Cryptography

Lecture 4

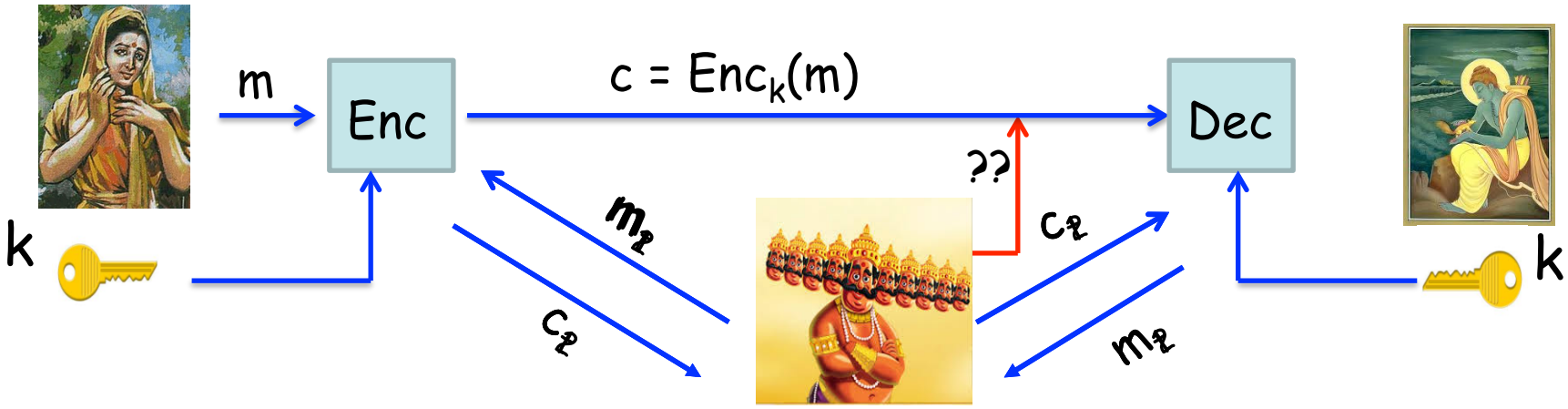
Arpita Patra

Quick Recall and Today's Roadmap

- CPA Security
- PRF-based construction
- Proof of Security
- Extension to CPA-MULT-security
- Modes of Operations (very efficient construction used in practice)

- CCA Security, more stronger than CPA security
- Is it practical? Yes we will break CBC Mode CPA secure scheme under CCA
- Introduction to MAC
- Security Definition
- PRF-based scheme
- Domain Extension for MAC

Chosen-Ciphertext Attacks (CCA) (Single-message Security)



$(m_1, c_1), (m_2, c_2), \ldots, (m_t, c_t)$: $c_i = Enc_k(m_i)$
$(c_1, m_1), (c_2, m_2), \ldots, (c_t, m_t)$: $m_i = Dec_k(c_i)$

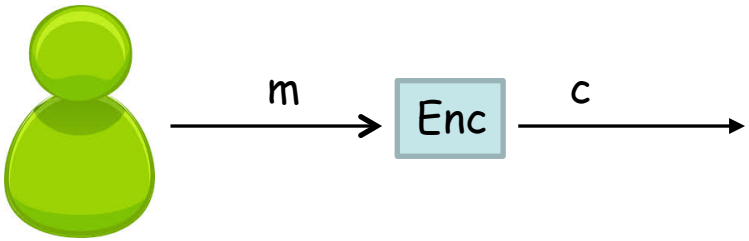
>> CCA is more powerful than CPA (subsumes CPA)

>> Getting Decryption Oracle (DO) Service is much easier than getting Encryption Oracle service

>> A little help from DO can be very very detrimental.

DO Service is Practical

m = transfer
$x from my
account to
account #y



Bank customer

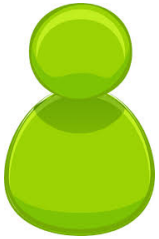
Bank

DO Service is Practical

m = transfer $x from my account to account #y

Dear customer: "did you instructed us to transfer $10000x from your account to account #y ?"

m' = transfer $10000x from my account to account #y



Bank customer

m

Enc

c

I see! So c' is the encryption for the message m' !

Bank

Adv no longer an eavesdropper, he is active and malicious!!

- Similar scenarios:

  >> An attacker sends an arbitrary ciphertext c'(for an unknown message) to army headquarters and waits for the ciphertext to be decrypted and observes the behavior/ movements of the army --- will give an hint what c' corresponds to

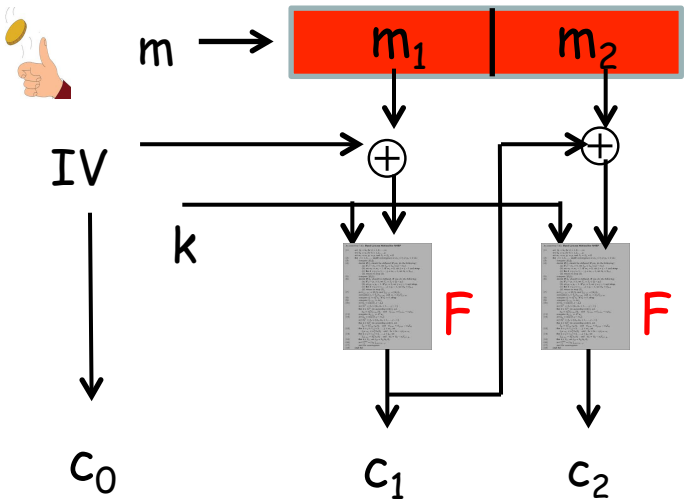
  >> As a part of the protocol, an honest party may give DO service; Think of a simple authentication protocol used in a small company.

DO is Extremely Powerful

- Even the knowledge of whether a modified ciphertext decrypted correctly or not can help an attacker to completely find the underlying plaintext !!
- Padding oracle attack --- can be easily launched on several practically deployed ciphers
- CBC-mode of encryption and de … block length L in bytes

But what if $|m| \neq l\,L$?

Encryption



Decryption

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

$m_1 = F^{-1}(c_1) \oplus c_0$

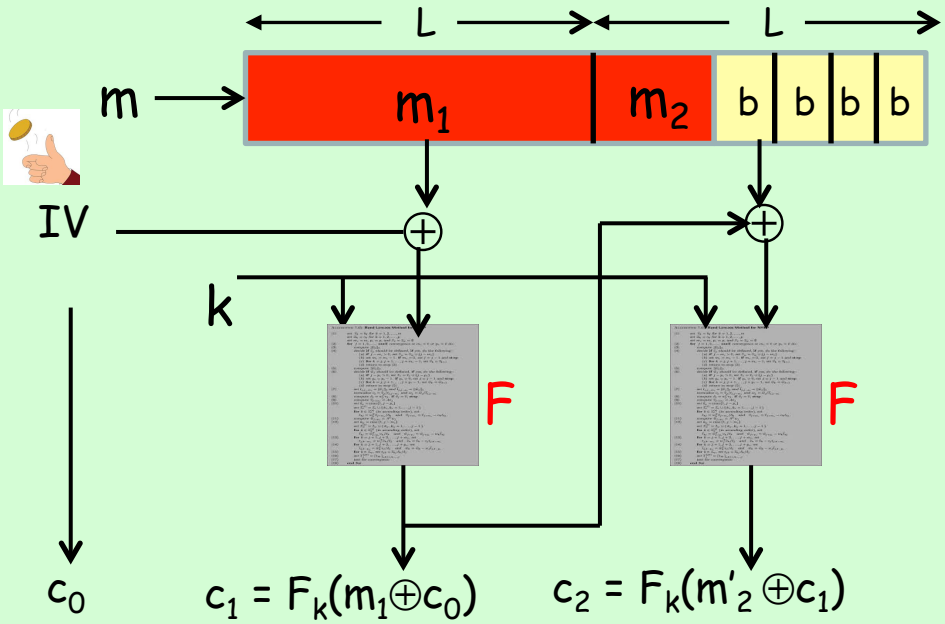
- PKCS#5 padding --- a popular padding

>> Let b be the number of bytes need to be appended in the last block of m to make its length L bytes --- $1 \leq b \leq L$

>> Append b bytes to the last block of m, each of them representing the integer value b

CBC Mode with PKCS#5 Padding

Encryption



Decryption

Decrypt as per usual CBC-mode decryption and obtain $m_1 \,||\, m'_2$

$$m'_2 = F^{-1}(c_2) \oplus c_1$$

$$m_1 = F^{-1}(c_1) \oplus c_0$$

Read the final byte value b

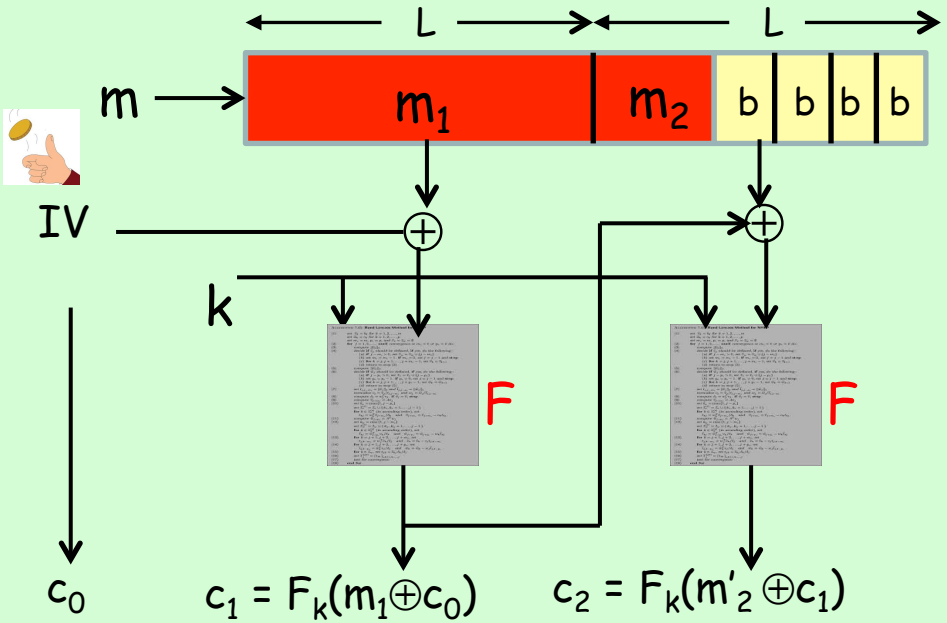
If the last b bytes of m'_2 all have value b then strip-off the pad and output m

Else output bad padding (request for re-transmission)

- An attacker can modify the ciphertexts and learn b (|m| leaked) and m.
- Hint: What will happen to the decryption of m_2 if the i^{th} byte of c_1 is modified by Δ ?
  - m'_2 on decryption will be modified by Δ at i^{th} byte !!

Padding Oracle Attack on CBC Mode

Encryption



Decryption

Decrypt as per usual CBC-mode decryption and obtain $m_1 \,||\, m'_2$

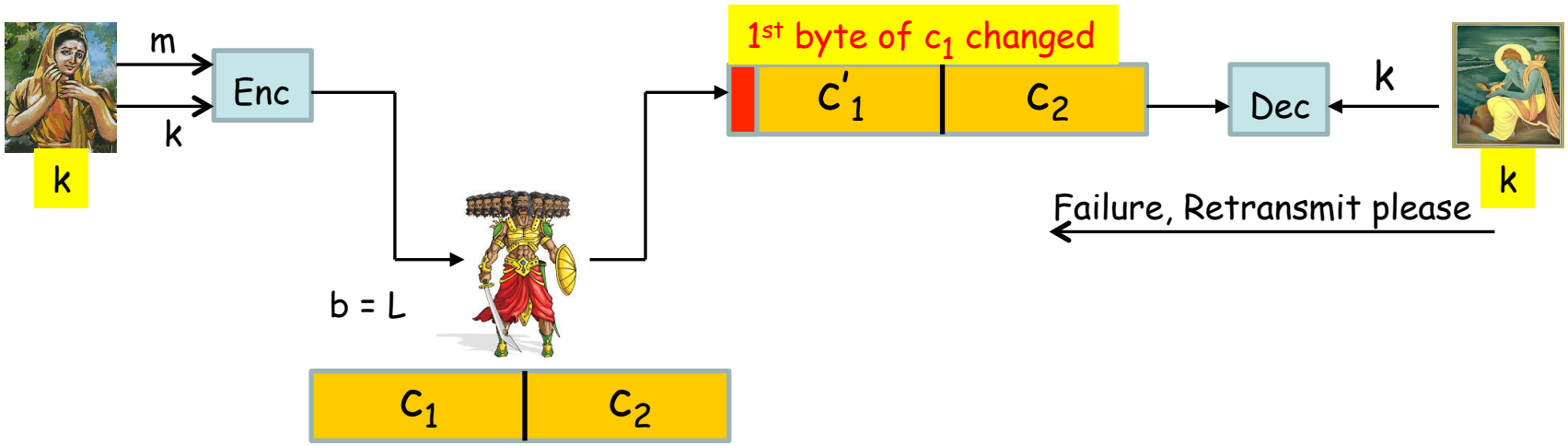
$$m'_2 = F^{-1}(c_2) \oplus c_1$$

$$m_1 = F^{-1}(c_1) \oplus c_0$$

Read the final byte value b

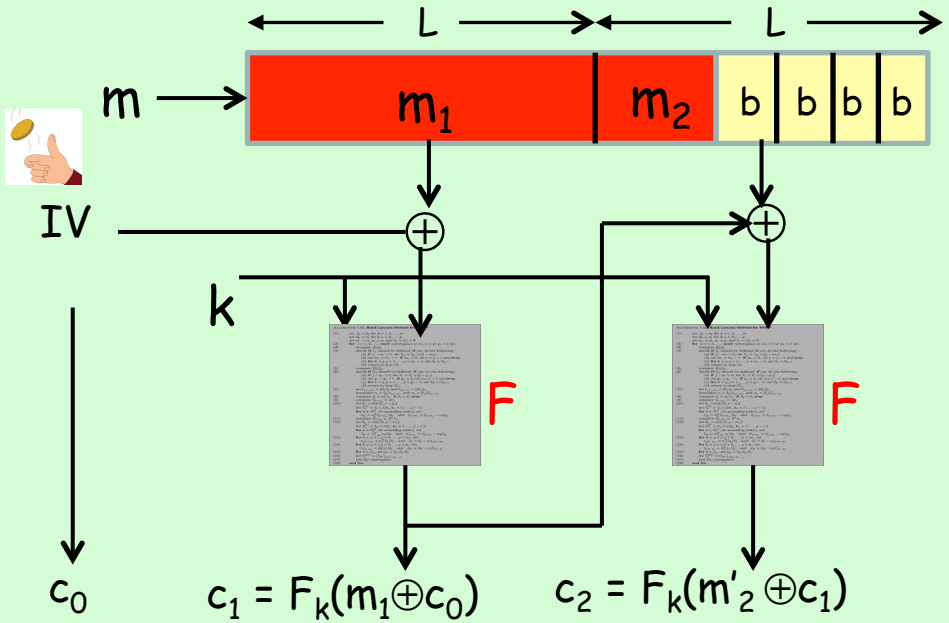
If the last b bytes of m'_2 all have value b then strip-off the pad and output m

Else output bad padding (request for re-transmission)



Padding Oracle Attack on CBC Mode

Encryption



Decryption

Decrypt as per usual CBC-mode decryption and obtain $m_1 \,||\, m'_2$

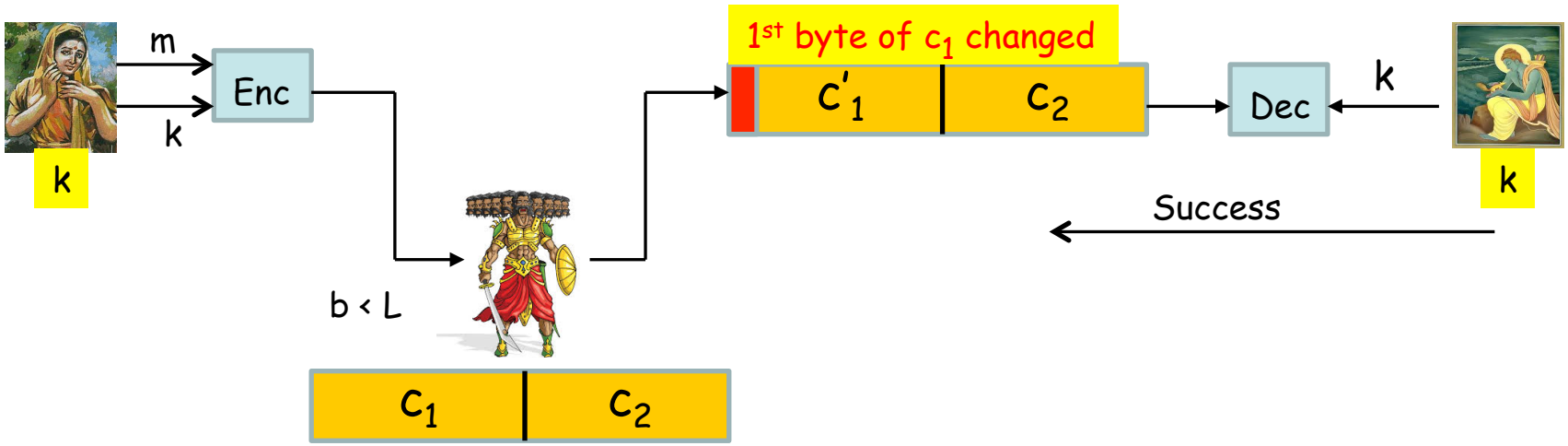
$$m'_2 = F^{-1}(c_2) \oplus c_1$$

$$m_1 = F^{-1}(c_1) \oplus c_0$$

Read the final byte value b

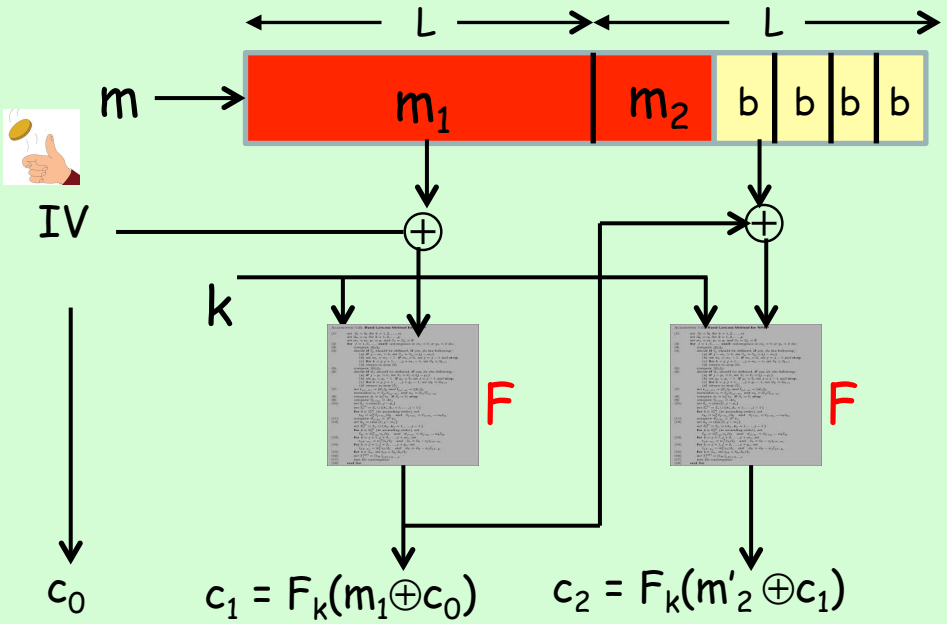
If the last b bytes of m'_2 all have value b then strip-off the pad and output m

Else output bad padding (request for re-transmission)



Padding Oracle Attack on CBC Mode

Encryption



Decryption

Decrypt as per usual CBC-mode decryption and obtain $m_1 \,||\, m'_2$

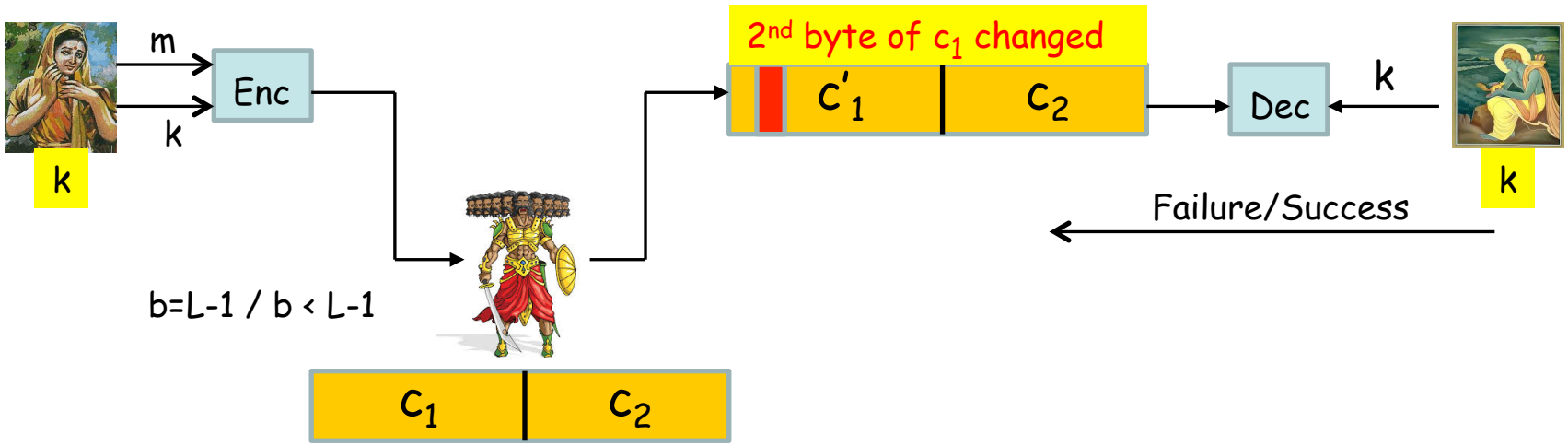
$$m'_2 = F^{-1}(c_2) \oplus c_1$$

$$m_1 = F^{-1}(c_1) \oplus c_0$$

Read the final byte value b

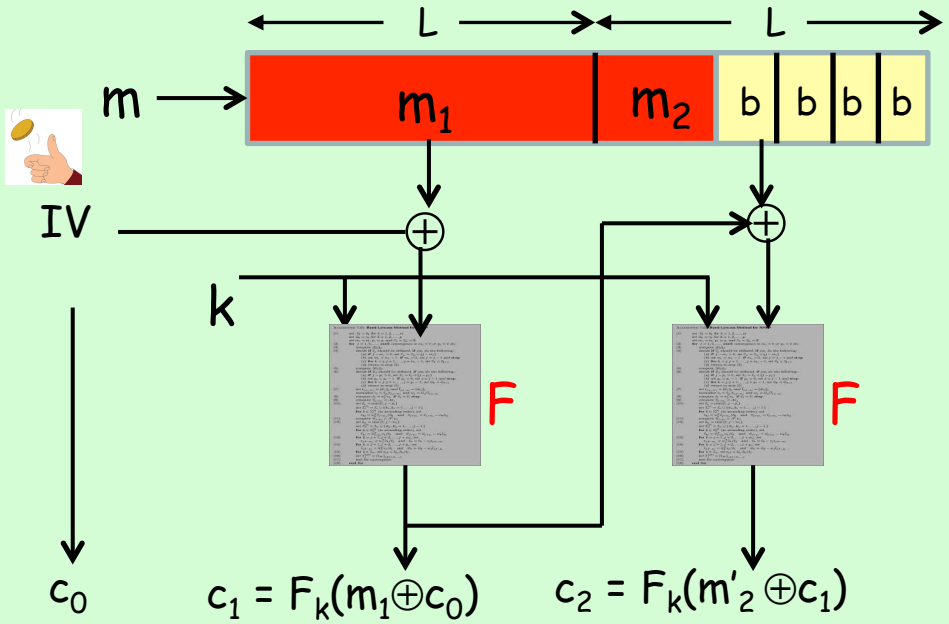
If the last b bytes of m'_2 all have value b then strip-off the pad and output m

Else output bad padding (request for re-transmission)



Padding Oracle Attack on CBC Mode

Encryption



Decryption

Decrypt as per usual CBC-mode decryption and obtain $m_1 \,||\, m'_2$

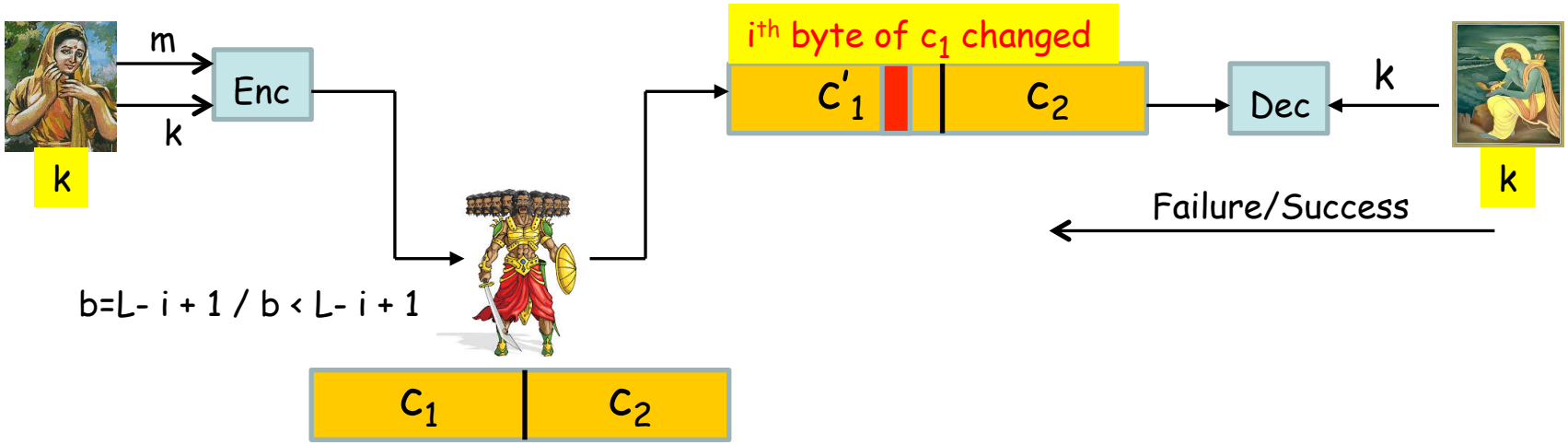
$$m'_2 = F^{-1}(c_2) \oplus c_1$$

$$m_1 = F^{-1}(c_1) \oplus c_0$$

Read the final byte value b

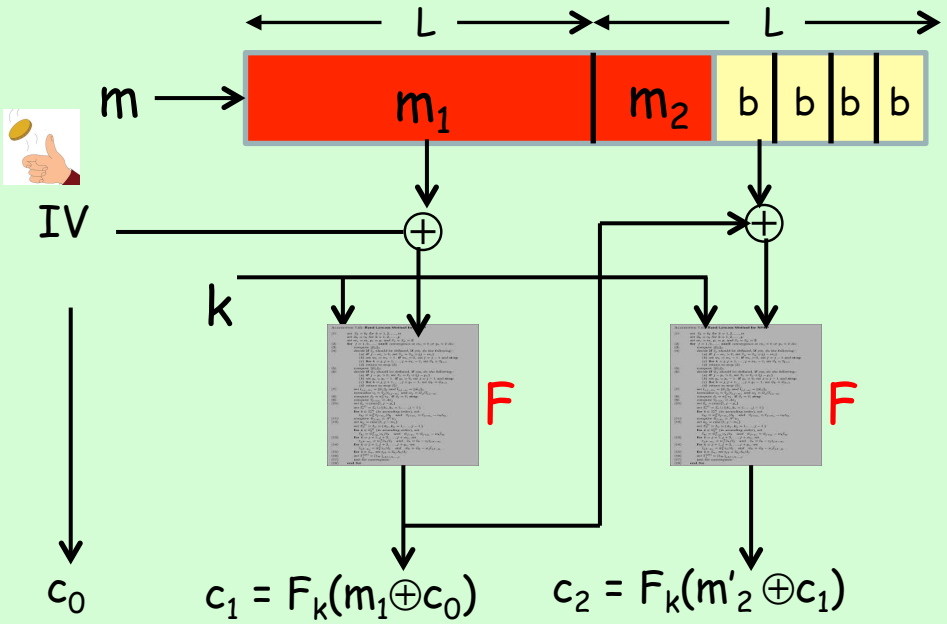
If the last b bytes of m'_2 all have value b then strip-off the pad and output m

Else output bad padding (request for re-transmission)



Padding Oracle Attack on CBC Mode

Encryption



Decryption

Decrypt as per usual CBC-mode decryption and obtain $m_1 \,||\, m'_2$

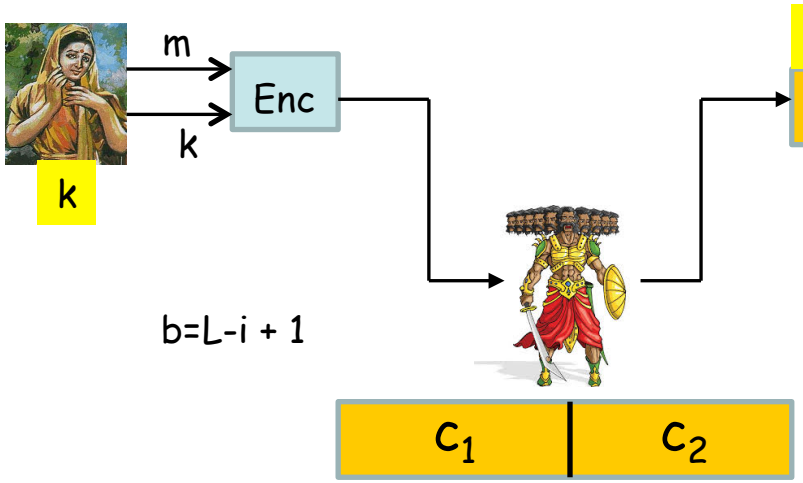
$$m'_2 = F^{-1}(c_2) \oplus c_1$$

$$m_1 = F^{-1}(c_1) \oplus c_0$$

Read the final byte value b

If the last b bytes of m'_2 all have value b then strip-off the pad and output m

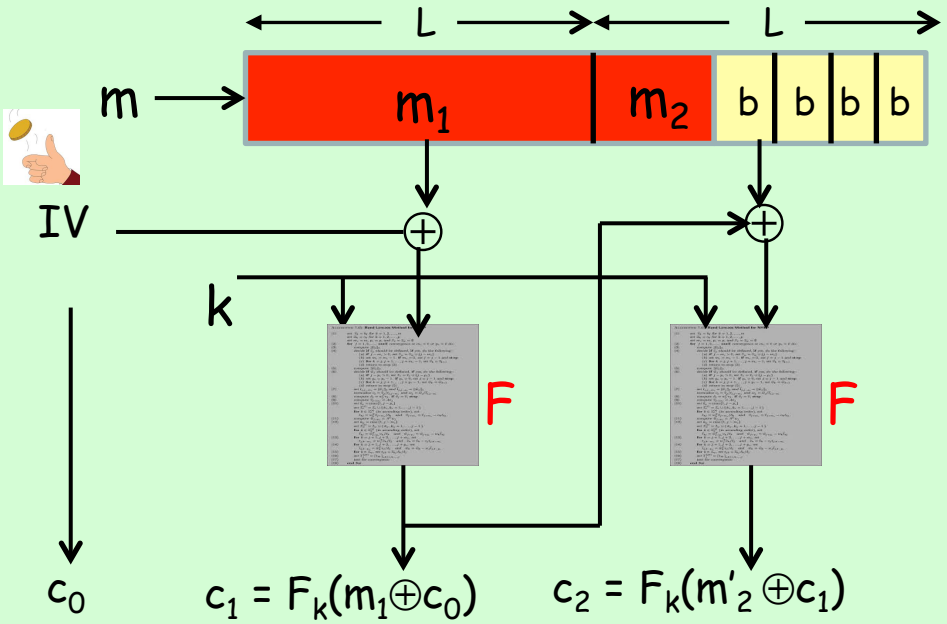
Else output bad padding (request for re-transmission)



- If i is the least indexed modified ciphertext corresponding to which "Failure" comes for then b = L – i + 1 ☺ b is leaked. |m| is leaked!!

Padding Oracle Attack on CBC Mode

Encryption



Decryption

Decrypt as per usual CBC-mode decryption and obtain $m_1 \,||\, m'_2$

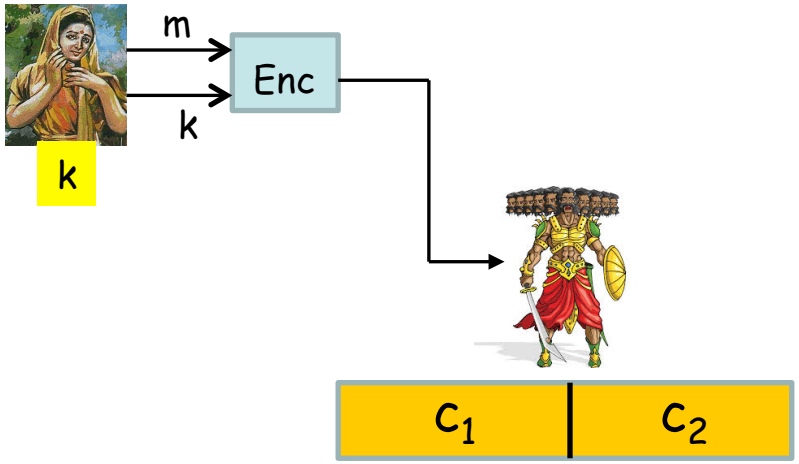
$$m'_2 = F^{-1}(c_2) \oplus c_1$$

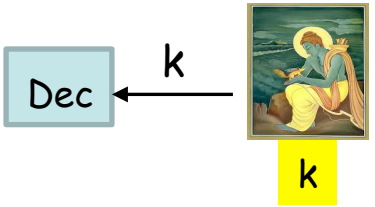
$$m_1 = F^{-1}(c_1) \oplus c_0$$

Read the final byte value b

If the last b bytes of m'_2 all have value b then strip-off the pad and output m

Else output bad padding (request for re-transmission)



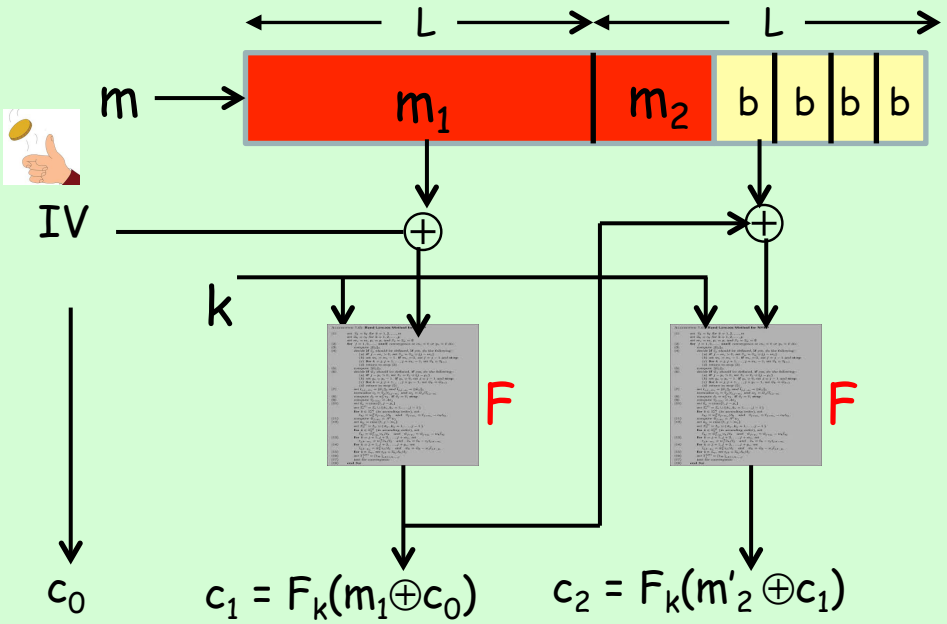


To do: find m.

We will see how adv can find the last byte of m. This can be extended for rest of the message bytes

Padding Oracle Attack on CBC Mode

Encryption



Decryption

Decrypt as per usual CBC-mode decryption and obtain $m_1 \,||\, m'_2$

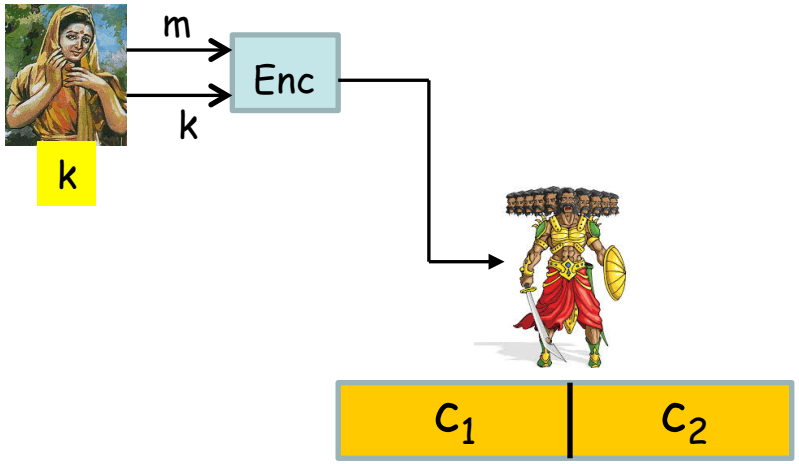
$$m'_2 = F^{-1}(c_2) \oplus c_1$$

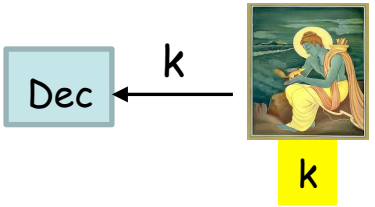
$$m_1 = F^{-1}(c_1) \oplus c_0$$

Read the final byte value b

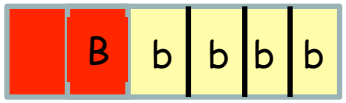
If the last b bytes of m'_2 all have value b then strip-off the pad and output m

Else output bad padding (request for re-transmission)





Once b is known attacker knows m_2 is of the form:



Padding Oracle Attack on CBC Mode

Encryption

m → m_1

IV

k

c_0 $c_1 = F_k(m_1 \oplus c_0)$ $c_2 = F_k(m_2 \oplus c_1)$

Decryption

Decrypt as per usual CBC-mode ...ion and obtain $m_1 \,||\, m'_2$

$m'_2 = F^{-1}(c_2) \oplus c_1$

... $\oplus c_0$

...final byte value b

...e last b bytes of m'_2 all have value ...n strip-off the pad and output m

Else ...tput bad padding (request for re-transmission)

CPA Secure CBC Mode Scheme Broken ☹

Last b+1 bytes of c_1 changed by Δ_1

Δ_1 = (000... 1 (b+1)⊕b (b+1)⊕b (b+1)⊕ b)

m

k

Enc

k

c'_1 c_2

Dec

k

k

Success/Failure

B = b / B ≠ b

B	b	b	b	b

c_1	c_2

	B+1	b+1	b+1	b+1	b+1

Run at most 256 times to know B exactly!!

Padding Oracle Attack

Serge Vaudenay:
Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS .... EUROCRYPT 2002: 534-546

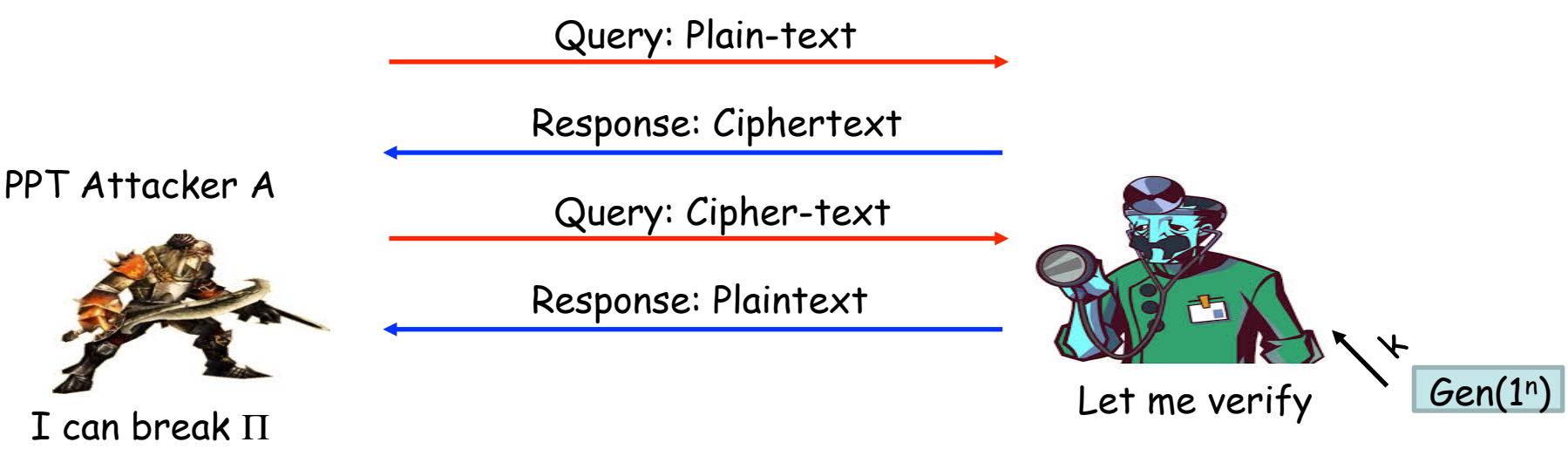
Morale of the Story

- Attacker can have control over "what" is decrypted
  - Will help the attacker to break the secrecy !!
- Remedy:
  - Capture *CCA* in the security definition.
  - Chosen-ciphertext attack (*CCA*) security

CCA Indistinguishability Experiment

$PrivK^{cca}_{A,\Pi}(n)$

Π = (Gen, Enc, Dec), $\mathcal{M}$

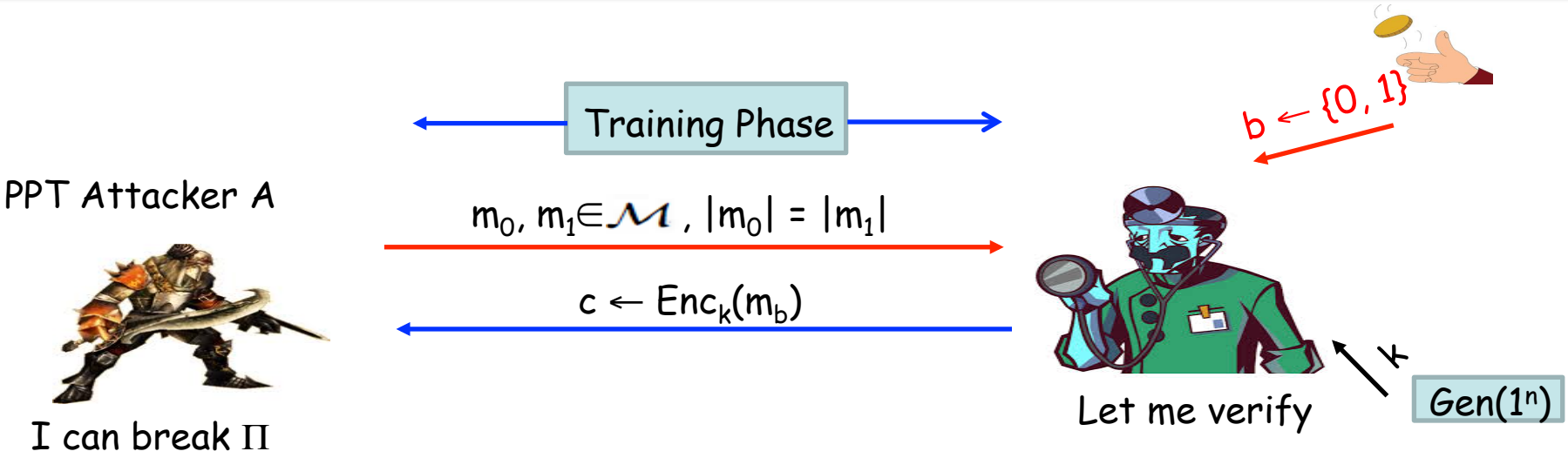


Training Phase:

- A is given oracle access to both $Enc_k()$ and $Dec_k()$
- A adaptively submits its queries (any query is allowed in any order) and receives response

CCA Indistinguishability Experiment

$\text{PrivK}^{cca}_{A,\Pi}(n)$ $\quad \Pi = (Gen, Enc, Dec), \mathcal{M}$

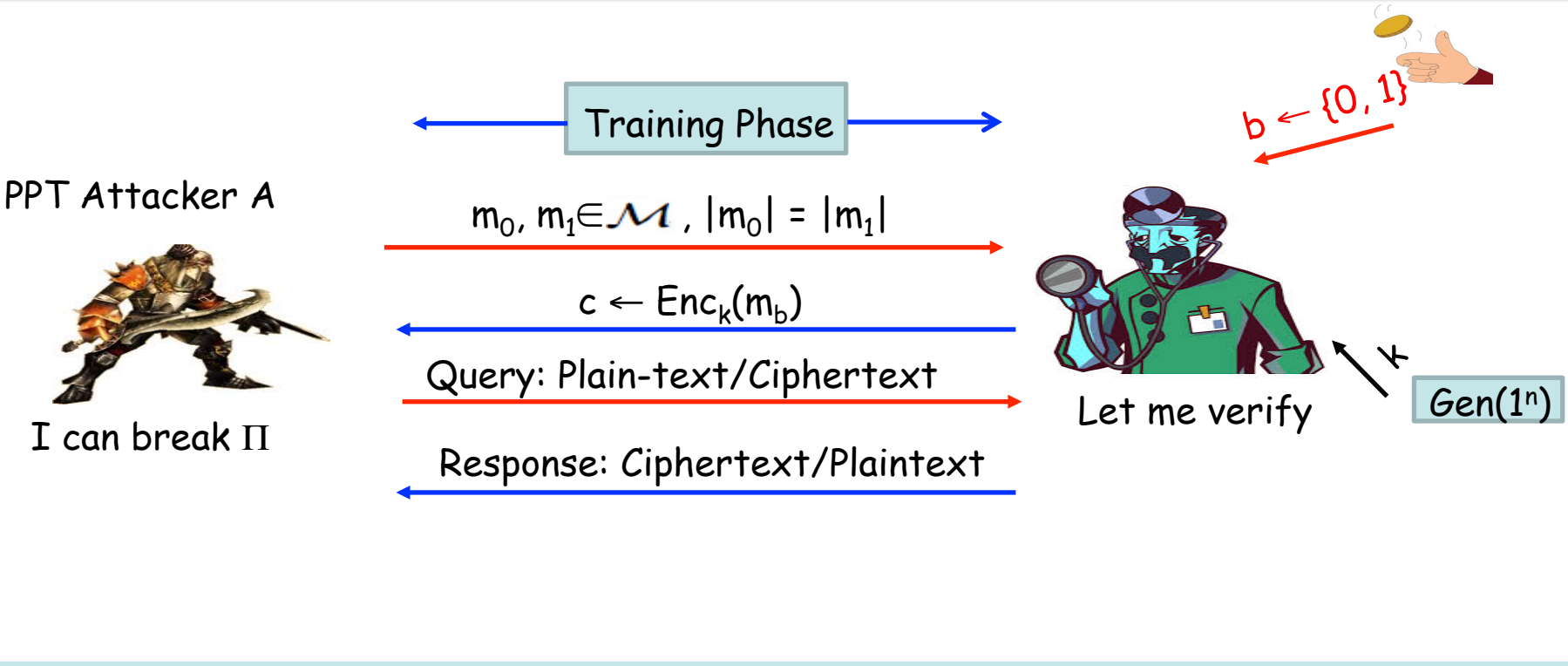


Challenge Phase:

- A submits two equal length challenge plaintexts
- A is free to submit any message of its choice (including the ones already queried during the training phase)
- One of the challenge plaintexts is randomly encrypted for A (using fresh randomness)

CCA Indistinguishability Experiment

$\text{PrivK}^{cca}_{A,\Pi}(n)$ $\quad \Pi = (Gen, Enc, Dec), \mathcal{M}$

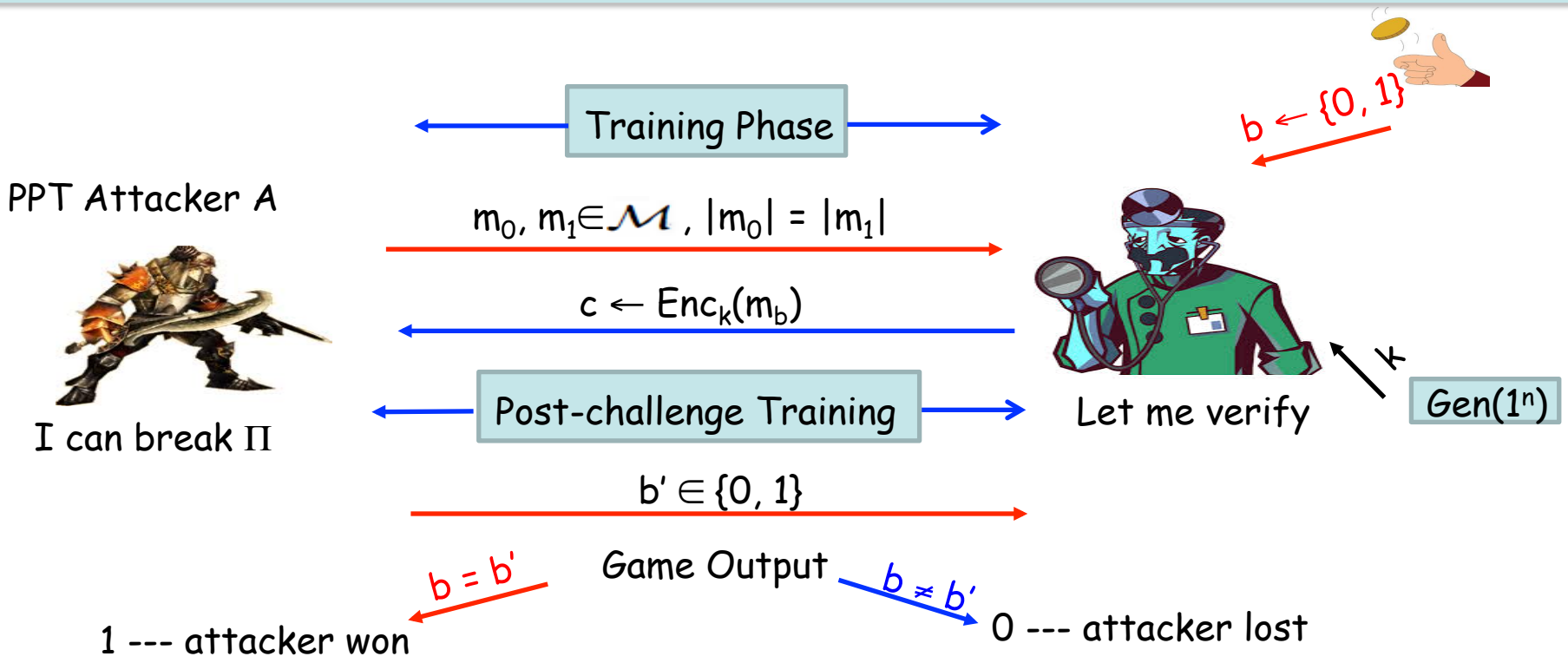


Post-challenge Training Phase:

- A is given oracle access to both $Enc_k()$ and Deck()
- A is restricted from submitting the challenge ciphertext c as the decryption query --- otherwise impossible to achieve any security

CCA Indistinguishability Experiment

$\text{PrivK}^{cca}_{A,\Pi}(n)$ $\quad \Pi = (Gen, Enc, Dec), \mathcal{M}$

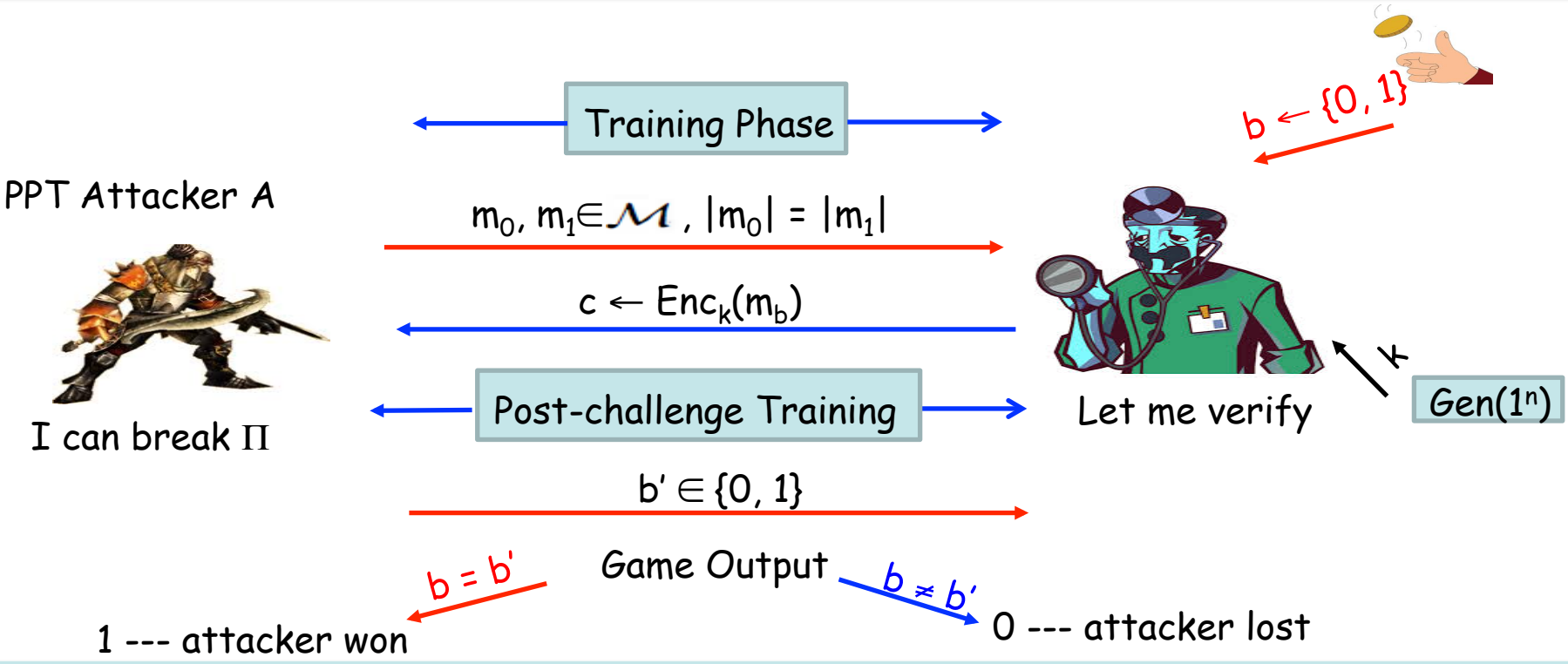


Response Phase:

- A finally submits its guess regarding encrypted challenge plain-text
- A wins the experiment if its guess is correct

CCA Indistinguishability Experiment

$\text{PrivK}^{cca}_{A,\Pi}(n)$ $\quad\quad$ $\Pi = (Gen, Enc, Dec), \mathcal{M}$

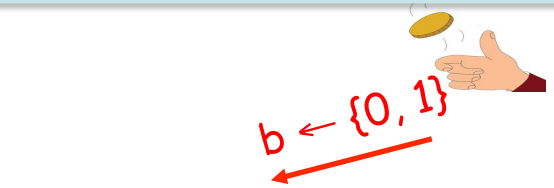


Π is CCA-secure if for every PPT A, there is a negligible function negl, such that:

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

CCA Security for Multiple Encryptions

$\text{PrivK}^{\text{cca-mult}}_{A,\Pi}(n)$ $\quad \Pi = (\text{Gen}, \text{Enc}, \text{Dec}), \mathcal{M}$



Training Phase

PPT Attacker A

$\vec{M_0} = (m_{0,1}, \ldots, m_{0,t})$ $\quad \vec{M_1} = (m_{1,1}, \ldots, m_{1,t})$

(freedom to choose any pair)

$c_1 \leftarrow \text{Enc}_k(m_{b,1}), \ldots, c_t \leftarrow \text{Enc}_k(m_{b,t})$

k

Gen(1^n)

I can break Π

Let me verify

Post-challenge Training

$b' \in \{0, 1\}$

Game Output

b = b'

b ≠ b'

1 --- attacker won

0 --- attacker lost

Π is *CCA-secure for multiple encryptions* if for every PPT A, there is a negligible function negl, such that:

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

CCA Multiple-message vs Single-message Security

- Experiment $\text{PrivK}^{\text{cca}}_{A,\Pi}(n)$ is a special case of $\text{PrivK}^{\text{cca-mult}}_{A,\Pi}(n)$
  - Set $|\vec{M}_0| = |\vec{M}_1| = 1$
- Any cipher that is CCA-secure for multiple encryptions is also CCA-secure (for single encryption)
- What about the converse ?

Theorem: Any cipher that is CCA-secure is also CCA-secure for multiple encryptions

>> Sufficient to prove CCA-security for single message; rest is "for free"

CCA Security is Stronger Than CPA-security

$\text{PrivK}^{cca}_{A,\Pi}(n)$

$\Pi = (\text{Gen}, \text{Enc}, \text{Dec}), \mathcal{M}, n$

$\text{Enc}_k(m) \rightarrow (r, F_k(r) \oplus m)$

PPT Attacker A

I can break Π

$m_0 = (00\ldots0)$ $\quad$ $m_1 = (11\ldots1)$

$b \leftarrow \{0, 1\}$

$c^* = (r, s^*) = (r, F_k(r) \oplus m_b)$

Plz decrypt c = (r, s) for me

(s is same as s* with 1st bit flipped)

$m = \text{Dec}_k(c)$

F

k

$\text{Gen}(1^n)$

Let me verify

$b' = 0$ if $m = 100\ldots0$

$b' = 1$ if $m = 011\ldots1$

- No encryption-oracle service used in the above attack !!
- What is the probability of A winning the game above ?
  - If $m_b = (00\ldots0)$ then $m = (100\ldots0)$. So A outputs $b' = 0 = b$ with probability 1
  - If $m_b = (11\ldots1)$ then $m = (011\ldots1)$. So A outputs $b' = 1 = b$ with probability 1

Towards Achieving CCA-Security

What capability of adv lets him win?

>> Easy to manipulate known ciphertexts to obtain new ciphertexts so that the relation between the underlying messages are known to him..then he gets DO service on the changed ciphertext to get the message.. Using the relation retrieve the original message

>> This is called malleability. CPA-secure scheme does not guarantee non-malleability

Need a SKE so that

>> Creating a new ciphertext will be nearly impossible…

>> Changing a ciphertext should either result in an incorrect ciphertext or should decrypt to a plaintext which is unrelated to the original plaintext

>> Together, the above two makes DO useless to the adversary.

Message Authentication Codes (MAC) helps us to get such a cipher!!

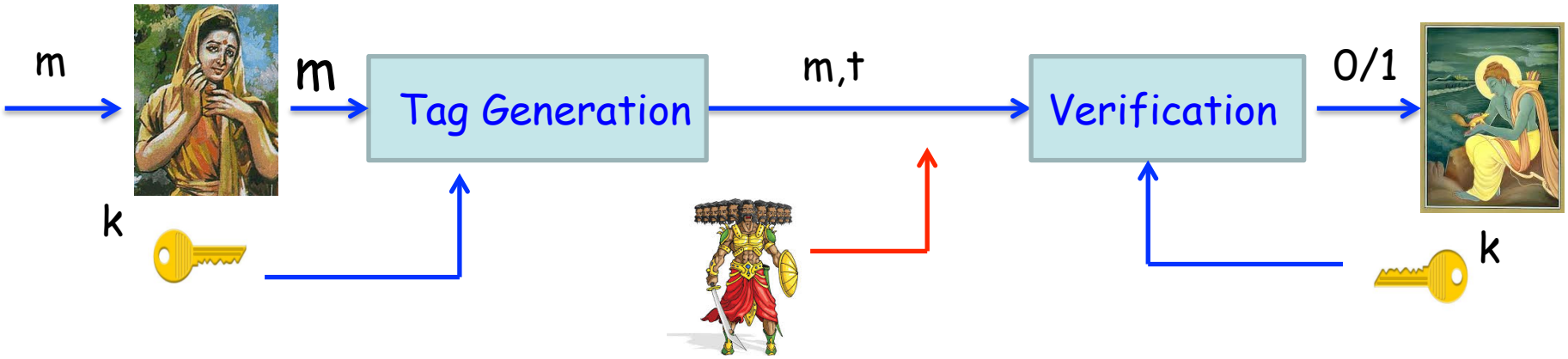
Message Integrity and Authentication

- In secure-communication, is it enough to keep privacy of the message?
  - What is the guarantee that a message received by R indeed originated from S and vice-versa ? --- issue of message authentication
  - Even if it is confirmed that the message received by R originated from S, what is the guarantee that the message content is genuine ? --- issue of message integrity
  - Message integrity and authentication are also part of secure communication

- Message authentication/integrity is important even when privacy is not a concern
  - Any kind of access control system needs them. Think of bank, institute, any organization

- Encryption scheme does not help (unless designed with specific purpose of MI and MA).
  - Consider all the CPA secure schemes considered so far (PRF-based, modes of operations); none provide MI/MA
  - Spoofing attack is easy. Changing ciphertext and thereby changing the underlying message is easy!!

Message Authentication in Private Key Setting



- Secret key k shared in advance (by "some" mechanism)
- m is the plain-text
- t is the tag
- Symmetry: same key used for encryption and decryption

Syntax of Message Authentication Codes (MAC)

A MAC is a 3-tuple (Gen, Mac, Vrfy) of algorithms with the following syntax

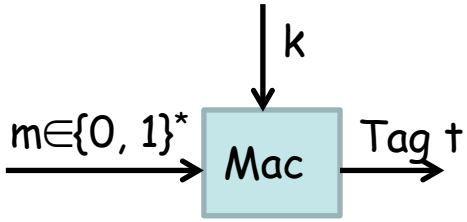
1. Key-generation Algorithm ($Gen(1^n)$):

Input: 1^n

Output: key k (usually uniform at random from $\{0,1\}^n$

Running time: $O(Poly(n))$; MUST be randomized

2. Tag Generation Algorithm ($Mac_k(m)$; m from $\{0,1\}^*$:

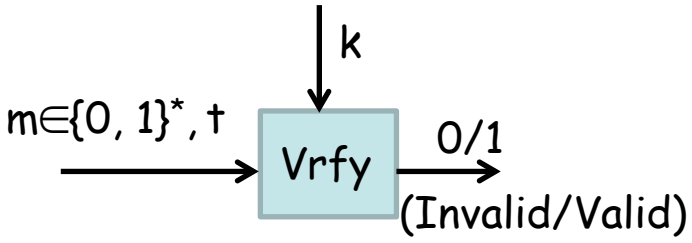


Input: m,k

Output: Tag t

Running time: $O(Poly(n))$; Deterministic/Randomized

3. Veification Algorithm ($Vrfy_k(m,t)$):



Input: (m,t),k

Output: 0/1

Running time: $O(Poly(n))$; Deterministic (usually)

Syntax of MAC

- Any MAC defines the following three space (sets):

1. Key space (K):
   - Set of all possible keys output by algorithm Gen
2. Plain-text (message) space (M):
   - Set of all possible "legal" message (i.e. those supported by Mac)
3. Tag space (T):
   - Set of all tags output by algorithm Mac
   - The sets M and K together define the set T

- Any MAC is defined by specifying (Gen, Mac, Vrfy) and M

Correctness: For every n, every k output by Gen and every message m the following should hold :

$$Vrfy_k(m, Mac_k(m)) = 1$$

Towards Defining Security of MAC

Two components of a security definition:

Threat: >> Computationally Bounded / negligible success prob.

>> Randomized

>> What kind of attacks he can mount?

- Chosen Message Attack (CMA) --- in spirit of CPA; models the fact that adv can influence the honest parties to authenticate a message of its choice.
- Chosen Message and Verification Attack (CMVA) --- in spirit of CCA models the fact that the adv can influence the honest parties to authenticate messages and verify tag, message pair of its choice.

Break: >> New (m,t) pair such that adv has not seen a tag on m

>> New (m,t) such that adv has not seen (m,t) before--stronger notion

MAC Experiment

Experiment Mac-forge$_{A, \Pi}$(n)

Π = (Gen, Mac, Vrfy), n

Attacker A

I can break Π

Run time: Poly(n)

Plz give me the tag for m_1

$t_1 \leftarrow Mac_k(m_1)$

Let me verify

Gen(1^n)

- Training phase :
  - A gets tag for several messages of its choice adaptively --- access to Mac-oracle

MAC Authentication Experiment

Experiment Mac-forge$_{A, \Pi}$(n)

Π = (Gen, Mac, Vrfy), n

Attacker A

I can break Π

Run time: Poly(n)

Q = {(m_1, …,m_l }

Training Phase

(m, t)

Forged tag generated by A

Let me verify

k

Gen(1^n)

game output

- 1 (A succeeds) if $Vrfy_k(m, t) = 1$ and $m \notin Q$
- 0 (A fails) otherwise

Π is existentially unforgeable under an adaptive chosen message attack or CMA- secure if

$$Pr[\text{Mac-forge}_{A, \Pi}(n) = 1] \leq \text{negl}(n)$$

MAC Authentication Experiment

Experiment Mac-sforge$_{A, \Pi}$ (n)

Π = (Gen, Mac, Vrfy), n

Attacker A

I can break Π

Run time: Poly(n)

$Q = \{(m_1, t_1), \ldots, (m_l, t_l)\}$

Training Phase

(m, t)

Forged tag generated by A

Let me verify

k

Gen(1^n)

game output

- 1 (A succeeds) if $Vrfy_k(m, t) = 1$ and $(m, t) \notin Q$
- 0 (A fails) otherwise

Π is existentially unforgeable under an adaptive chosen message attack or strong CMA-secure if

$$\Pr[\text{Mac-sforge}_{A, \Pi}(n) = 1] \leq \text{negl}(n)$$

What is not Captured in MAC Security Definition

- If A returns (m,t) for a already queried message, we don't consider that as the break.

  >> What it captures in real scenario? if (m,t) is a valid pair generated by the sender, then there is no harm if the receiver accepts it even though adv forwards it (may be at a later point of time)

  >> Is it problematic?

  >> Let a bank user X sends the following instruction to the bank:

  "transfer $1000 from account #X to account #Y"

  >> What if an attacker simply sends 10 copies of the original (message, tag) pair --Bank will consider each request genuine --- disaster for X

  >> The above attack is called replay attack

- Why Replay Attack is not taken care in MAC Definition

  >> Whether this attack is of concern depends on actual application scenario

  >> So it is better to deal with this in the outer protocol (that uased MAC for authentication)

  >> Additional techniques like (synchronized) counters, timestamp, etc are used

Thank You!