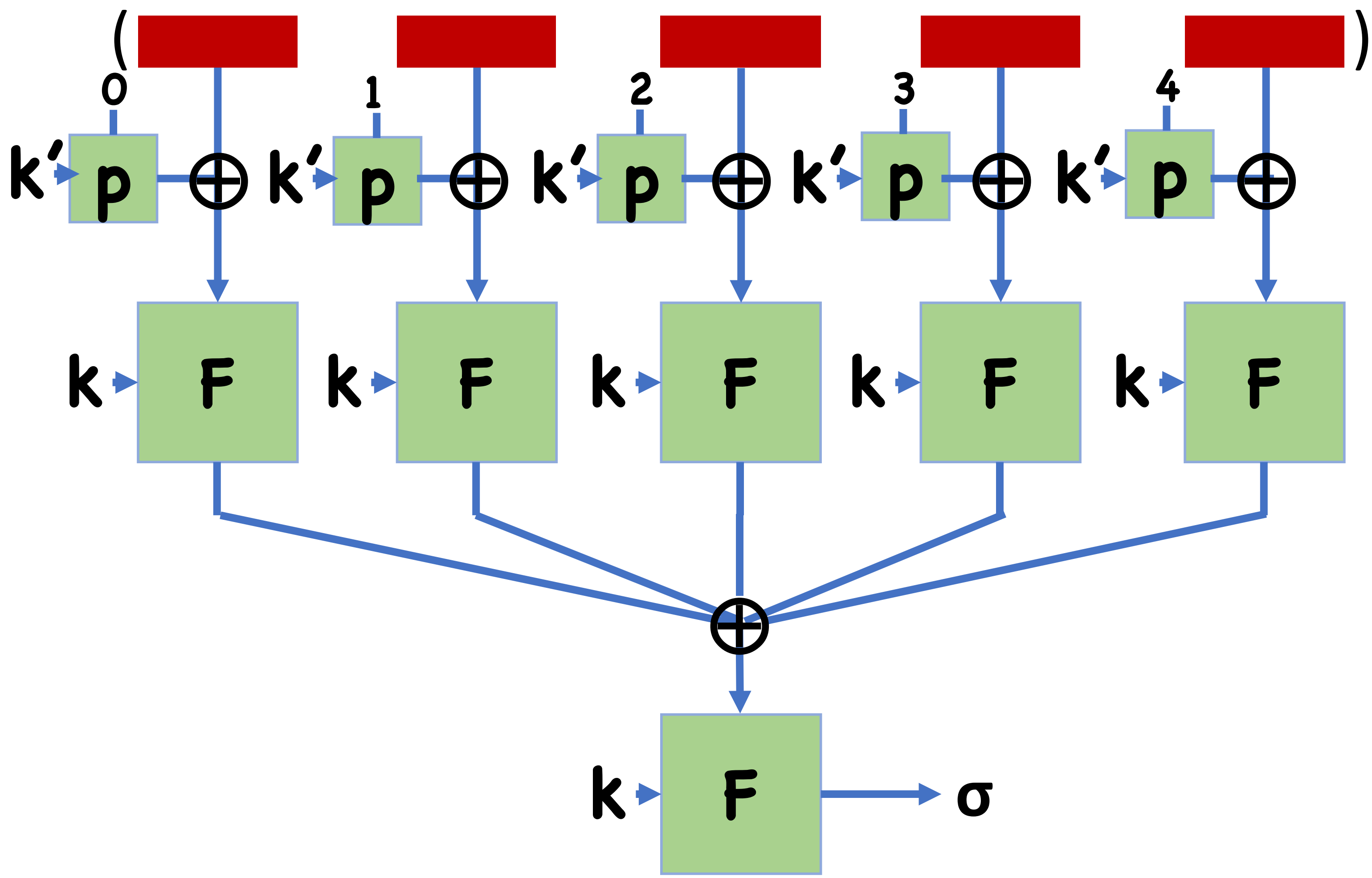
MAC(k',m):

- Choose a random $r \leftarrow R$
- Set $\sigma = (r, F(k,r) \oplus h(m))$

h much more efficient that PRFs

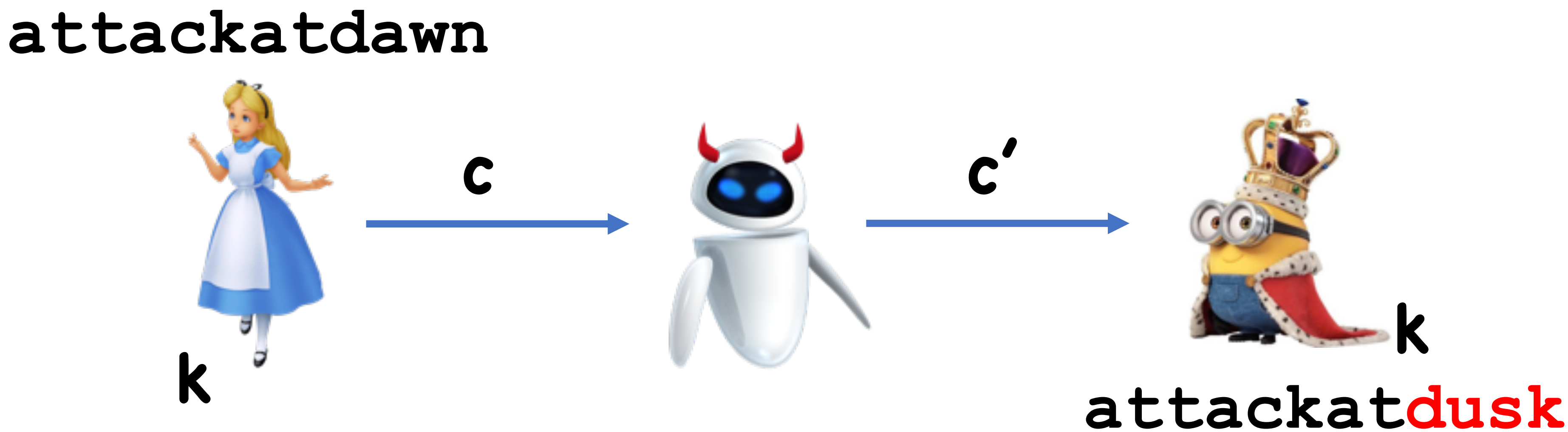
PRF applied only to small nonce **r**
h applied to large message **m**

PMAC: A Parallel MAC



Authenticated Encryption

Authenticated Encryption



Goal: Eve cannot learn nor change plaintext

- Authenticated Encryption will satisfy two security properties

Syntax

Syntax:

- **Enc:** $K \times M \rightarrow C$
- **Dec:** $K \times C \rightarrow M \cup \{\perp\}$

Correctness:

- For all $k \in K$, $m \in M$,

$$\Pr[\mathrm{Dec}(k, \mathrm{Enc}(k,m)) = m] = 1$$

Unforgeability

$m_i \in M$

c_i

c^*

$k \leftarrow K_\lambda$

$c \leftarrow \mathrm{Enc}(k, m_i)$

Output 1 iff:

- $c^* \notin \{c_1, \ldots\}$
- $\mathrm{Dec}(k, c^*) \neq \perp$

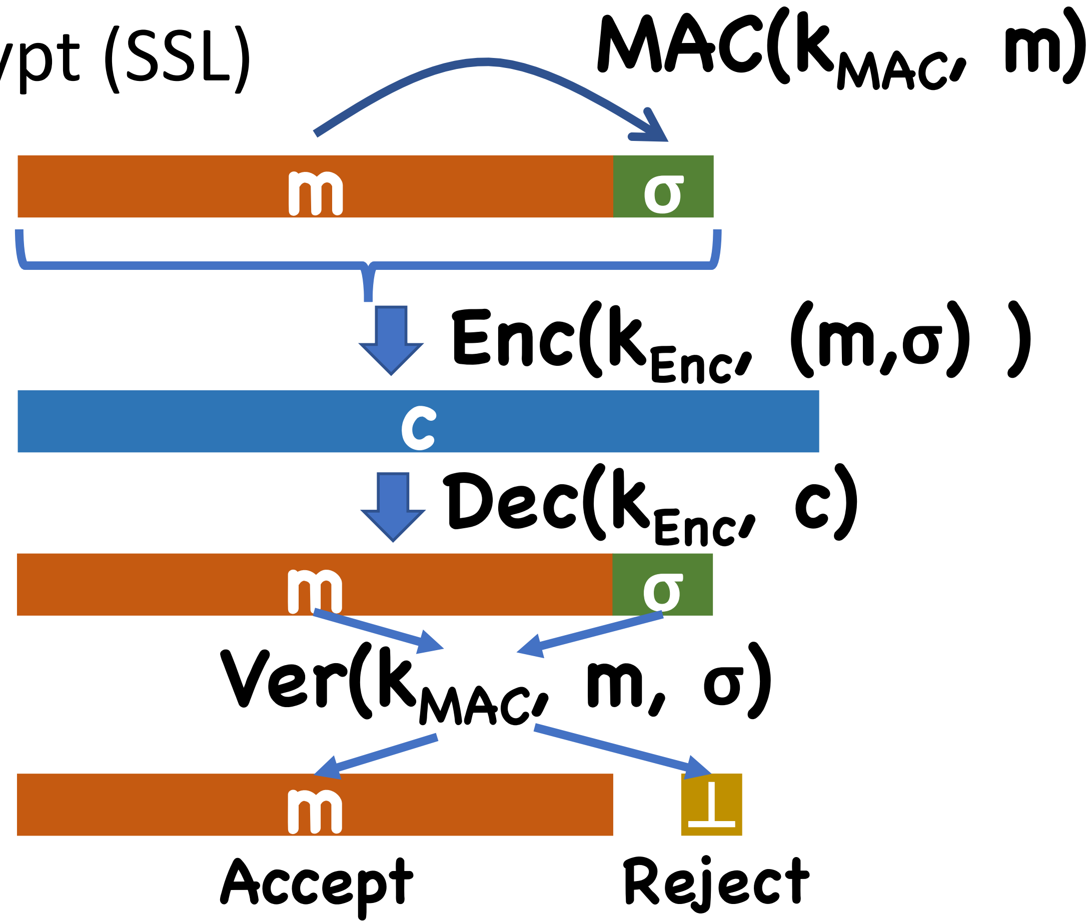
Definition: An encryption scheme **(Enc,Dec)** is an **authenticated encryption scheme** if it is unforgeable and CPA secure

Constructing Authenticated Encryption

Three possible generic constructions:

1. MAC-then-Encrypt (SSL)

$k = (k_{Enc}, k_{MAC})$

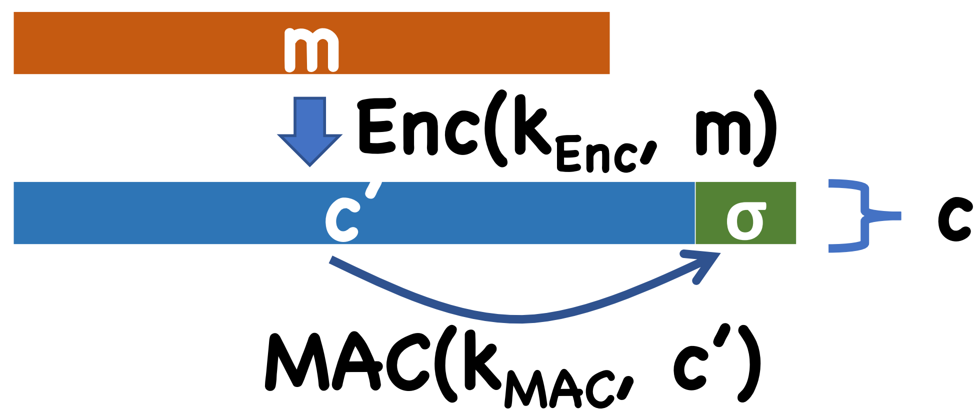


Constructing Authenticated Encryption

Three possible generic constructions:

2. Encrypt-then-MAC (IPsec)

$k = (k_{Enc}, k_{MAC})$

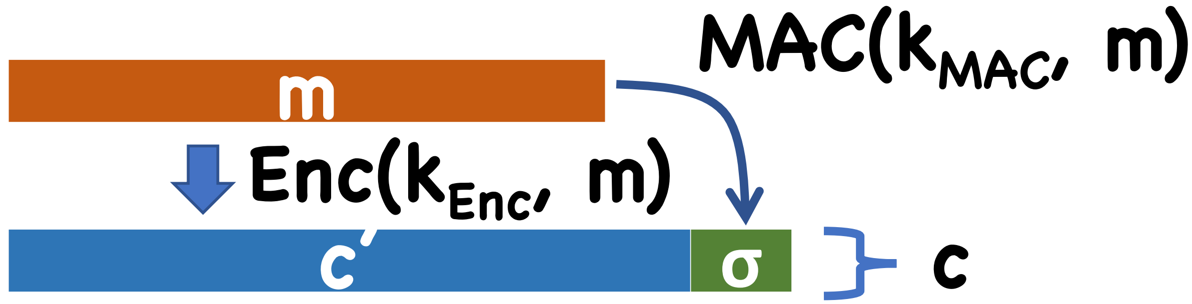


Constructing Authenticated Encryption

Three possible generic constructions:

3. Encrypt-and-MAC (SSH)

$k = (k_{Enc}, k_{MAC})$



Constructing Authenticated Encryption

1. MAC-then-Encrypt

2. Encrypt-then-MAC

3. Encrypt-and-MAC

Which one(s) **always** provides authenticated encryption (assuming strongly secure MAC)?

Constructing Authenticated Encryption

MAC-then-Encrypt?

- Encryption not guaranteed to provide authentication
- May be able to modify ciphertext to create a new ciphertext
- Toy example: **Enc(k,m) = (0,Enc'(k,m))**
  Dec(k, (b,c)) = Dec'(k,c)

Constructing Authenticated Encryption

Encrypt-then-MAC?

- Inner encryption scheme guarantees secrecy, regardless of what MAC does
- (strongly secure) MAC provides integrity, regardless of what encryption scheme does

Theorem: Encrypt-then-MAC is an authenticated encryption scheme for any CPA-secure encryption scheme and *strongly* CMA-secure MAC

✓

Constructing Authenticated Encryption

Encrypt-and-MAC?

- MAC not guaranteed to provide secrecy
- Even though message is encrypted, MAC may reveal info about message
- Toy example: **MAC(k,m) = (m,MAC'(k,m))**

Constructing Authenticated Encryption

1. MAC-then-Encrypt ✘
2. Encrypt-then-MAC ✓
3. Encrypt-and-MAC ✘

Which one(s) **always** provides authenticated encryption (assuming strongly secure MAC)?

Constructing Authenticated Encryption

Just because MAC-then-Encrypt and Encrypt-and-MAC are insecure for *some* MACs/encryption schemes, they may be secure in some settings

Ex: MAC-then-Encrypt with CTR or CBC encryption

- For CTR, any one-time MAC is actually sufficient

Theorem: MAC-then-Encrypt with any one-time MAC and CTR-mode encryption is an authenticated encryption scheme

Chosen Ciphertext Attacks

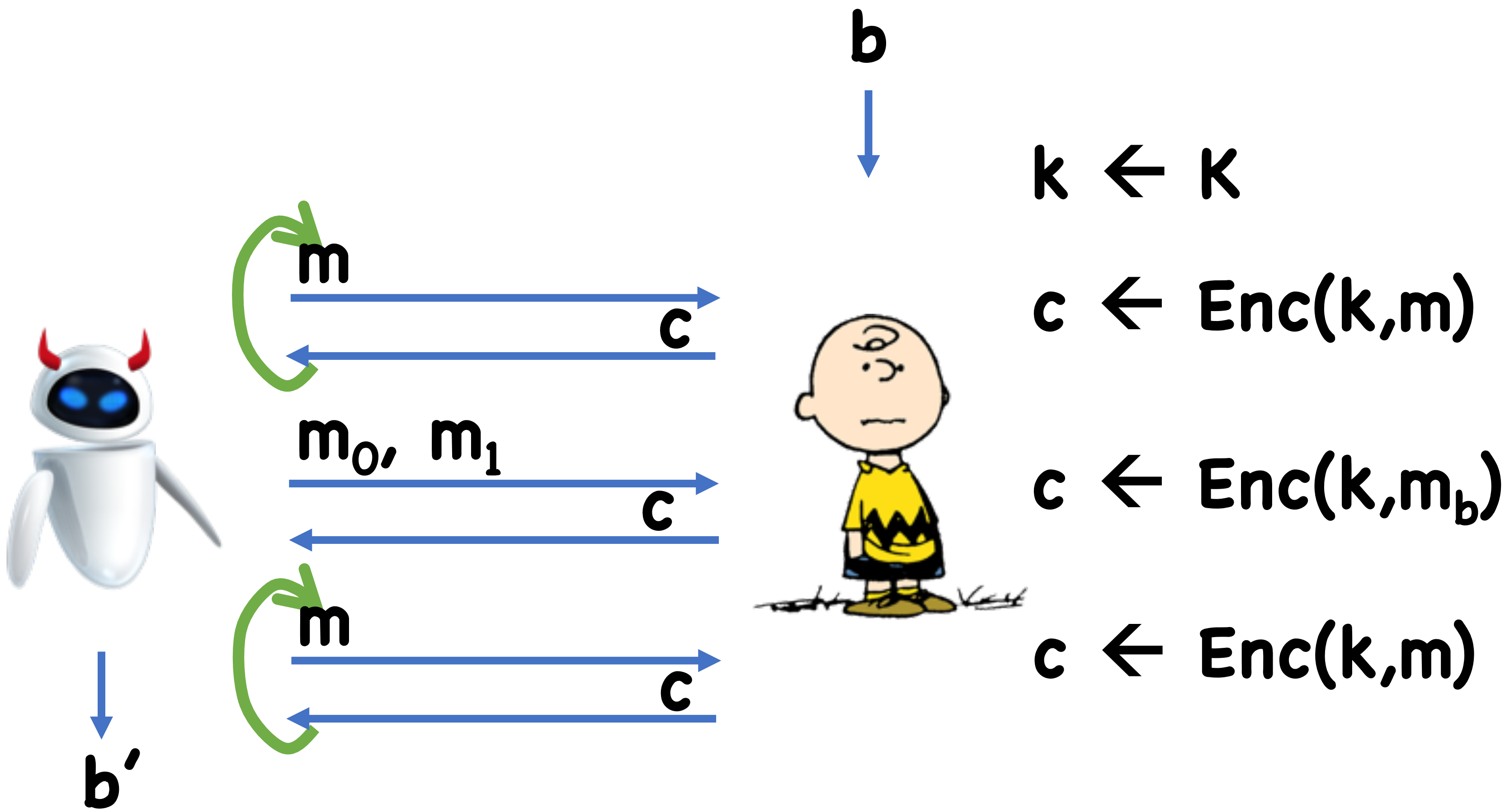
Chosen Ciphertext Attacks

Often, adversary can fool server into decrypting certain ciphertexts

Even if adversary only learns partial information (e.g. whether ciphertext decrypted successfully), can use info to decrypt entire message

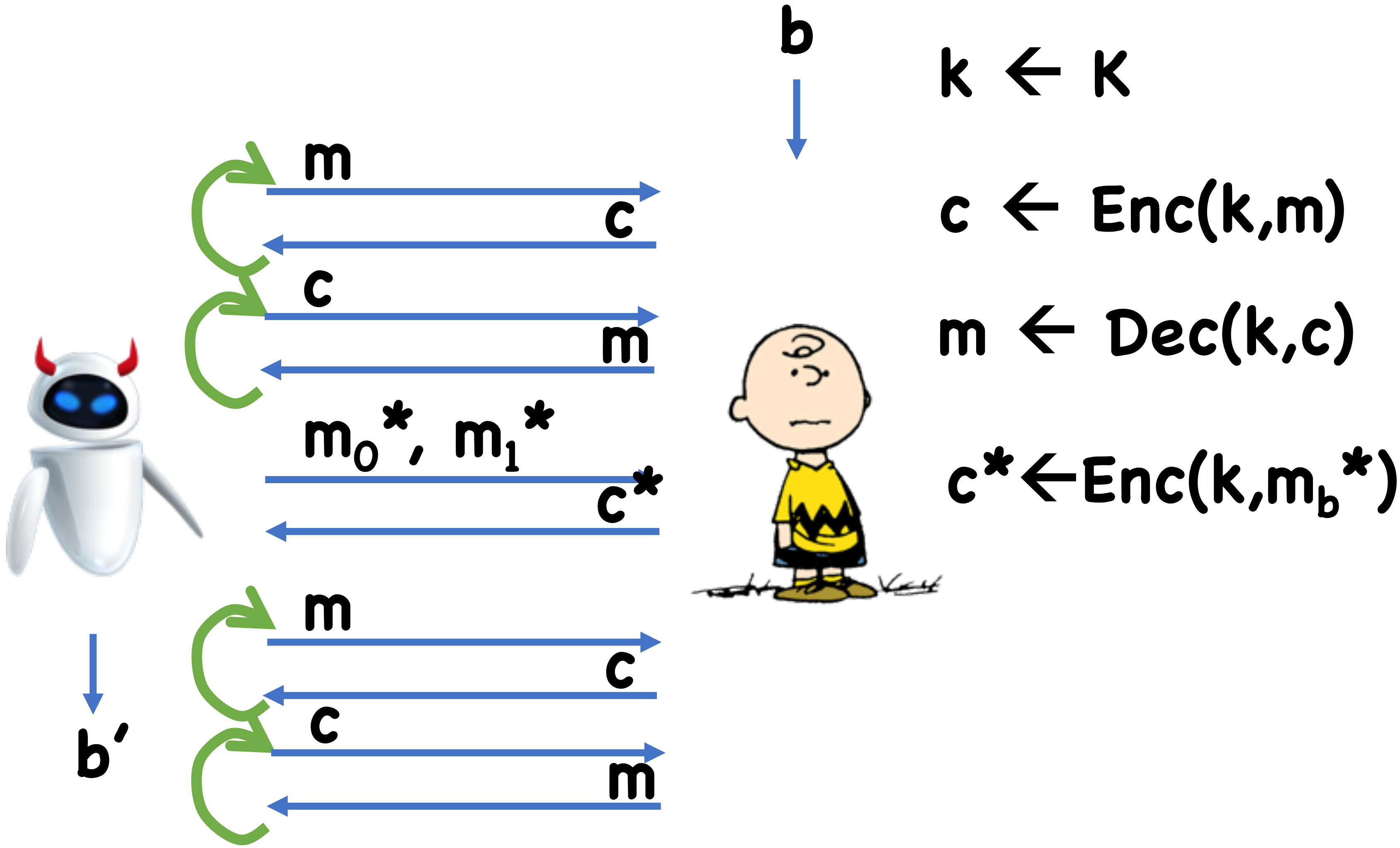
Therefore, want security even if adversary can mount decryption queries

Chosen Plaintext Security

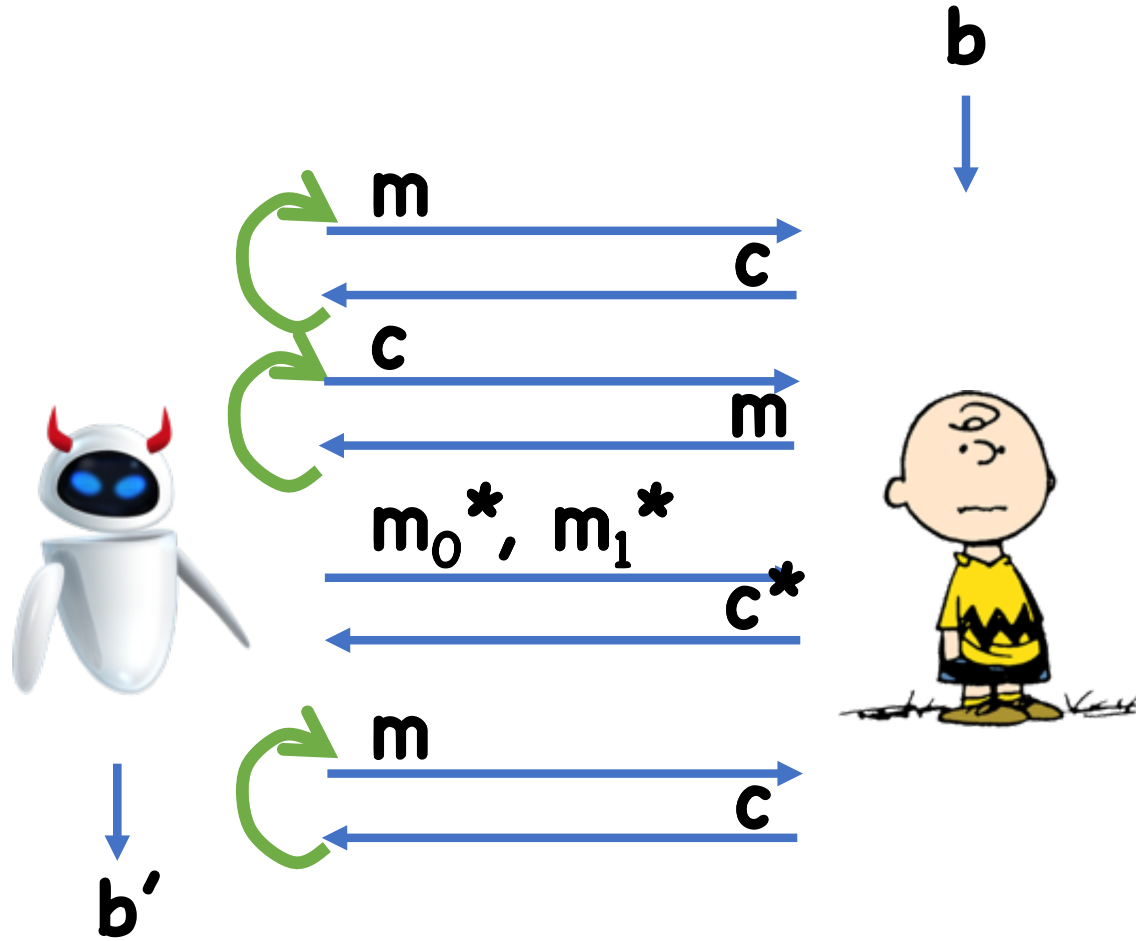


CPA-Exp$_b$( **, λ)**

Chosen Ciphertext Security?



Lunch-time CCA (CCA1)



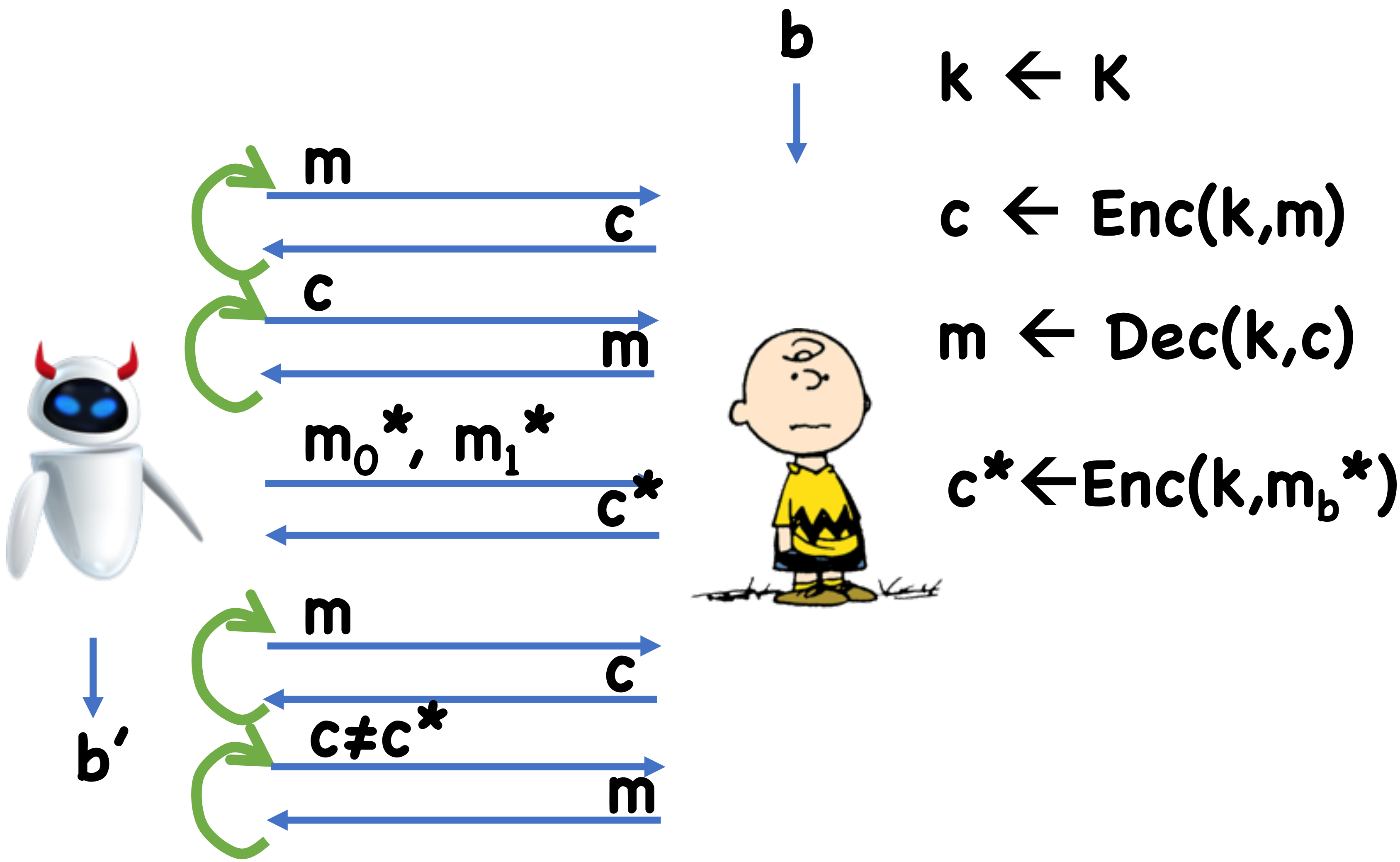
$k \leftarrow K$

$c \leftarrow Enc(k,m)$

$m \leftarrow Dec(k,c)$

$c^* \leftarrow Enc(k,m_b^*)$

Full CCA (CCA2)



Theorem: If **(Enc,Dec)** is an authenticated encryption scheme, then it is also CCA secure

Proof Sketch

For any decryption query, two cases

1. Was the result of a CPA query
- In this case, we know the answer already!

2. Was not the result of an encryption query
- In this case, we have a ciphertext forgery