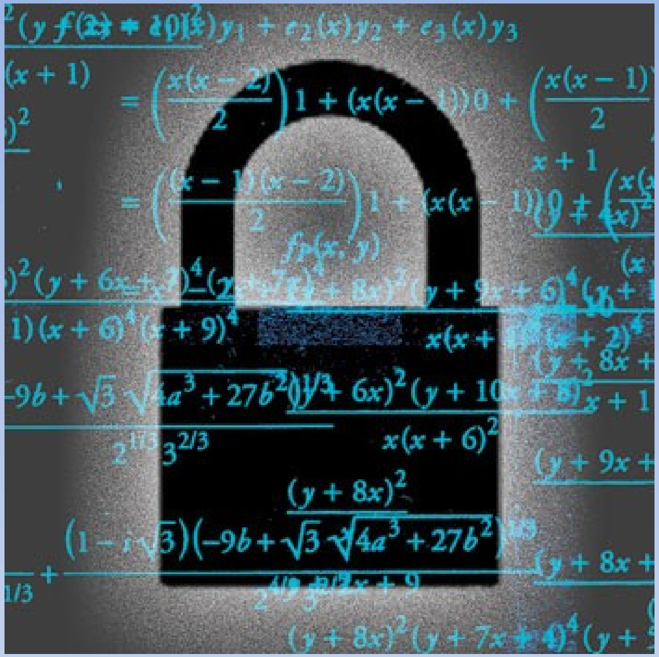


Cryptography
Lecture – Symmetric key Cryptanalysis

Dr. Panagiotis Rizomiliotis

Agenda

- Encryption Security
- Padding attack
- Hash security
- MAC security

Confidentiality

SYMMETRIC ENCRYPTION SCHEMES

Security

- *perfect security,*
  - *an information-theoretic notion introduced by* Shannon and showed by him to be met by the one-time pad scheme.
  - *regardless of the computing power available to the adversary, the ciphertext provides it no information about the plaintext beyond the a priori information it had prior to seeing the ciphertext*
  - it requires a key as long as the total amount of data encrypted
- *computational security*
  - The security will only hold with respect to adversaries of limited computing power.

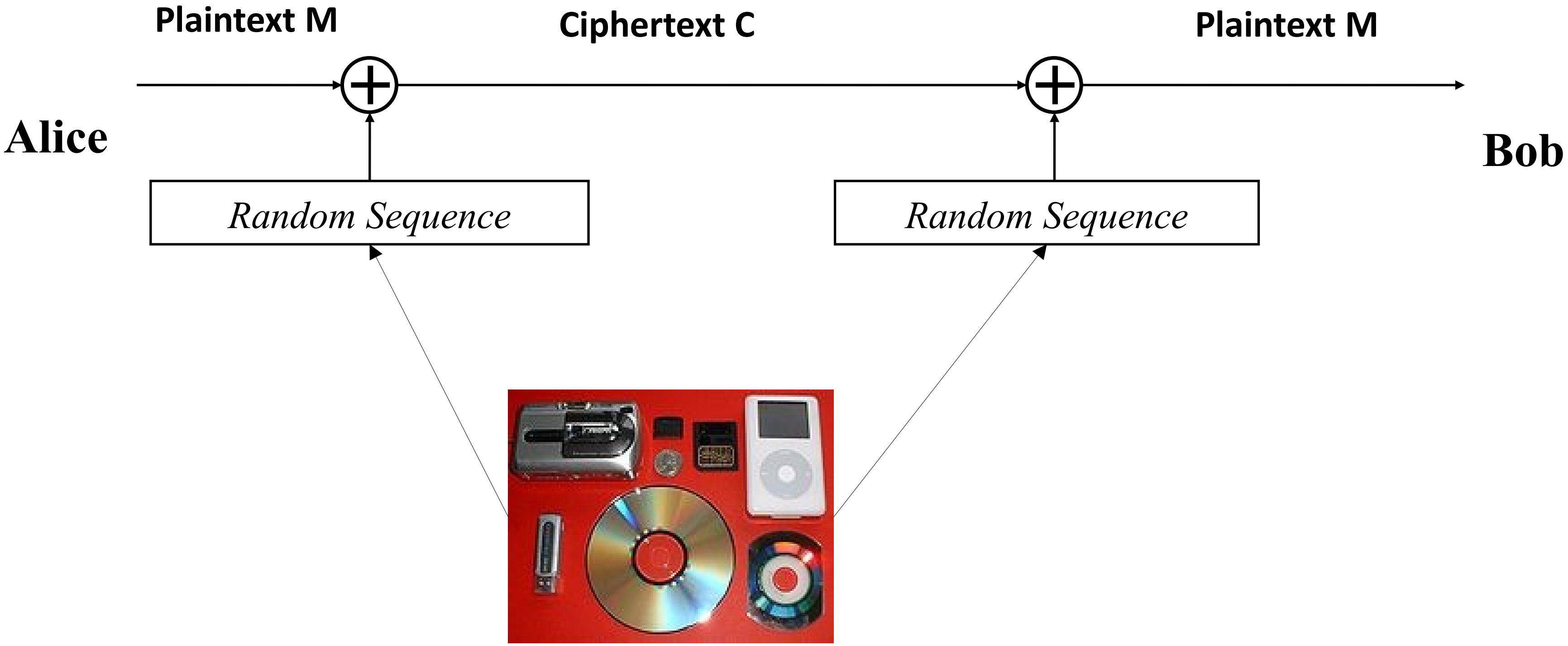
Shannon's perfect secrecy definition

Let (E,D) be a cipher over (K,M,C)

(E,D) has perfect secrecy if $\forall\ m_0, m_1 \in M \quad (\ |m_0| = |m_1|\ )$

$\{\ E(k,m_0)\ \} \quad = \quad \{\ E(k,m_1)\ \}$ where $k \longleftarrow K$

One-time pas has perfect secrecy



Plaintext M

Ciphertext C

Plaintext M

Alice

Bob

Random Sequence

Random Sequence

Perfect Secrecy

Theorem

One time pad has perfect secrecy.

- Proof: easily using Information Theory

Theorem

Perfect secrecy implies that the size of the key K (i.e. the One-time pad's random sequence) must be greater or equal to plaintext M

Thus, the key must be used only once. Impractical!

Quiz

Can a stream cipher have perfect secrecy?

Yes, if the PRG is really “secure”

No, there are no ciphers with perfect secrecy

Yes, every cipher has perfect secrecy

No, since the key is shorter than the message

Quiz

Can a stream cipher have perfect secrecy?

Yes, if the PRG is really “secure”

No, there are no ciphers with perfect secrecy

Yes, every cipher has perfect secrecy

[✓ No, since the key is shorter than the message]

Two times padding/ re-using the IV (attack 1)

- Let
  - $c1 = m1 \oplus RS$
  - $c2 = m2 \oplus RS$
- Eve eavesdrops c1,c2

1st attack: Known Plaintext Attack

Let m1 be a known plaintext to Eve. Then trivially:

$$c1 \oplus c2 \oplus m1 = m1 \oplus RS \oplus m2 \oplus RS \oplus m1 = m2$$

Example

- Alice (two times the same keystream)

m1: 0 1 1 0 1 1 1

$\oplus$

RS: 1 0 1 1 0 1 0

---

c1: 1 1 0 1 1 0 1

m2: 1 0 0 1 0 1 1

$\oplus$

RS: 1 0 1 1 0 1 0

---

c2: 0 0 1 0 0 0 1

Example

- Eve

m1: 0 1 1 0 1 1 1

c2: 0 0 1 0 0 0 1

$\oplus$

c1: 1 1 0 1 1 0 1

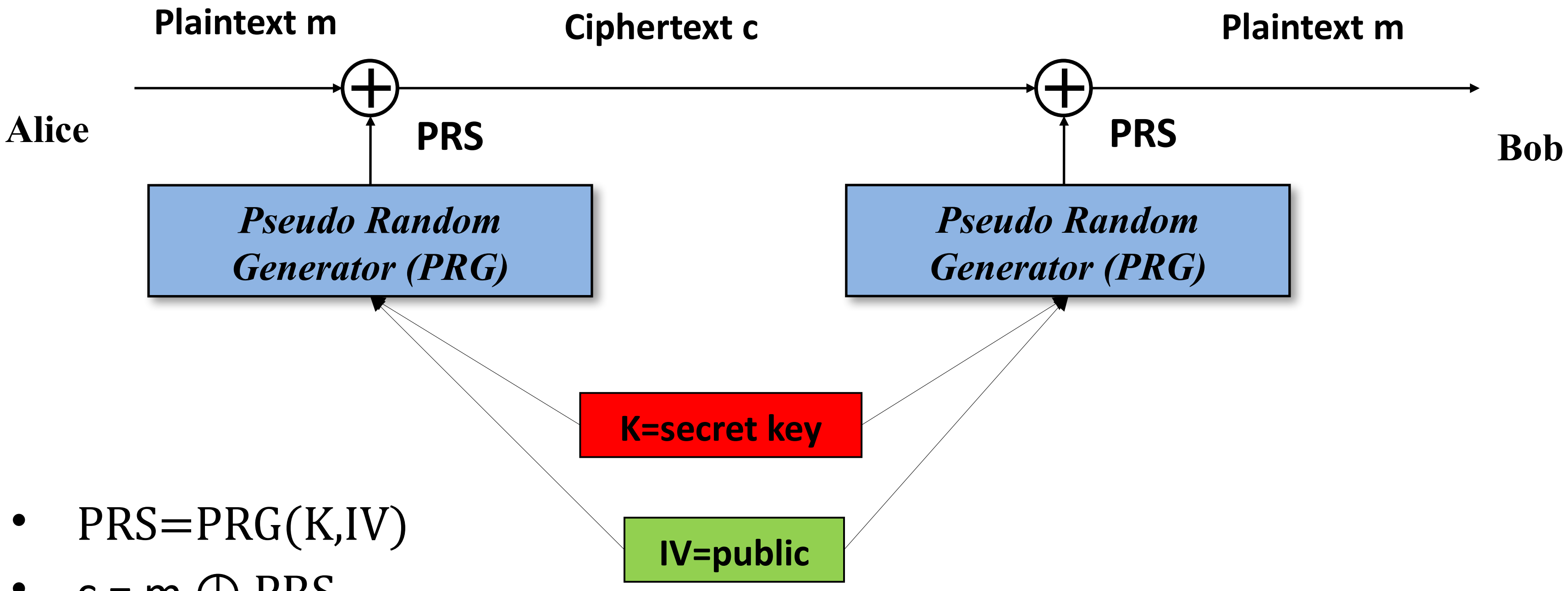
---

m2: 1 0 0 1 0 1 1

Stream Ciphers

- One time padding is also a stream cipher requirement, since the same attack applies
- Remember that:
  - The generator PRG produces a pseudorandom sequence PRS
    - $PRS = PRG(K, IV)$
    - c = m $\oplus$ PRS

Stream Ciphers (synchronous)



- PRS=PRG(K,IV)
- $c = m \oplus PRS$

Stream Ciphers

- When the same key/IV pair is used the generator produces the same PRS
- Thus, we have
  - c1 = m1 $\oplus$ PRS
  - c2 =m2 $\oplus$ PRS
- The same attach. The IV must never repeat for the same key.

Example

- Alice (two times the same IV)

m1: 0 1 1 0 1 1 1

⊕

PRS: 1 0 1 1 0 1 0

---

c1: 1 1 0 1 1 0 1

m2: 1 0 0 1 0 1 1

⊕

PRS: 1 0 1 1 0 1 0

---

c2: 0 0 1 0 0 0 1

Example

- Eve

m1: 0 1 1 0 1 1 1

c2: 0 0 1 0 0 0 1

⊕

c1: 1 1 0 1 1 0 1

---

m2: 1 0 0 1 0 1 1

Two times padding/ re-using the IV (attack 2)

- Let
  - $c1 = m1 \oplus RS$ (or PRS)
  - $c2 = m2 \oplus RS$ (or PRS)
- Eve eavesdrops c1,c2

2nd attack: Known Plaintext Statistics

- Eve computes:

  $c=c1 \oplus c2=m1 \oplus RS \oplus m2 \oplus RS =m1\oplus m2$

  - Eve combines the (most probable) values of m1 and m2 until she produces c
  - It is an efficient way to find candidate pairs (m1,m2)

Example

- Alice (two times the same keystream)

m1: 0 1 1 0 1 1 1

⊕

k: 1 0 1 1 0 1 0

---

c1: 1 1 0 1 1 0 1

m2: 1 0 0 1 0 1 1

⊕

k: 1 0 1 1 0 1 0

---

c2: 0 0 1 0 0 0 1

Example

- Eve

```
c1:   1 1 0 1 1 0 1
                        ⊕
c2:    0 0 1 0 0 0 1
--------------------------
c:     1 1 1 1 1 0 0
```

Trivial leakage:

When the bits of c are zero then the corresponding bits of m1 and m2 are the same

Example

- Eve

$$\begin{array}{ll} c1: & 1\ 1\ 0\ 1\ 1\ 0\ 1 \\ & \qquad\qquad\qquad\qquad \oplus \\ c2: & 0\ 0\ 1\ 0\ 0\ 0\ 1 \\ \hline c: & 1\ 1\ 1\ 1\ 1\ 0\ 0 \end{array}$$

Scenario:
Let any m used by Alice be of the form

m=X||D

where X is one of {111, 010,011,000}.

Then, m1 = X1|D1 and m2= X2|D2

We have that for the different possible X:

$$111 \oplus 010 = 101$$
$$111 \oplus 011 = 100$$
$$010 \oplus 011 = 001$$
$$111 \oplus 000 = 111$$
$$011 \oplus 000 = 011$$
$$010 \oplus 000 = 010$$

Since the last 3 bits of c are 100, then the last 3 of m1, m2 are {111,011}/ we don't know which is which.

We can improve the attack we more ciphertexts

Recall Shannon's perfect secrecy

Let (E,D) be a cipher over (K,M,C)

(E,D) has perfect secrecy if $\forall\, m_0, m_1 \in M \quad (\; |m_0| = |m_1| \;)$

$$\{ E(k,m_0) \} \quad = \quad \{ E(k,m_1) \} \qquad \text{where} \quad k \longleftarrow K$$

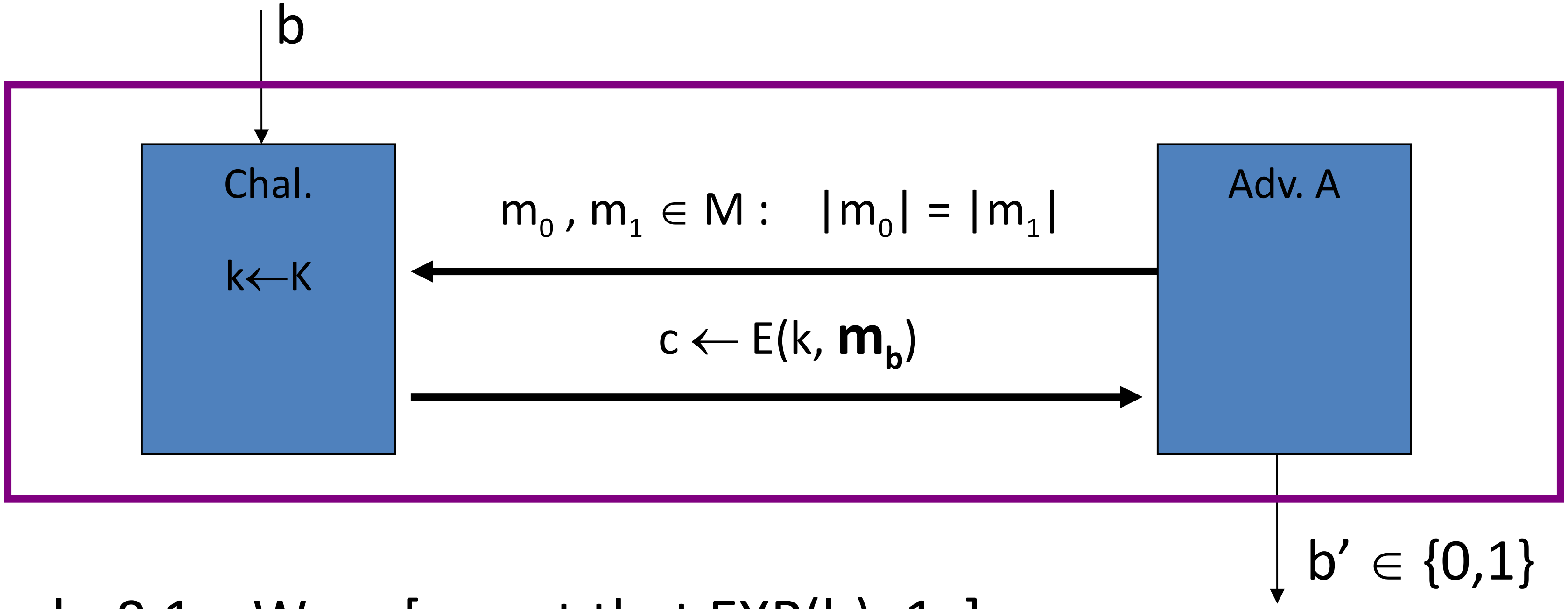
(E,D) has (almost) perfect secrecy if $\forall\, m_0, m_1 \in M \; (|m_0| = |m_1| \;)$

$$\{ E(k,m_0) \} \;\approx_p\; \{ E(k,m_1) \} \qquad \text{where} \quad k \longleftarrow K$$

... but also need adversary to exhibit $m_0, m_1 \in M$ explicitly

Semantic Security/ IND-CPA

For b=0,1 define experiments EXP(0) and EXP(1) as:


b
Chal.
k←K
$m_0, m_1 \in M$: $|m_0| = |m_1|$
$c \leftarrow E(k, \mathbf{m_b})$
Adv. A
$b' \in \{0,1\}$

for b=0,1: W_b := [event that EXP(b)=1]

$$\mathrm{Adv}_{SS}[A,E] := \big|\ \Pr[\ W_0\ ] - \Pr[\ W_1\ ]\ \big| \quad \in [0,1]$$

Semantic Security (one-time key)

Def: $\mathbb{E}$ is **semantically secure** if for all efficient A

$\text{Adv}_{SS}[A,\mathbb{E}]$ is negligible.

- Indistinguishability under chosen-plaintext attack (IND-CPA)

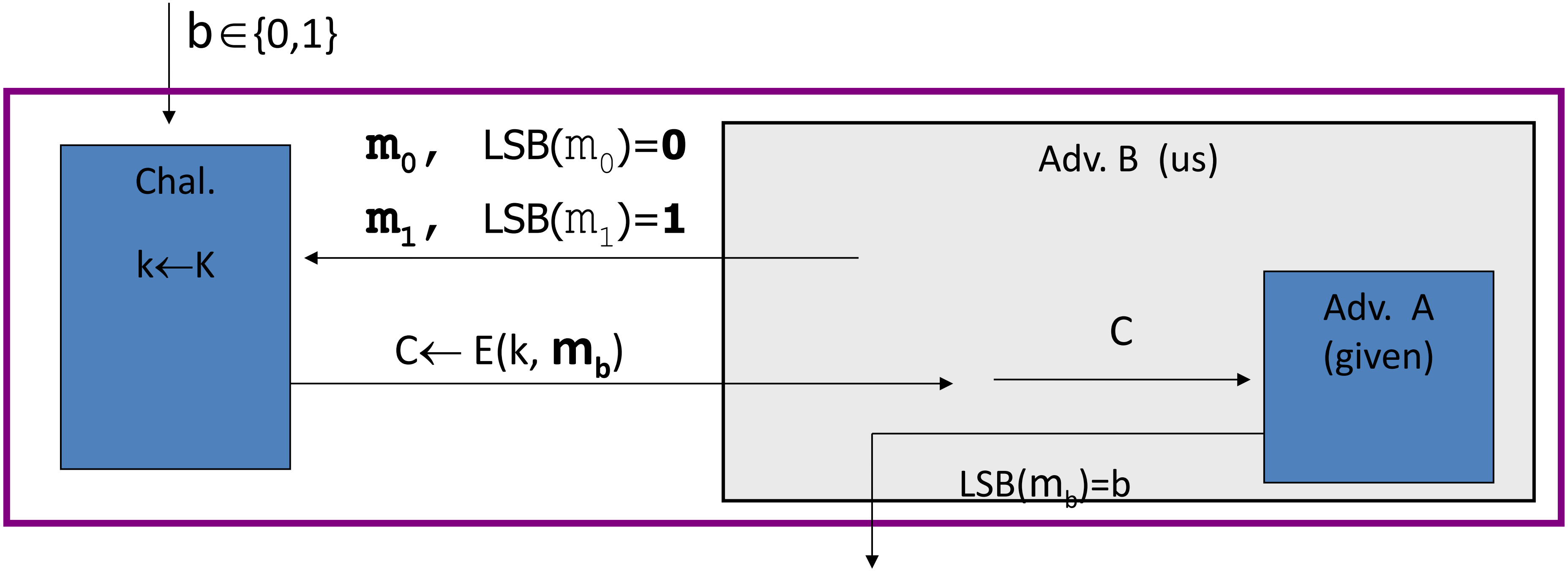
⇒ for all explicit $m_0 , m_1 \in M$:

$$\{ E(k,m_0) \} \approx_p \{ E(k,m_1) \}$$

Example 1

Suppose efficient A can always deduce LSB of PT from CT.

⇒ $\mathbb{E}$ = (E,D) is not semantically secure.



$b \in \{0,1\}$
Chal.
$k \leftarrow K$
$\mathbf{m}_0$, $\mathrm{LSB}(m_0)=\mathbf{0}$
$\mathbf{m}_1$, $\mathrm{LSB}(m_1)=\mathbf{1}$
$C \leftarrow E(k, \mathbf{m}_b)$
Adv. B (us)
C
Adv. A (given)
$\mathrm{LSB}(m_b)=b$

Then $\mathrm{Adv}_{SS}[B, \mathbb{E}] = \Big|\, \Pr[\, \mathbf{EXP(0)}=1\,] - \Pr[\, \mathbf{EXP(1)}=1\,] \,\Big| = |0 - 1| = 1$

Example 1

- When algorithm A works with probability p (not certain) then the attack is the same, only the advantage changes.
- Example, p = 0.8

Then $\text{Adv}_{SS}[B, \mathbb{E}] = \left| \, 0{,}8 - 0{,}2 \, \right| = |0{,}6| = 0{,}6$

Example 2

- Έστω ότι ο (E,D) είναι ένας semantically secure cipher όπου ο χώρος του μηνύματος και του ciphertext είναι $\{0,1\}^n$. Εϊναι το ακόλουθο σχήμα κρυπτογράφησης είναι semantically secure?

$$E'(k,m)=E(k,m)\|\|(\mathrm{LSB}(m)\oplus \mathrm{MSB}(m))$$

Proof (sketch):

1. Use the definition to evaluate the scheme
2. We are looking for two messages such that $\mathrm{LSB}(m)\oplus \mathrm{MSB}(m)$ gives different output
3. Two such messages, for any m, are
   - m0 = (0||m||0)
   - m1 = (1||m||0)

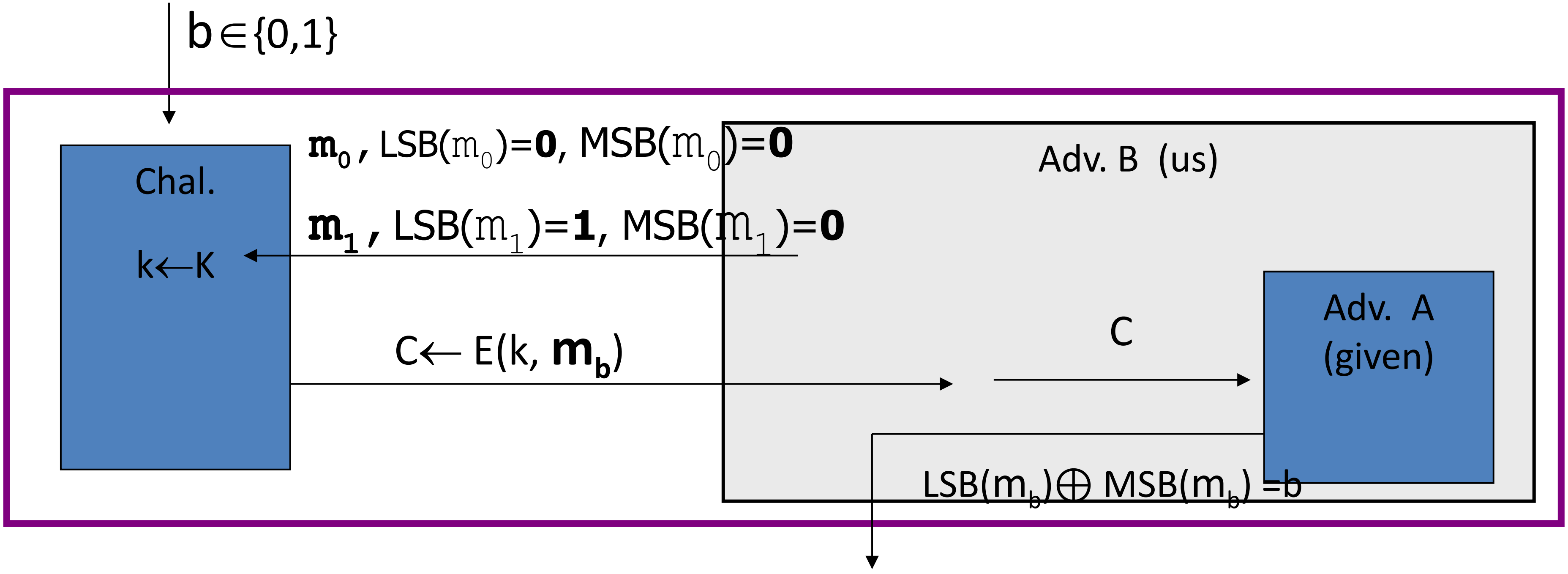
   (there are also other choices of course.)

- The advantage is 1.

Example 2

Suppose efficient A can always deduce LSB$\oplus$ MSB of PT from CT.

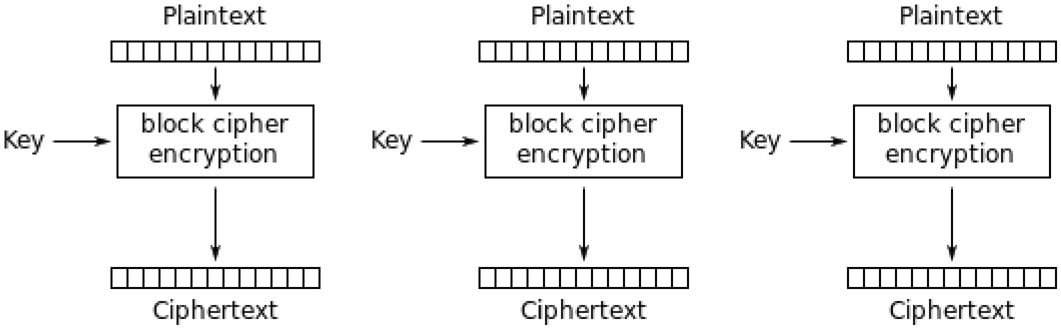
$\Rightarrow$ $\mathbb{E}$ = (E,D) is not semantically secure.


$b \in \{0,1\}$
Chal.
$k \leftarrow K$
$\mathbf{m_0}$, LSB(m_0)=**0**, MSB(m_0)=**0**
$\mathbf{m_1}$, LSB(m_1)=**1**, MSB(m_1)=**0**
$C \leftarrow E(k, \mathbf{m_b})$
Adv. B (us)
C
Adv. A
(given)
LSB(m_b)$\oplus$ MSB(m_b) =b

Then $\mathrm{Adv}_{SS}[B, \mathbb{E}]$ = $\Big|$ Pr[**EXP(0)**=1] − Pr[**EXP(1)**=1] $\Big|$ = $|0 - 1|$ = 1

Block Cipher ECB Mode

- Encryption

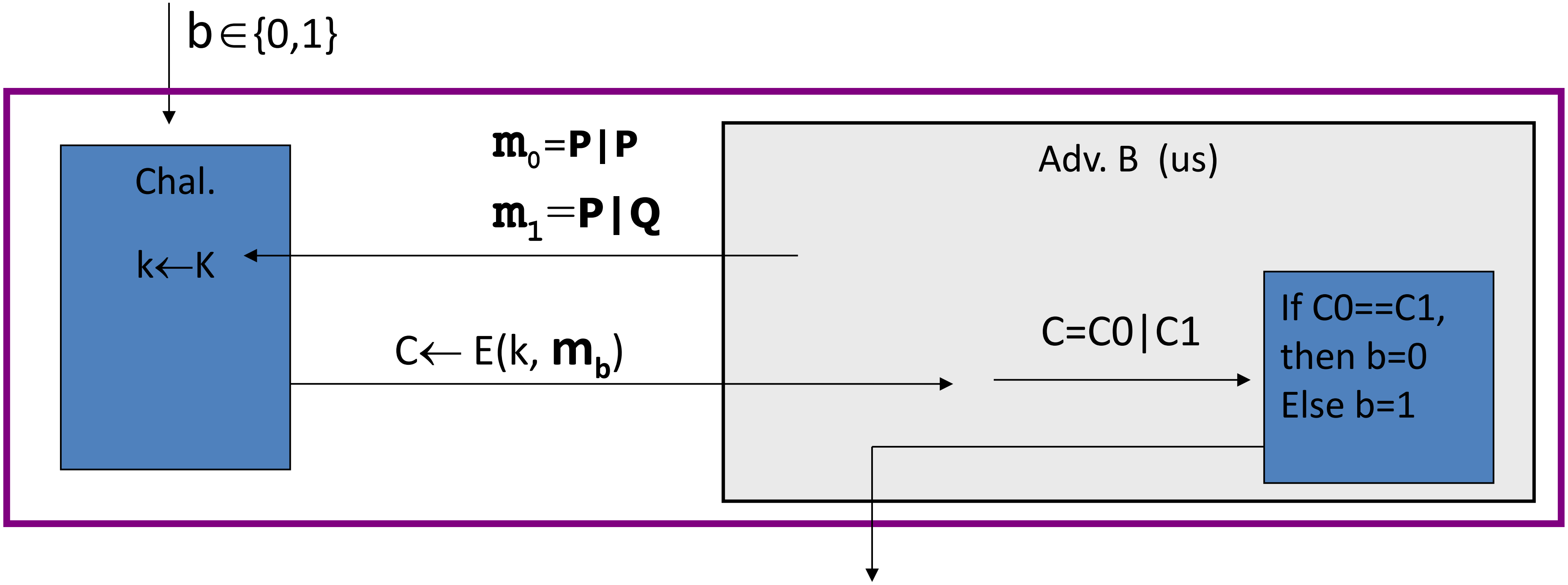


Electronic Codebook (ECB) mode encryption

Example 3 (ECB mode)

Suppose that ECB mode of a block cipher B is used.

⇒ $\mathbb{E}$ = (E,D) is not semantically secure.



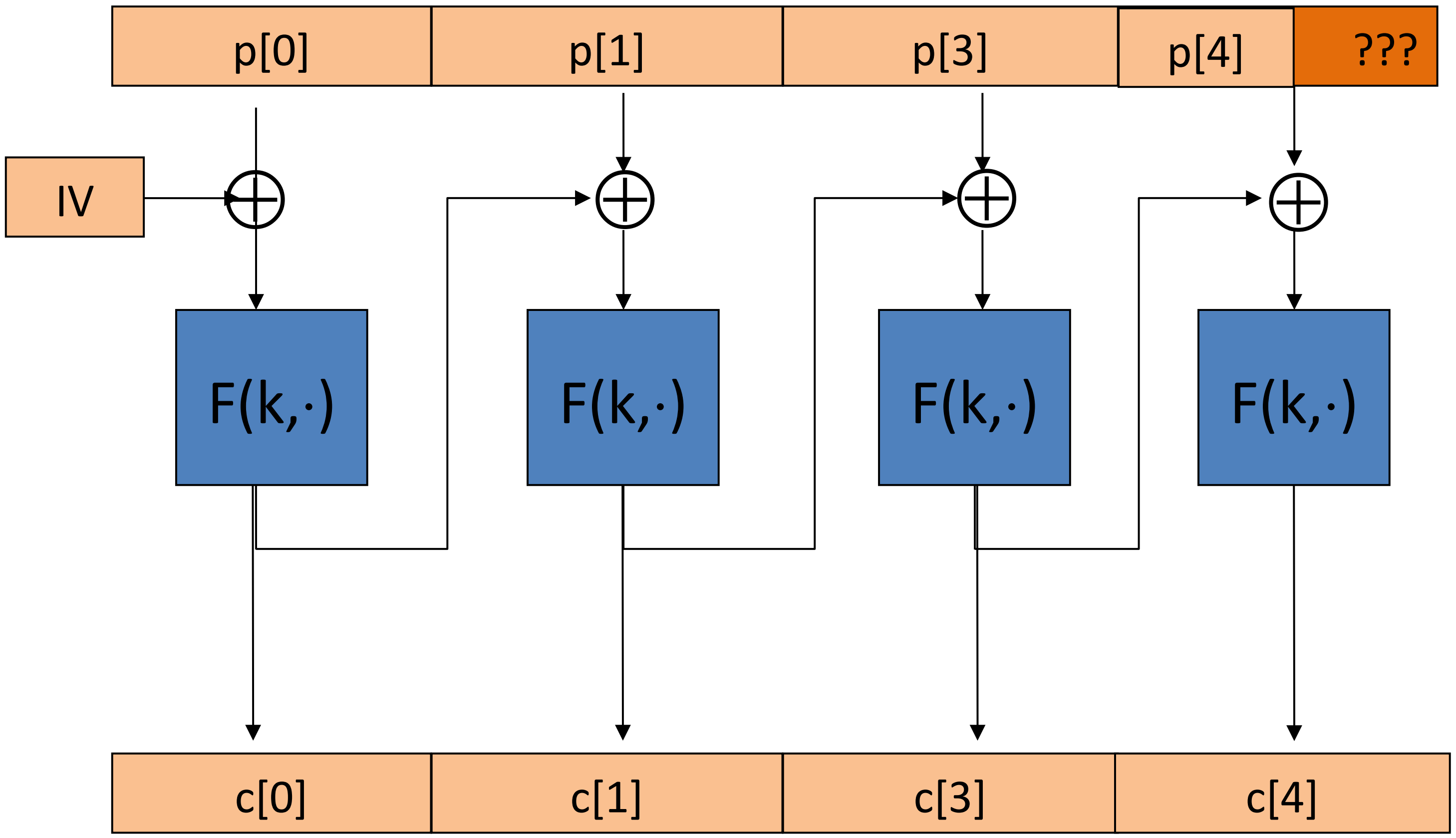
Then $\text{Adv}_{SS}[B, \mathbb{E}] = \Big| \Pr[\,\mathbf{EXP(0)}=1\,] - \Pr[\,\mathbf{EXP(1)}=1\,] \Big| = |0 - 1| = 1$

PADDING

Padding

- Padding it is needed when the input doesn't have the necessary length.
  - For instance when a plaintext is processed in blocks (CBC mode, ECB mode etc
  - When message is processed in blocks (hash functions)
- There are padding oracle attacks
  - An application exhibits padding errors while decrypting a ciphertext
  - The attacker can choose the ciphertext (chosen ciphertext attack)
    - (for instance keywrapping)

Padding in CBC mode



Padding

- There are several padding schemes
  - Zero padding (ex. CBC-CS1/2/3, NIST Special Publication 800-38A)
  - One and zeros padding
  - PKCS5/PKCS7

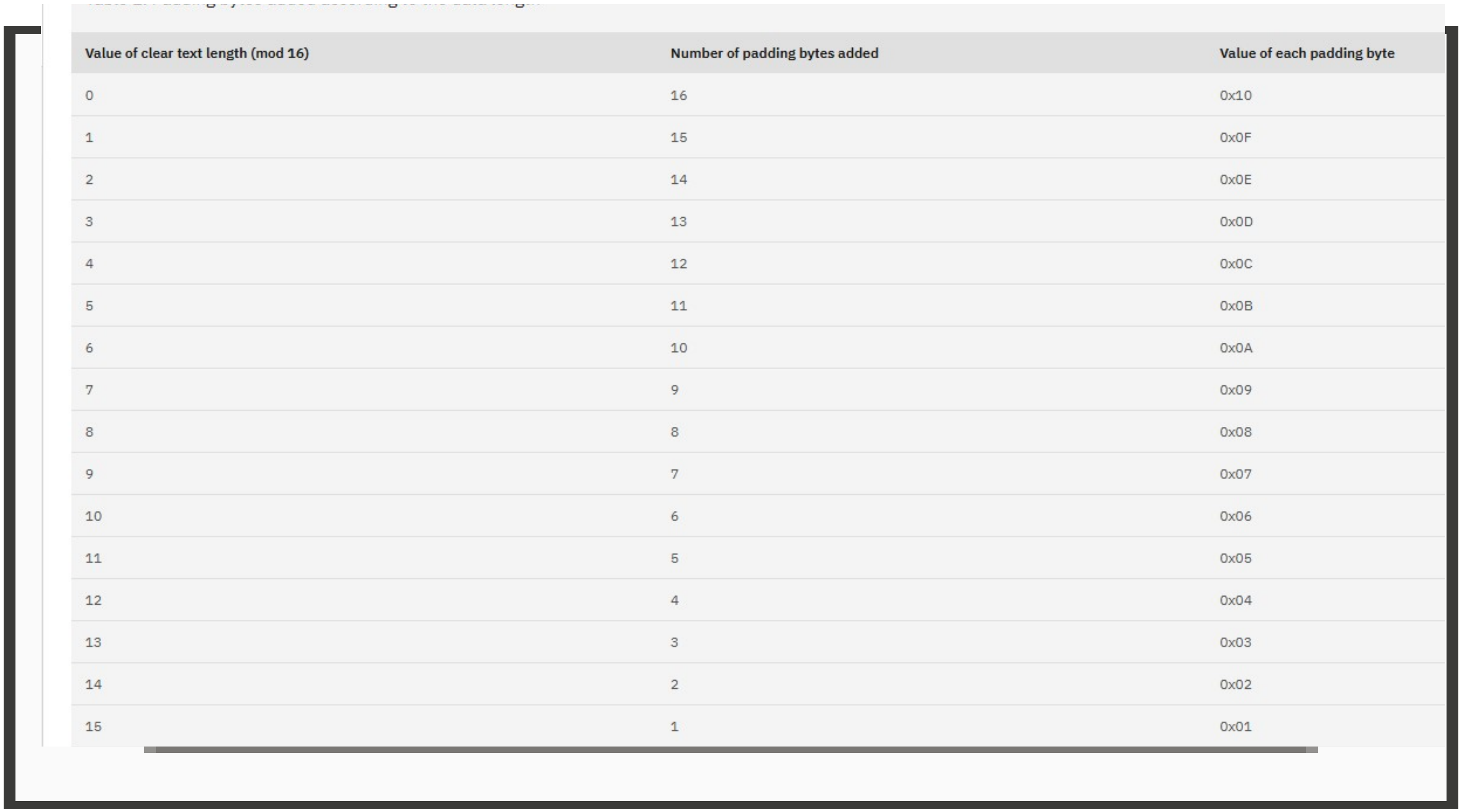
PKCS5 Padding

- Originally designed for block ciphers operating on 64-bit blocks (e.g., DES).
- Up to 8-bytes (64-bit) block sizes.
- The plaintext is padded by adding bytes, each of which is the number of padding bytes added.
  - Example, if 3 bytes of padding are added, the padding will be 0x03 0x03 0x03.
  - if 5 bytes of padding are added, the padding will be 0x05 0x05 0x05 0x05 0x05.

PKCS7 Padding(similar to PKCS5)

- Supports block sizes of up to 255 bytes (maximum number stored in a byte).
- if a message is a multiple of blockSize, PKCS7 still “pads out” with a block of 16 0x10s

PKCS7



Value of clear text length (mod 16)	Number of padding bytes added	Value of each padding byte
0	16	0x10
1	15	0x0F
2	14	0x0E
3	13	0x0D
4	12	0x0C
5	11	0x0B
6	10	0x0A
7	9	0x09
8	8	0x08
9	7	0x07
10	6	0x06
11	5	0x05
12	4	0x04
13	3	0x03
14	2	0x02
15	1	0x01

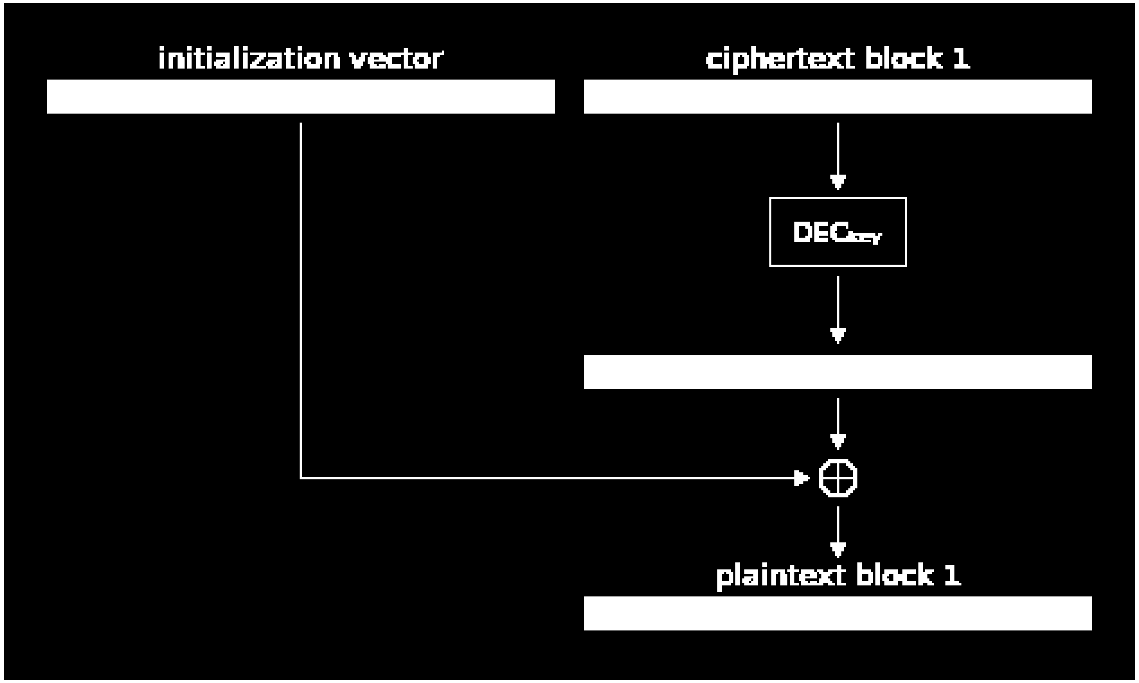
Single block attack

The encrypted message consists of an IV and a single ciphertext block.

- We can set the IV to whatever we want. Initially IV is all zeros
- The padding oracle will compute tells us only whether or not the resulting plaintext block ends with valid padding.
- By making modifications to the IV, we can predictably modify the plaintext block, since flipping a bit in the IV will flip the corresponding bit in the plaintext.
- Setting the IV’s final byte to any value will xor that value into the plaintext’s final byte.
- If we iterate through every possible value for the final IV byte, eventually one of them will set the plaintext’s final byte to 0x01
  - The padding oracle will tell us when this happens, because 0x01 is valid padding!
  - A trailing 0x01 byte meets the padding standard, so the oracle accepts it just like it would accept 0x02 0x02 or 0x03 0x03 0x03.

Padding Attack (1)

- Single block messages

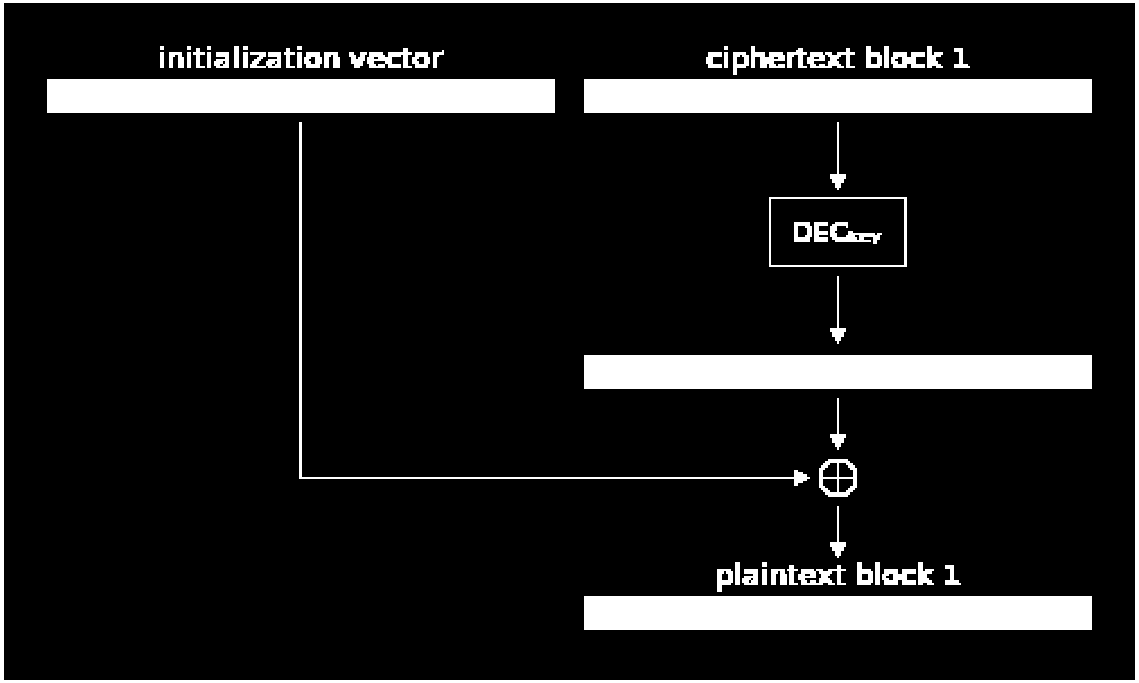


Single block attack

- The search for a valid IV byte ends when we reach 0x2e, because 0x2e ⊕ 0x2f = 0x01.
- We can start to construct what I'll call a zeroing IV, i.e. the IV which will set some (eventually all) of the plaintext's bytes to zero.
  - zero gives us options. If we want to set a plaintext byte to any value other than zero, we can just xor that value into the zeroing IV. In other words, the zeroing IV gives us a way of manipulating the plaintext however we like.
- As soon as we set the plaintext's final byte to 0x01, we can take the corresponding IV byte and xor that against 0x01. This modified IV byte will set the plaintext's final byte to 0x00 – and so it will work as the final byte of our zeroing IV.
- Once we have that, we can derive a new IV which is guaranteed to set the plaintext's final byte to 0x02, and we can start trying to set the plaintext's penultimate byte to 0x02 as well.
- Once we've found valid one-byte padding, we can use a similar process to search for valid two-byte padding.
- This search will terminate when the plaintext's final two bytes equal 0x02 0x02.This permits us to move on to attacking the third-from-last byte, then the fourth-from-last, and so on.
- Note: If the plaintext's penultimate byte is already set to 0x02, then the message's padding would be valid if the final byte is set to either 0x01 or 0x02. If our ultimate byte search hits 0x02 before 0x01, but we assume that we found 0x01 and not 0x02, and the attack fails. We get an affirmative result from the oracle, by changing the IV's penultimate byte and query the oracle again. If both queries succeed, this tells us that the penultimate byte is not part of the message's (valid) padding, proving that the padding has length one and thus must have value 0x01 as well. On the other hand, if this second query fails, we've run into a false positive and should keep searching.

Padding Attack (2)

- Single block messages



Padding Attack

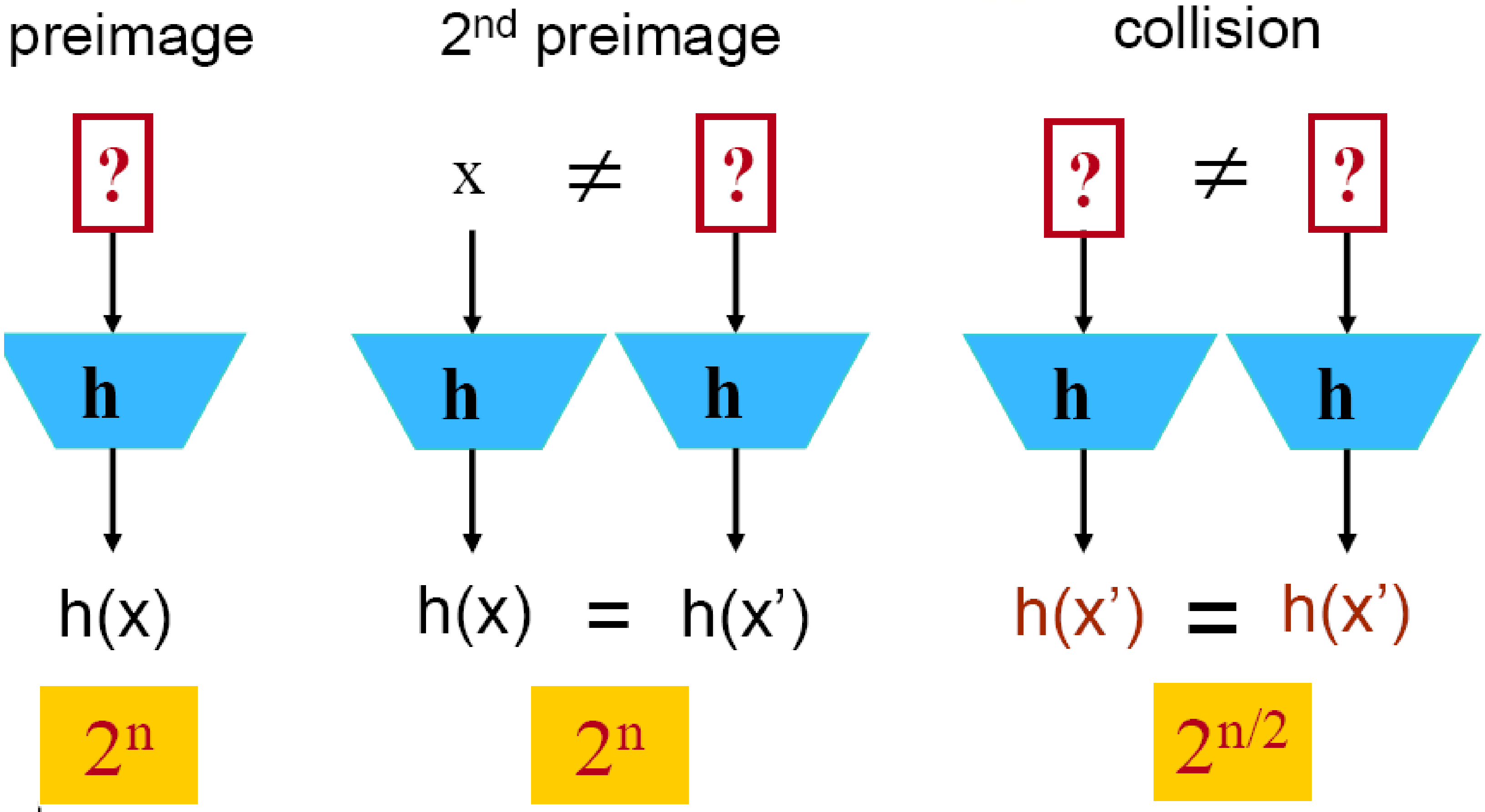
- The attack can be generalized to multi-block plaintexts [1]
- A real world attack against TLS 1.2, IPsec
  - It takes advantage of MAC-then-encrypt
- This attack was first reported against TLS by Serge Vaudenay in 2002 [2].

[1] https://www.nccgroup.com/us/research-blog/cryptopals-exploiting-cbc-padding-oracles/

[2] https://www.iacr.org/archive/eurocrypt2002/23320530/cbc02_e02d.pdf

HASH FUNCTIONS

Cryptographic properties



preimage

?

h

h(x)

2^n

2nd preimage

x ≠ ?

h h

h(x) = h(x')

2^n

collision

? ≠ ?

h h

h(x') = h(x')

$2^{n/2}$

Cryptanalysis

We can do one of the following:

1) Show that there is an attack more efficient than the generic attacks, i.e.
   - ✓ Guess a preimage (complexity 2^n)
   - ✓ Guess a 2nd preimage (complexity 2^n)
   - ✓ Guess a collision pair (complexity $2^{n/2}$)
2) Build on a secure primitive and prove that the construction is secure by showing that if there is an efficient attack against the construction then you can adapt this attack and mount a new one against the secure primitive.

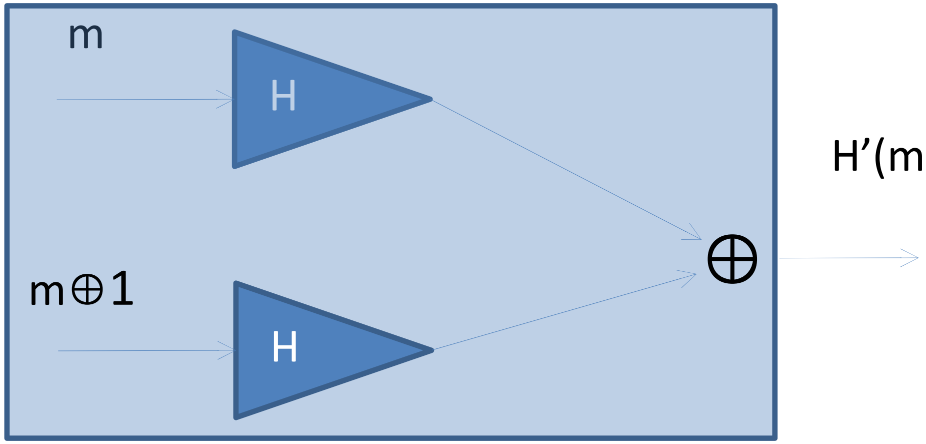
Exercise 1

Let $H:\{0,1\}^* \rightarrow T$ be a collision resistant hash function. Is the following hash function collision resistant?

$$H'(m)=H(m)\oplus H(m\oplus 1^{|m|})$$

where $|m|$ is length of m and 1^x is a string of x 1's.

For instance, let m=10101. Then, $|m|=5$ and $1^{|m|}=1^5=11111$



Exercise 1

Proof

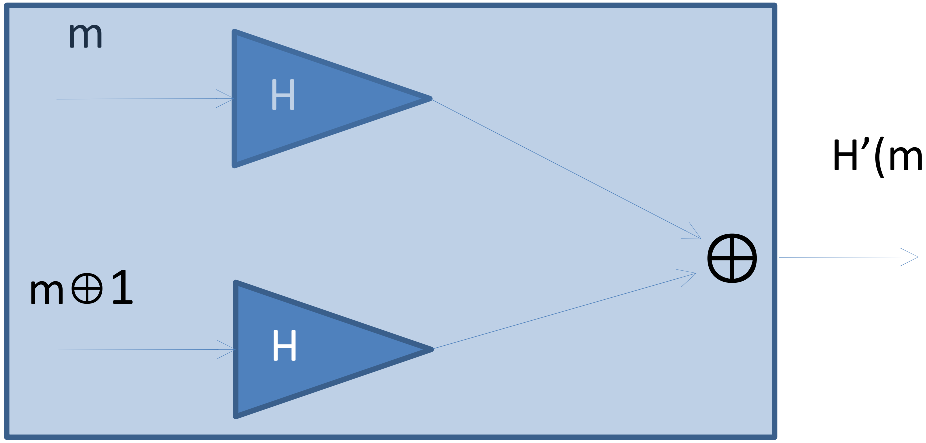
Clearly, it is easy to show that any two message m, m’ such that $m' = m \oplus 1^{|m|}$, they have that same hash value:

$H'(m') = H(m') \oplus H(m' \oplus 1^{|m'|}) = H(m \oplus 1^{|m|}) \oplus H(m \oplus 1^{|m|} \oplus 1^{|m|}) =$

$= H(m \oplus 1^{|m|}) \oplus H(m) = H'(m)$

Thus, it is not collision resistant.

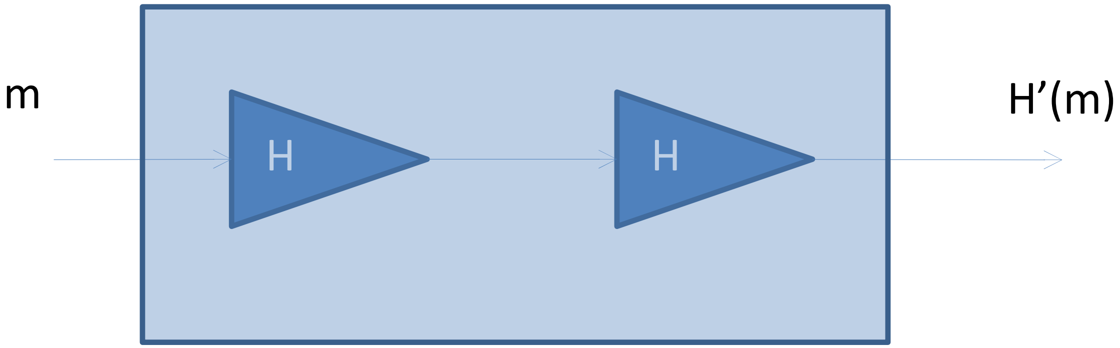
(Try as an example the pair (m = 110001, m’ = 001110)



Exercise 2

1. Let $H:\{0,1\}^* \to \{0,1\}^n$ be a collision resistant hash function. Is the following hash function collision resistant?

$$H'(m)=H(H(m))$$



Proof (sketch)

- Let H’(m)=H(H(m))

and let’s assume that H’(m) is not collision resistant.

Thus, there is a **polynomial algorithm A** that can compute a pair of **different** messages **m1** and **m2**, more efficiently than O($2^{n/2}$), such that:

H’(m1)=H’(m2)

Thus, it holds

H(H(m1))=H(H(m2)).

Proof (sketch)

Thus, it holds

$$H(H(m1))=H(H(m2)).$$

We distinguish two cases:

1. H(m1)=H(m2). Then, the algorithm A can compute collisions for H(m), more efficiently than $O(2^{n/2})$. **This is a contradiction.**

2. H(m1)≠H(m2). Then, the messages y1=H(m1) and y2=H(m2)

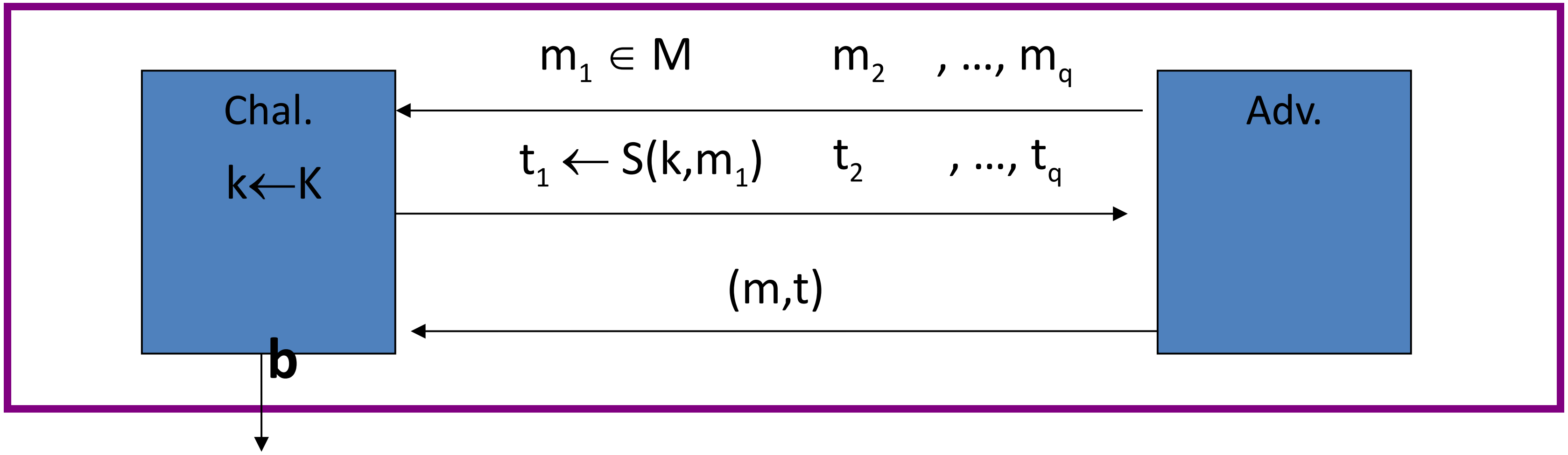
$$H(H(m1))=H(H(m2)) <=> H(y1)=H(y2)$$

are collisions for H(m). That is that, the algorithm A can compute collisions for H(m), more efficiently than $O(2^{n/2})$. **This is a contradiction.**

MAC SECURITY

Strong Unforgeability under Chosen Message Attack (SUF-CMA)

- For a MAC I=(S,V) and adv. A define a MAC game as:



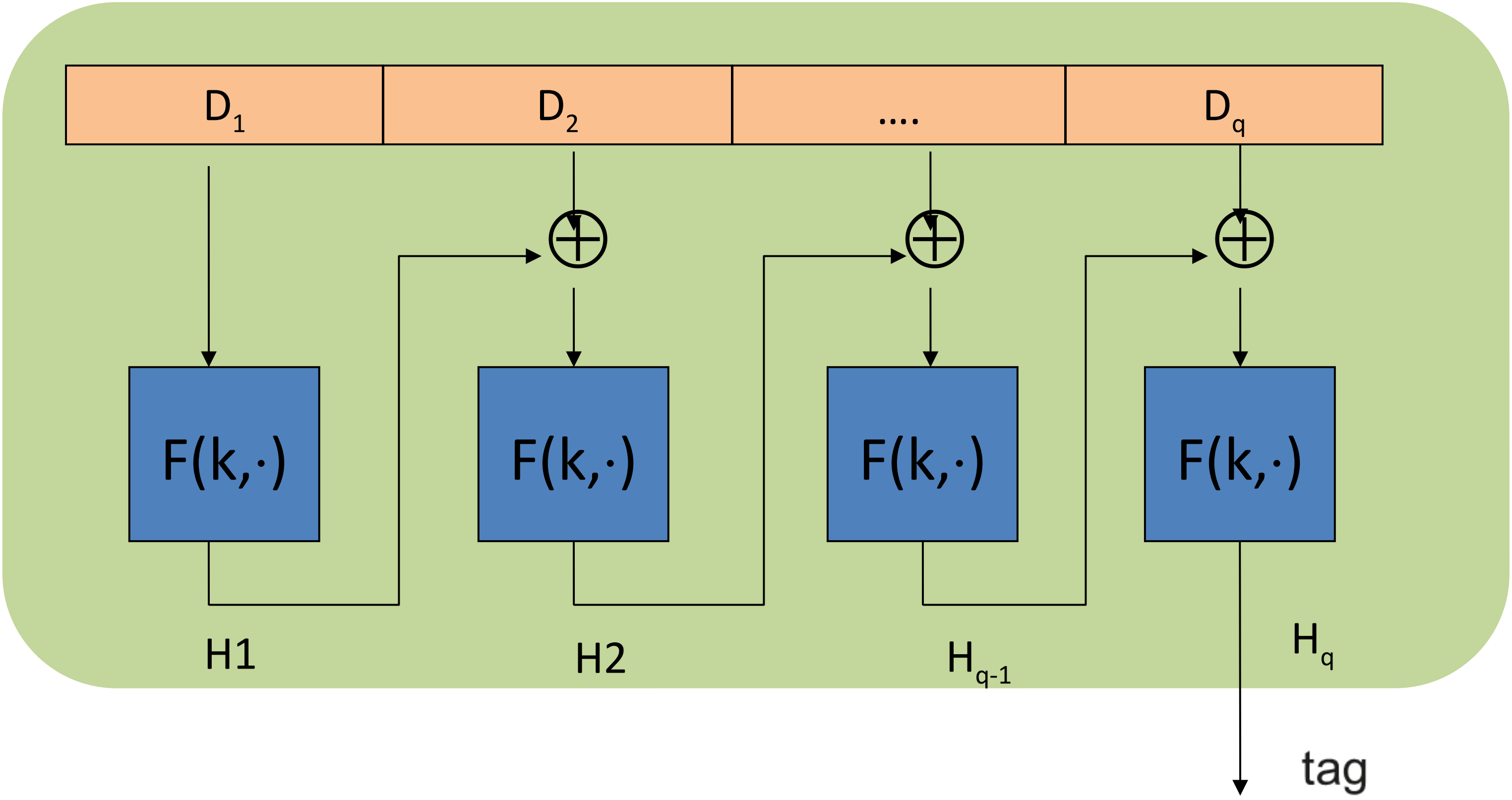
Chal.
$k \leftarrow K$
$m_1 \in M \quad m_2 \quad , \ldots, m_q$
$t_1 \leftarrow S(k,m_1) \quad t_2 \quad , \ldots, t_q$
(m,t)
Adv.
b

$$\begin{cases} \mathbf{b}=1 & \text{if } V(k,m,t) = \text{`yes'} \text{ and } (m,t) \notin \{ (m_1,t_1), \ldots, (m_q,t_q) \} \\ \mathbf{b}=0 & \text{otherwise} \end{cases}$$

Def: I=(S,V) is a **<u>secure MAC</u>** if for all "efficient" A:

$\text{Adv}_{\text{MAC}}[A,I]$ = Pr[Chal. outputs 1] is "negligible."

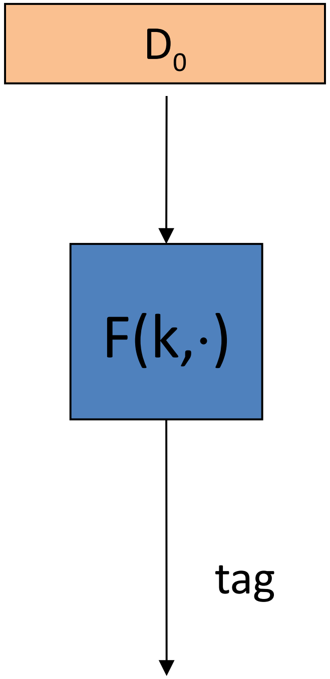
(RAW) CBC-MAC security



(RAW) CBC-MAC security

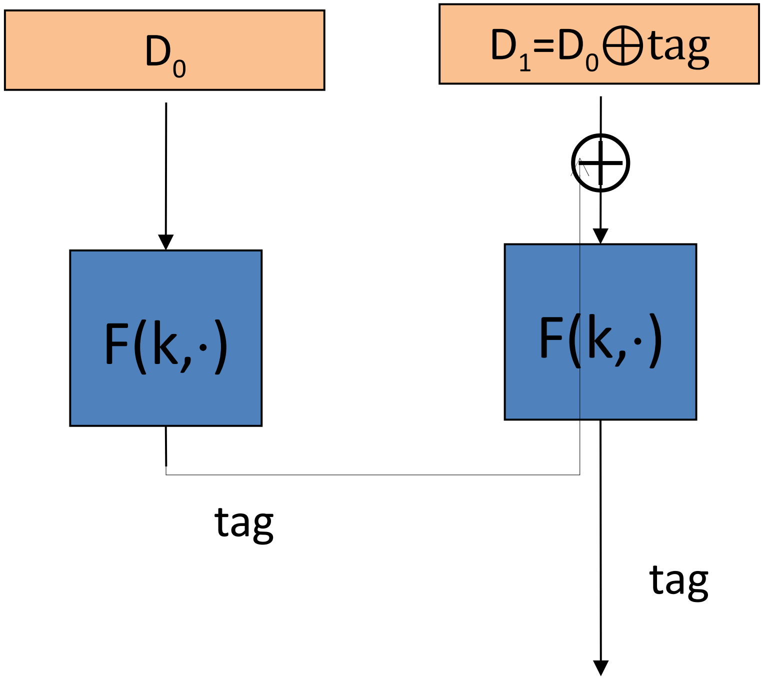
- Let m = D0

(size single block)

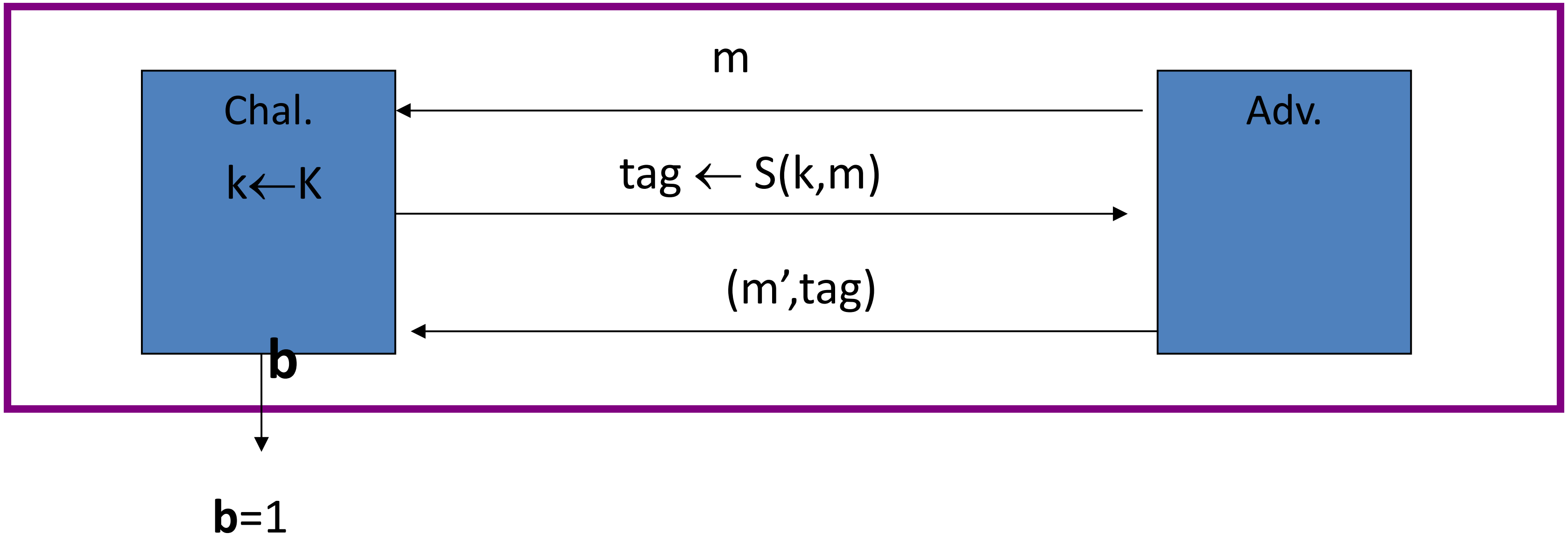


- Let m’ = D0||D0⊕tag

(size two blocks)



Strong Unforgeability under Chosen Message Attack (SUF-CMA)



$$\text{Adv}_{\text{MAC}}[A,I] \;=\; \Pr[\text{Chal. outputs } 1] = 1$$

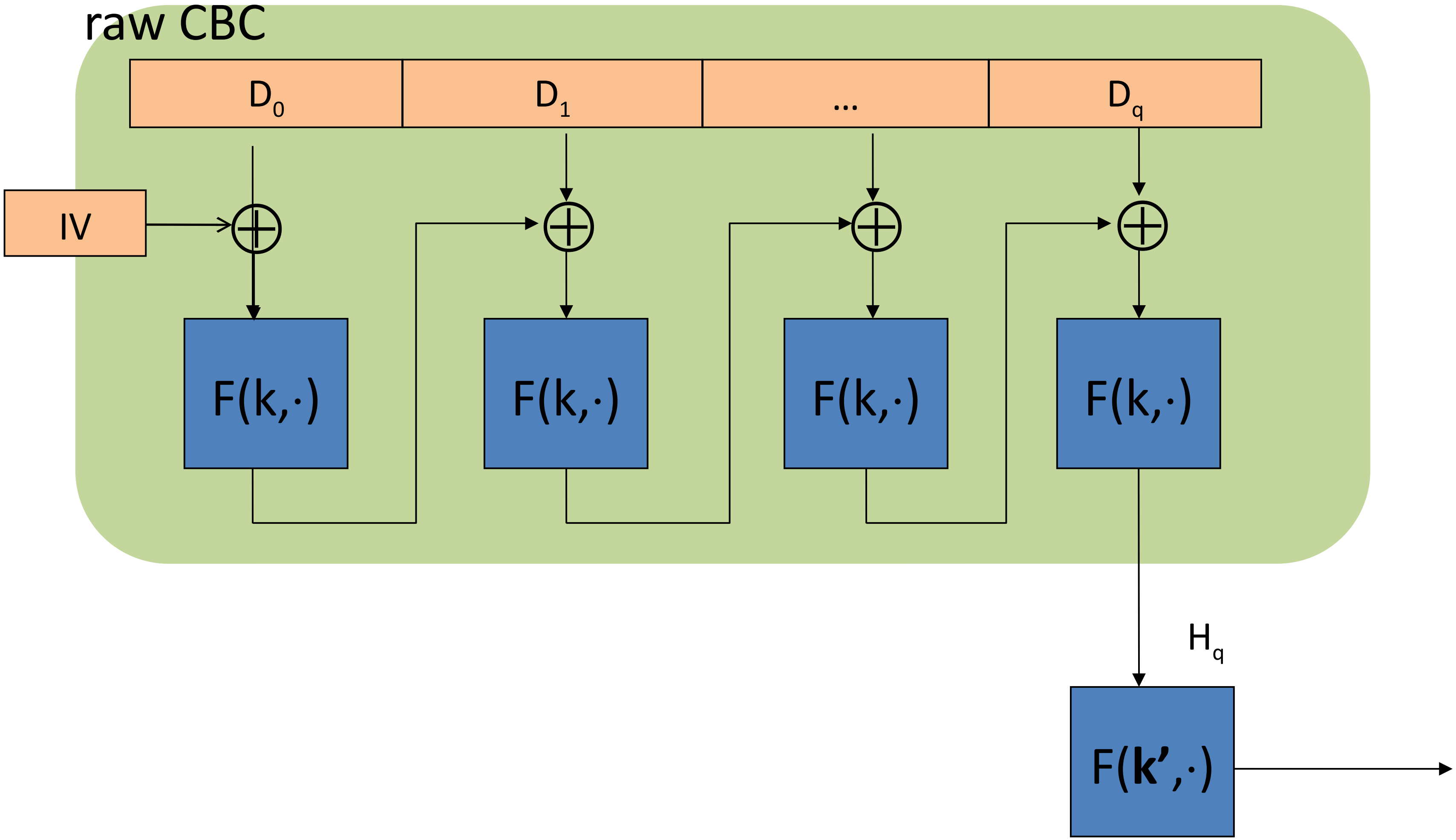
Example

Lets assume that the ECBC_{IV} is a modified version of EMAC which uses a randomly selected **IV** for each message **m** and produces the output as it appears in the next slide. The IV will then be part of the tag, i.e.

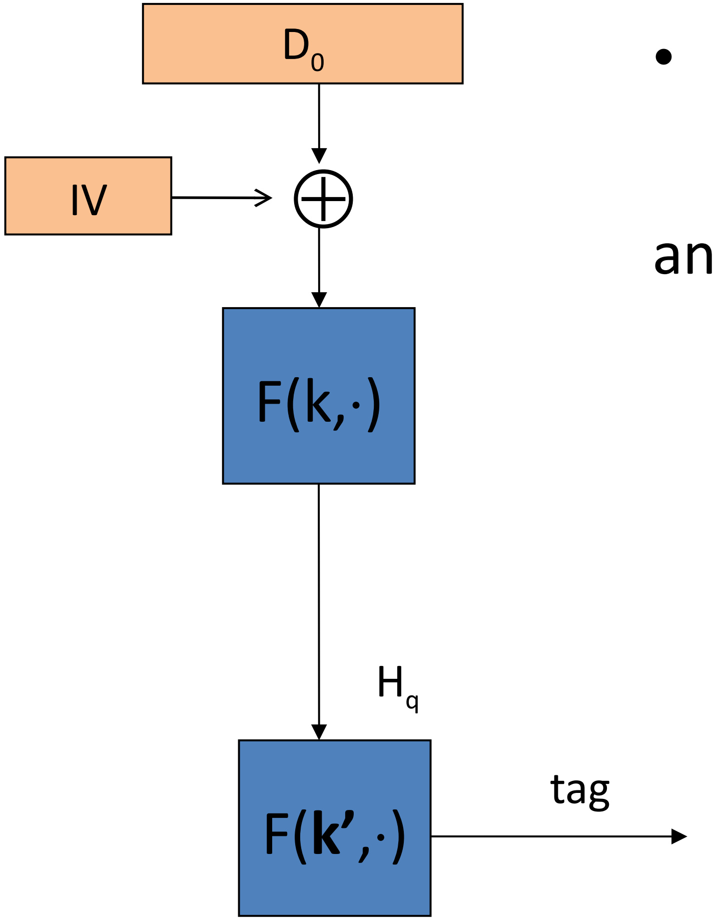
$$\text{tag}=(\text{IV},\ \text{ECBC}_{\text{IV}}(k,k',m))$$

where (k,k') are the two keys. Is the new MAC SUF-CMA secure?

Modified EMAC ($ECBC_{IV}$)



Modified EMAC ($ECBC_{IV}$)

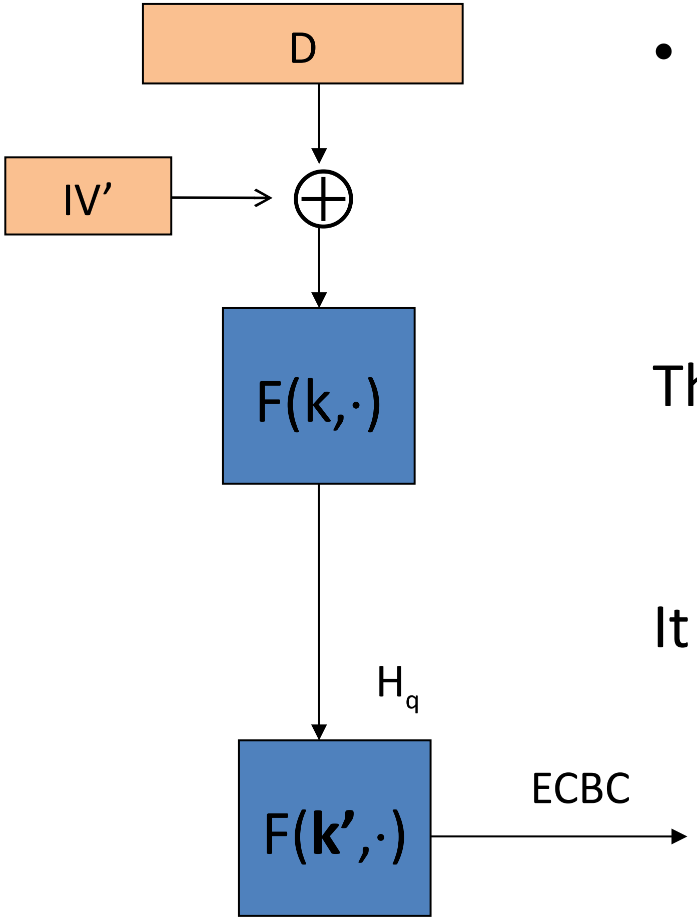


- Let $m_1 = D_0$. Then,

  tag=(IV, $ECBC_{IV}(k,m_1)$)

and IV $\oplus D_0$ is the input of the cipher F

Modified EMAC ($ECBC_{IV}$)



- It is easy to verify that for any message (for any D)
  - m_2= D, and
  - $IV' = IV \oplus D \oplus D_0$

The input to F is again

$$IV' \oplus D = IV \oplus D \oplus D_0 \oplus D = IV \oplus D_0$$

It holds that

$$tag=(IV', \ ECBC_{IV'}(k,m_2))$$

ECBC-MAC and HMAC analysis

Theorem: Let q be the number of different messages for which a tag was produced (using the same key) and let L be the total length in bits. If Adv[B,F] be the advantage of an efficient attacker B against a block cipher F and Adv[B,H] be the advantage of an efficient attacker B against a hash function H. Then:

$$\mathrm{Adv}_{PRF}[A, F_{ECBC}] \leq \mathrm{Adv}_{PRP}[B, F] + 2\, q^2 / |X|$$

$$\mathrm{Adv}_{PRF}[A, F_{HMAC}] \leq q \cdot L \cdot \mathrm{Adv}_{PRF}[B, F] + q^2 / 2|K|$$

Where |X| is the total number of different input blocks for the block cipher and |K| the total number of keys

ECBC-MAC and HMAC analysis

1. When the block cipher is secure we Adv[B,F]=0 and

$$\mathrm{Adv}_{\mathrm{PRF}}[A, F_{\mathrm{ECBC}}] \leq 2\, q^2 / |X|$$

Thus, CBC-based MAC is secure as long as $q \ll |X|^{1/2}$

2. When the block cipher is secure we Adv[B,H]=0 and

$$\mathrm{Adv}_{\mathrm{PRF}}[A, F_{\mathrm{HMAC}}] \leq q^2 / 2|K|$$

Thus, HMAC is secure as long as $q \ll |K|^{1/2}$

An example

$\text{Adv}_{\text{PRF}}[A, F_{\text{ECBC}}] \leq \text{Adv}_{\text{PRP}}[B, F] + \mathbf{2\, q^2 / |X|}$

q = # messages MAC-ed with k

Suppose we want $\text{Adv}_{\text{PRF}}[A, F_{\text{ECBC}}] \leq 1/2^{32}$ $\Leftarrow$ $q^2 / |X| < 1/ 2^{32}$

- AES: $|X| = 2^{128}$ $\Rightarrow$ $q < 2^{48}$

  So, after 2^{48} messages we must change key

- 3DES: $|X| = 2^{64}$ $\Rightarrow$ $q < 2^{16}$

  So, after 2^{16} messages we must change key

WEP - SECURITY

Wired Equivalent Privacy (WEP)

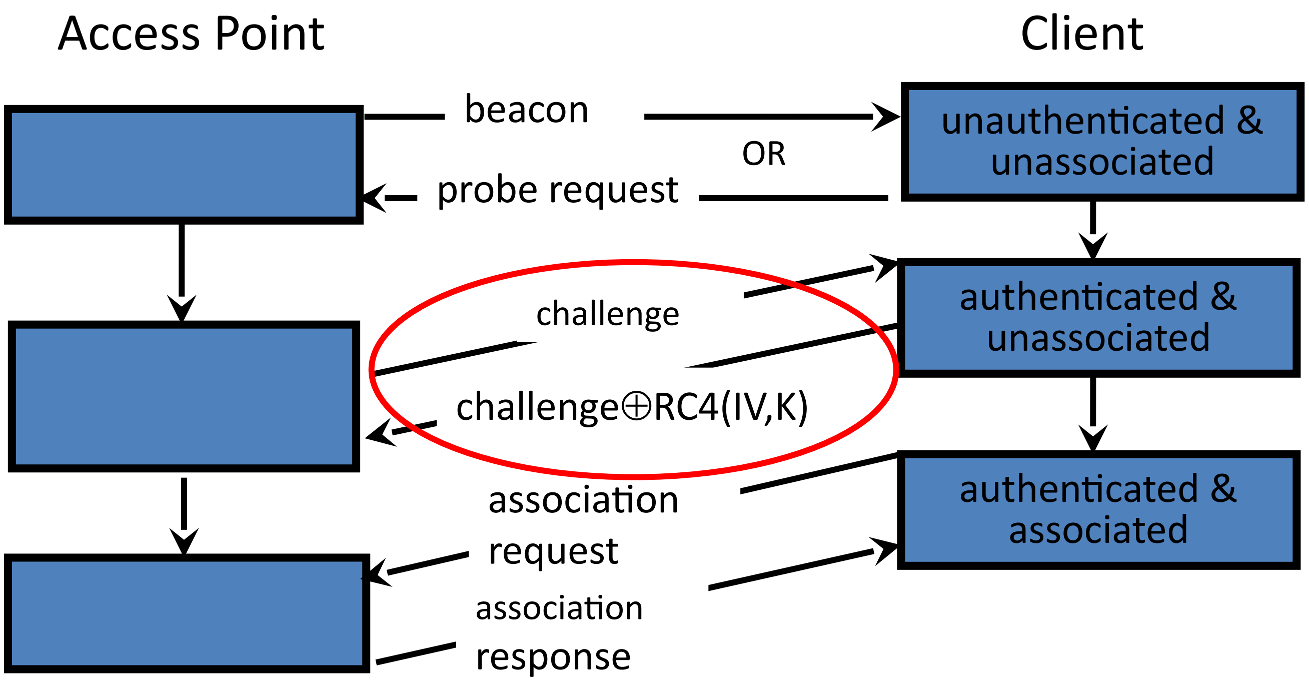
- WEP - Part of original 802.11 specification published in 1999.
- Confidentiality
  - Uses RC4 Stream cipher
  - Has static 40-bit base key (common for all the clients)
  - A 64-bit per-packet key
  - A 24-bit Initialization Vector (IV)
- Integrity
  - Uses Integrity Check Value (ICV) to verify integrity
  - No key!!

Characteristics - notes

- Stateless protocol
  - Mobile stations and access points are not required to keep past state
- Encrypted CRC-32 used as integrity check
  - Fine for random errors, but not deliberate ones
  - Linear
    - CRC(X+Y) = CRC(X)+CRC(Y)
- RC4 keystream should not be reused
  - One-time pad

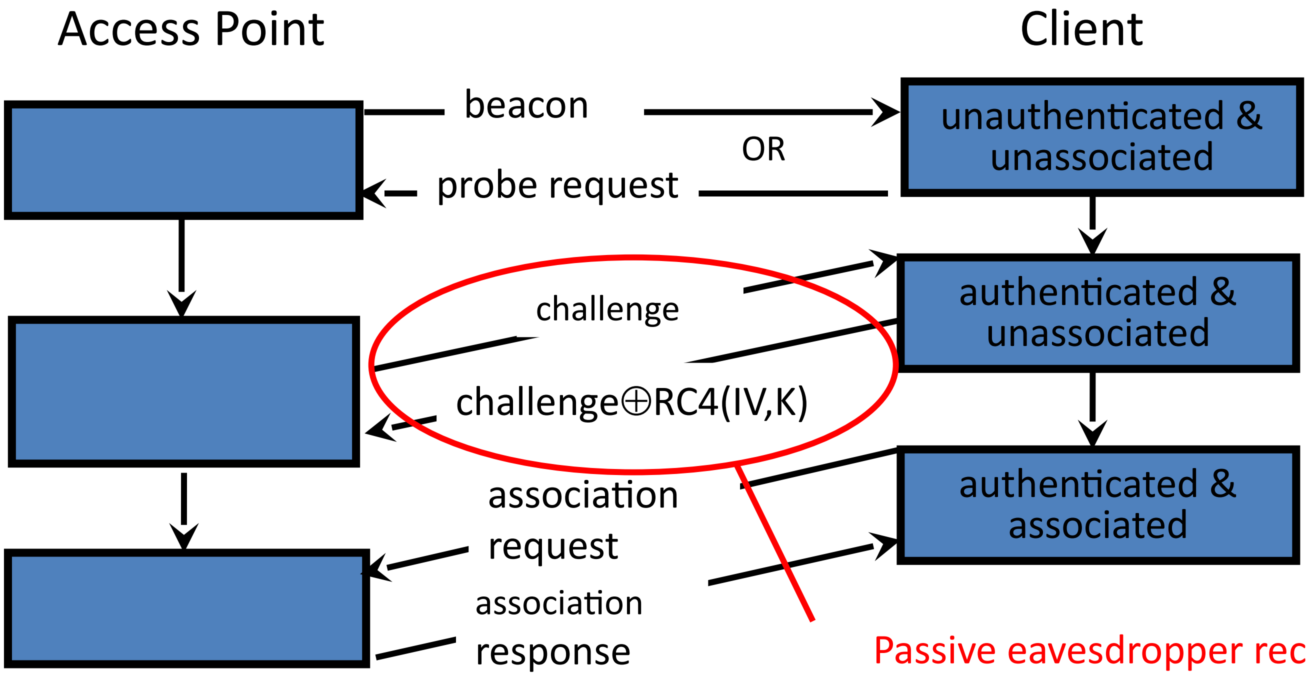
Shared-Key Authentication

Prior to communicating data, access point may require client to authenticate



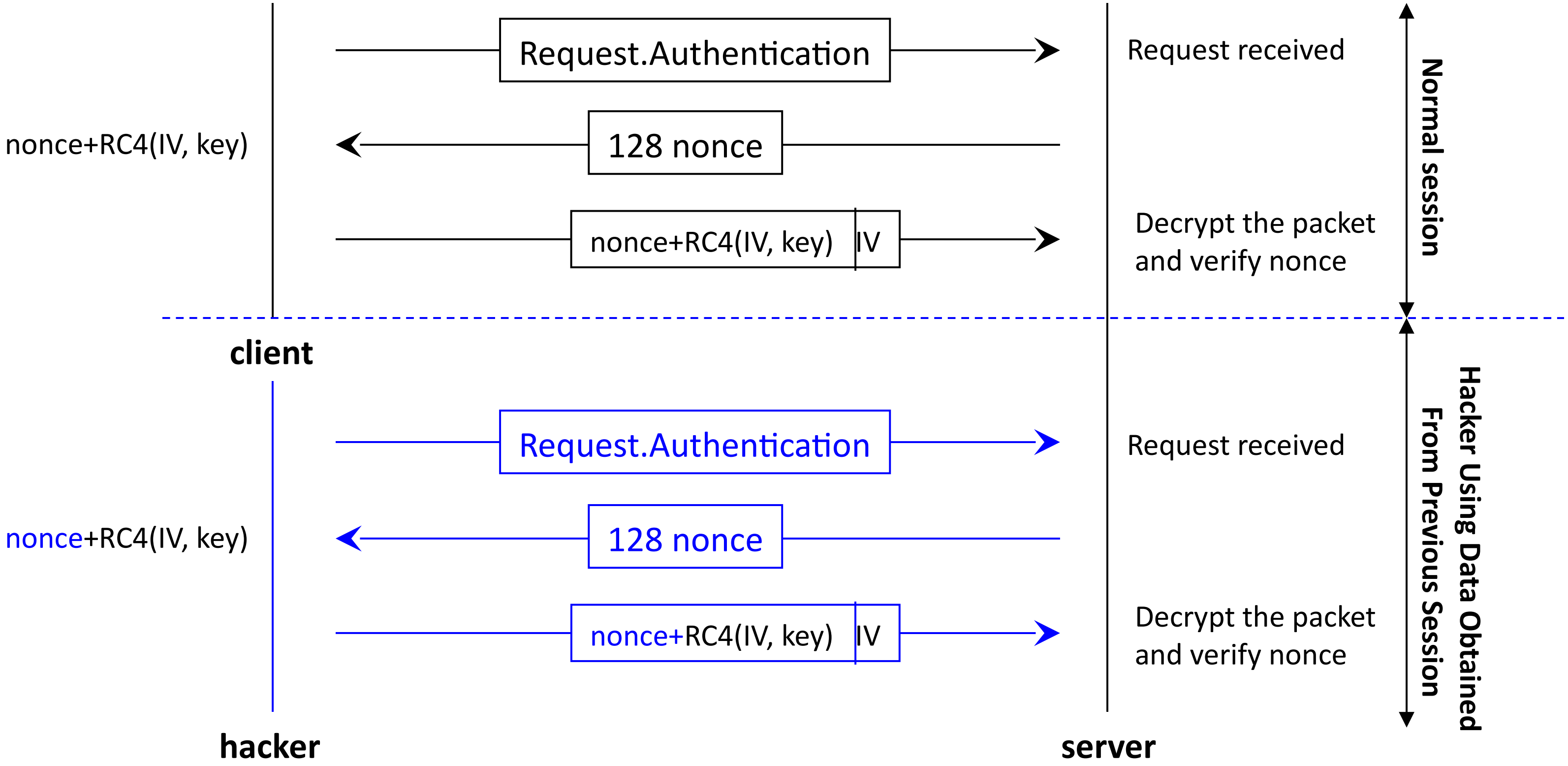
Shared-Key Authentication

Prior to communicating data, access point may require client to authenticate



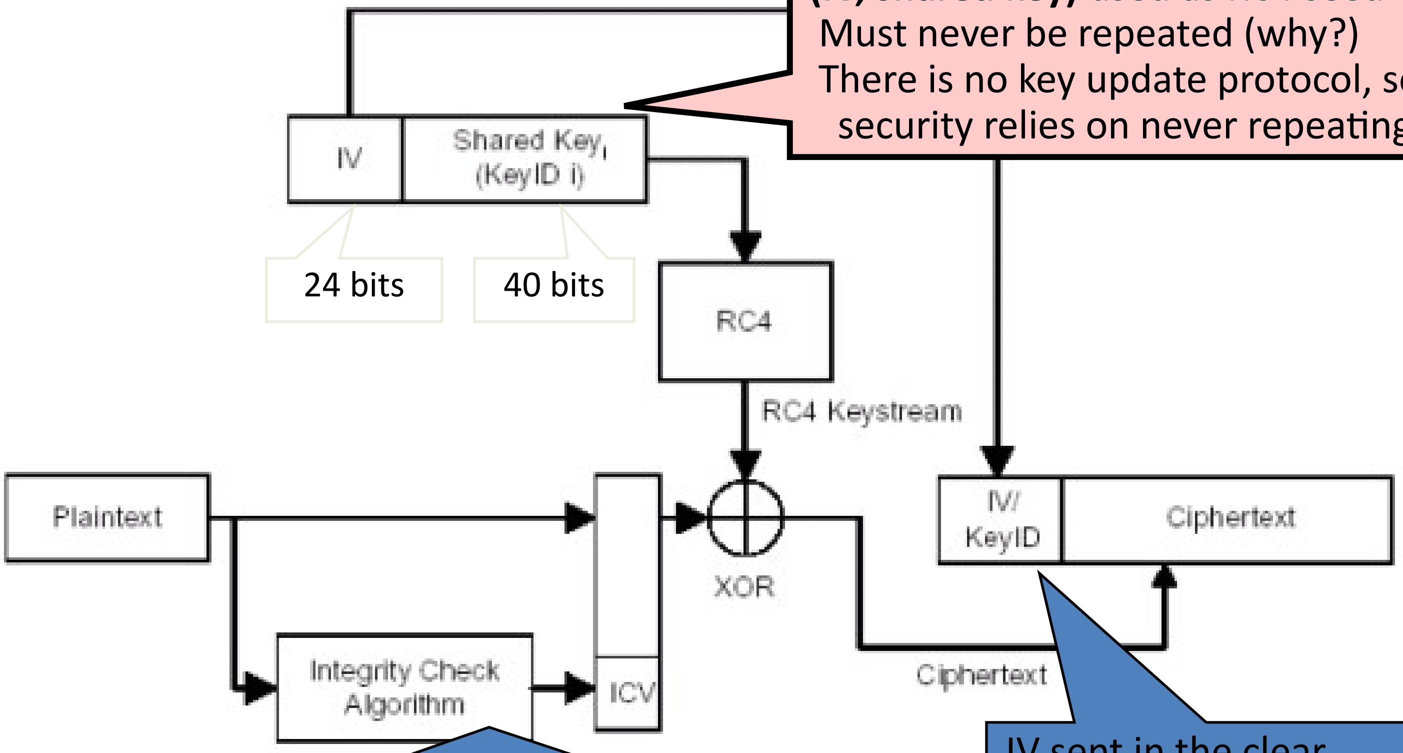
Passive eavesdropper recovers RC4(IV,K), can respond to any subsequent challenge without knowing K

Attack on Access Control



- It is possible to get authenticated without knowing the secret key! (shown in blue)
- We only need a plaintext, ciphertext pair of a legitimate authentication. (shown in black)

How WEP “Privacy” Works



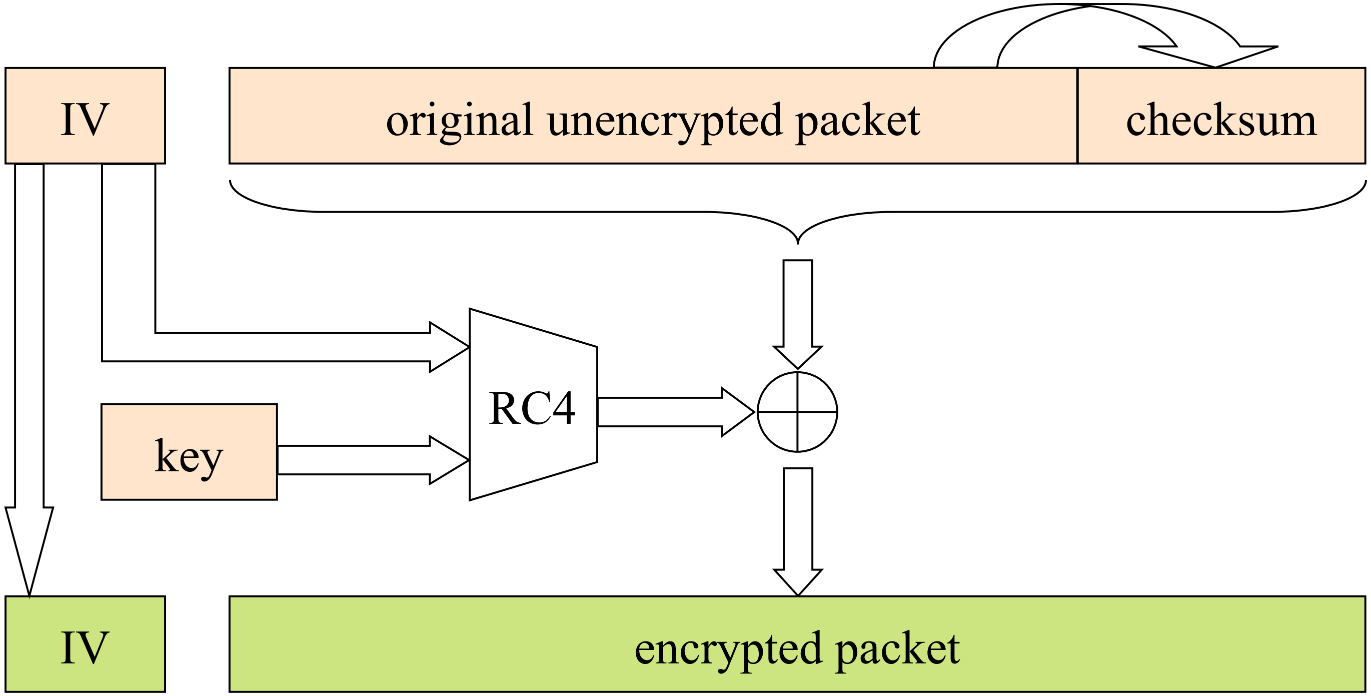
(IV, shared key) used as RC4 seed
Must never be repeated (why?)
There is no key update protocol, so
security relies on never repeating IV

IV sent in the clear
Worse: <u>changing IV with each packet is optional!</u>

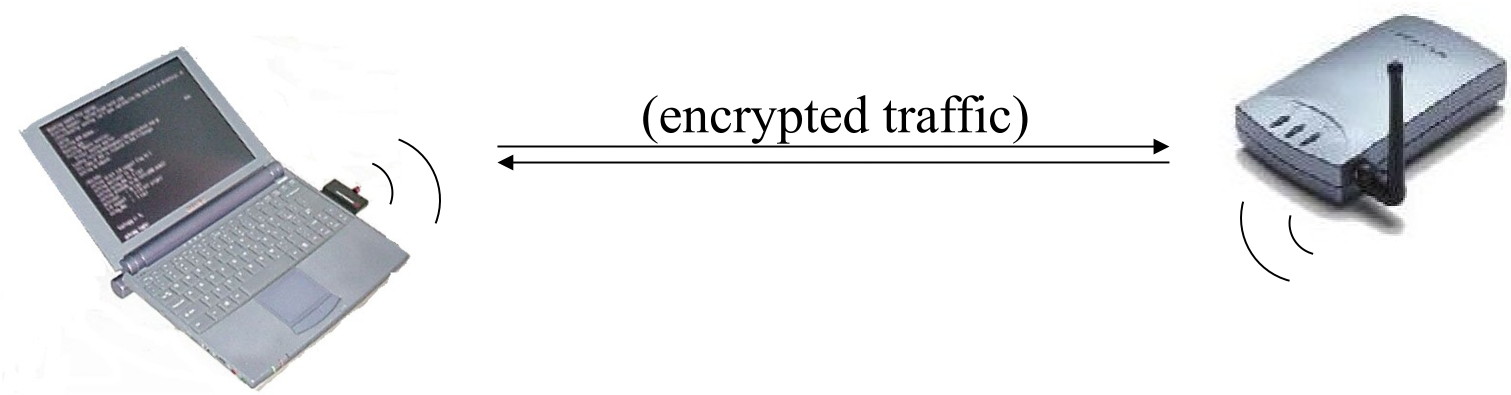
CRC-32 checksum is linear in ⊕:
if attacker flips some plaintext bits, he knows which bits of CRC to flip to produce the <u>same checksum</u>

no integrity!

WEP

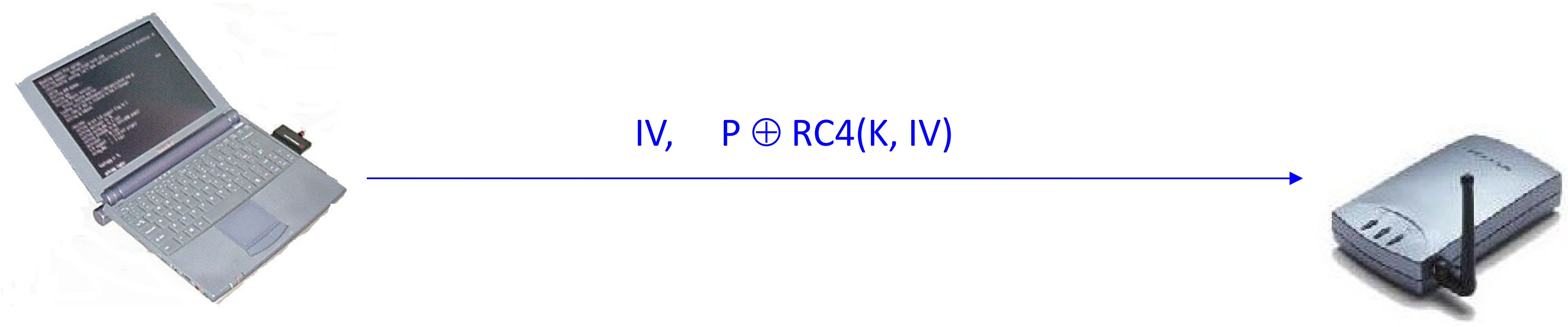

IV
original unencrypted packet
checksum
RC4
key
IV
encrypted packet

WEP



- Share a single cryptographic key among all devices
- Encrypt all packets sent over the air, using the shared key
- Use a checksum to prevent injection of spoofed packets

WEP - A Little More Detail



IV, P ⊕ RC4(K, IV)

- WEP uses the RC4 stream cipher to encrypt a TCP/IP packet (P) by xor-ing it with keystream (RC4(K, IV))

Attack #1: Keystream Reuse

- WEP didn't use RC4 carefully
- The problem: IV's frequently repeat
  - The IV is often a counter that starts at zero
  - Hence, rebooting causes IV reuse
  - Also, there are only 16 million possible IV's, so after intercepting enough packets, there are sure to be repeats
- Attackers can eavesdrop on 802.11 traffic
  - An eavesdropper can decrypt intercepted ciphertexts even without knowing the key

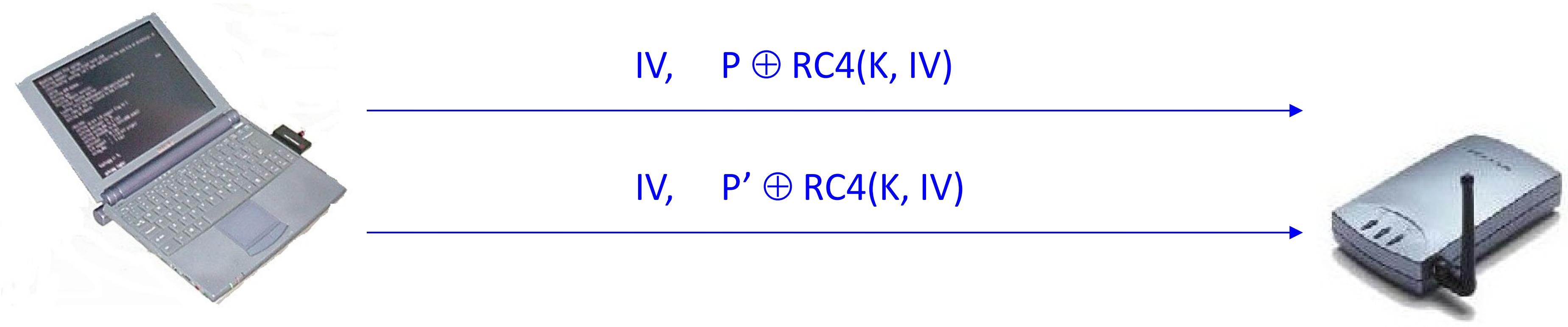
Attack #2: Spoofed Packets

- Attackers can inject forged 802.11 traffic
  - Learn RC4(K, IV) using previous attack
  - Since the checksum is unkeyed, you can then create valid ciphertexts that will be accepted by the receiver

A Property of RC4

- Keystream leaks, under known-plaintext attack
  - Suppose we intercept a ciphertext C, and suppose we can guess the corresponding plaintext P
  - Let PRS = RC4(K, IV) be the RC4 keystream
  - Since $C = P \oplus PRS$, we can derive the RC4 keystream PRS by $P \oplus C = P \oplus (P \oplus PRS) = PRS$
- This is not a problem ... unless keystream is reused!

A Risk of Keystream Reuse



IV, $P \oplus \mathrm{RC4}(K, IV)$

IV, $P' \oplus \mathrm{RC4}(K, IV)$

- If IV’s repeat, confidentiality is at risk
  - If we send two ciphertexts (C, C') using the same IV, then the xor of plaintexts leaks ($P \oplus P' = C \oplus C'$), which might reveal both plaintexts

➤ Lesson: If RC4 isn’t used carefully, it becomes insecure

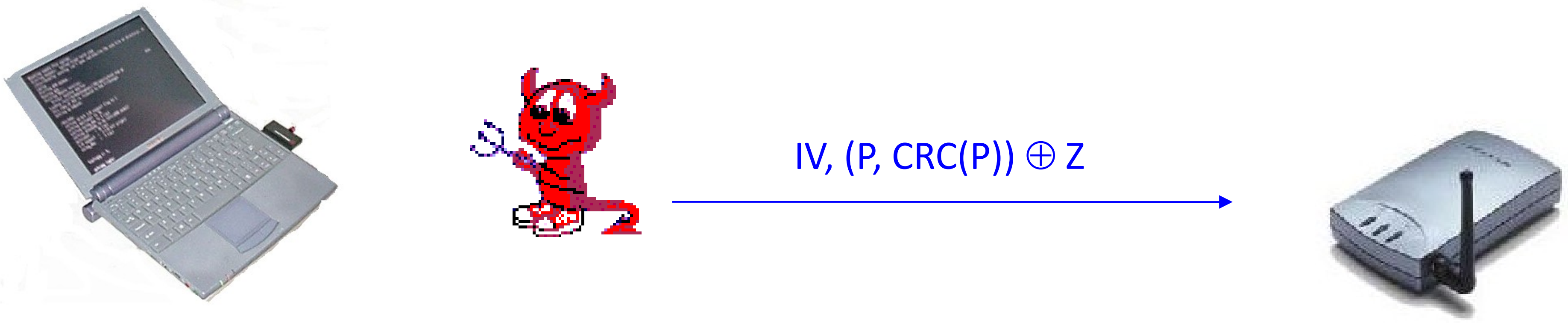
A note on IVs

- What if random IVs were used?
- IV space – 2^{24} possibilities
- Collision after 4000 packets
- Rough estimate: a busy AP sends 1000 packets/sec
- Collision every 4s!
- Even with counting IV (best case), rollover every few hours

So..

- If we have 2^{24} known plaintexts, can decrypt every packet!!!!

Attack #2: Spoofed Packets

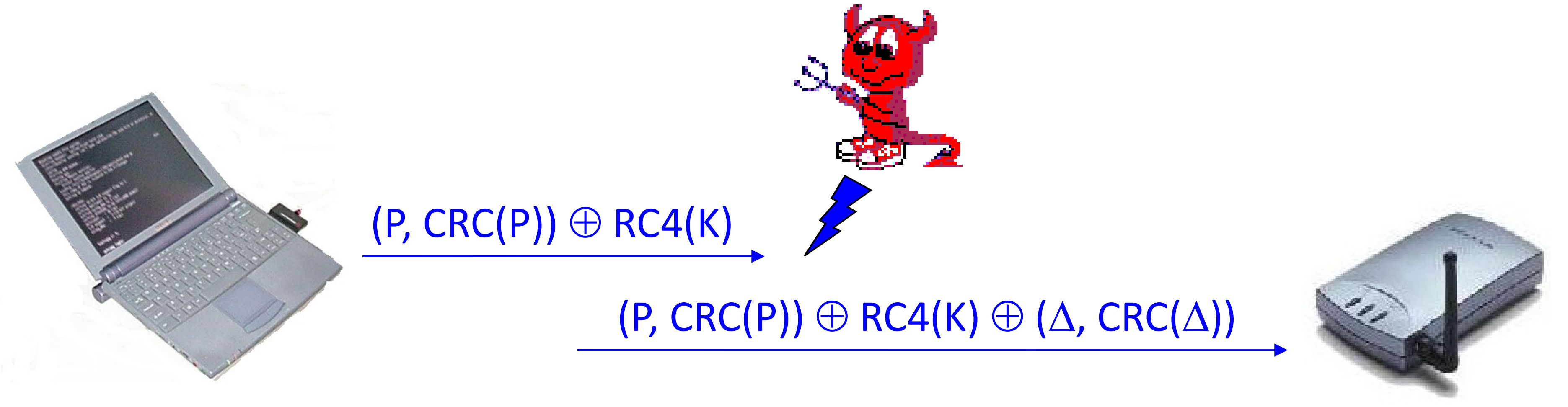


$\mathrm{IV}, (\mathrm{P}, \mathrm{CRC}(\mathrm{P})) \oplus \mathrm{Z}$

- Attackers can inject forged 802.11 traffic
  - Learn $\mathrm{Z} = \mathrm{RC4}(\mathrm{K}, \mathrm{IV})$ using previous attack
  - Since the CRC checksum is unkeyed, you can then create valid ciphertexts that will be accepted by the receiver

- Attackers can bypass 802.11b access control
  - All computers attached to wireless net are exposed

Attack #3: Packet Modification



$(P, \mathrm{CRC}(P)) \oplus \mathrm{RC4}(K)$

$(P, \mathrm{CRC}(P)) \oplus \mathrm{RC4}(K) \oplus (\Delta, \mathrm{CRC}(\Delta))$

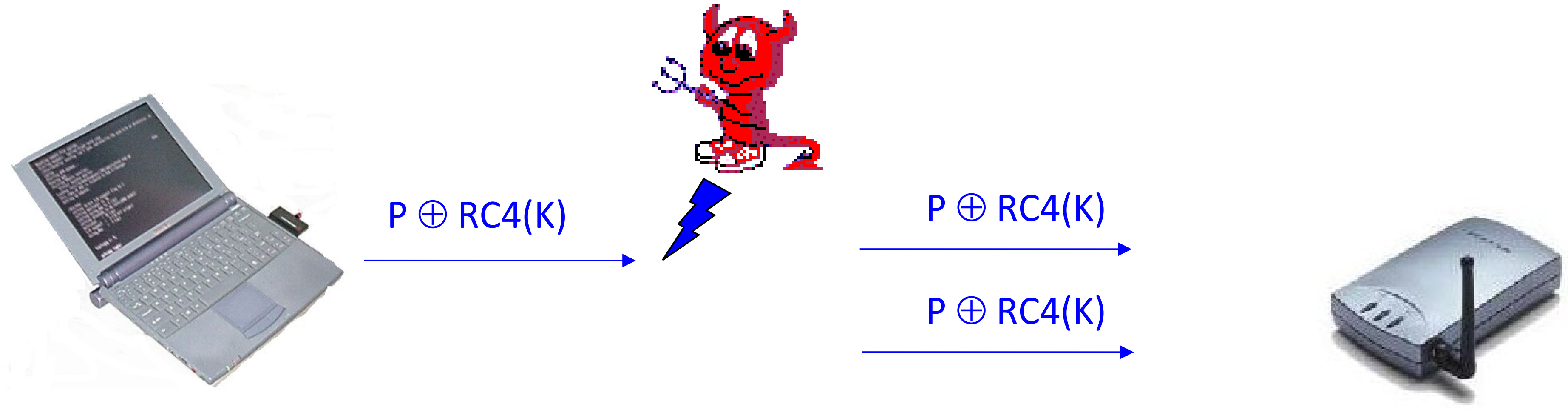
- CRC is linear

  ⇒ $\mathrm{CRC}(P \oplus \Delta) = \mathrm{CRC}(P) \oplus \mathrm{CRC}(\Delta)$

  ⇒ the modified packet ($P \oplus \Delta$) has a valid checksum

➤ Attacker can tamper with packet (P) without breaking RC4

Attack #4: Replay Attacks



P ⊕ RC4(K)

P ⊕ RC4(K)

P ⊕ RC4(K)

- Attacker can replay plaintext (P) without breaking RC4
- Stateless!!

Attacks

- Andrea Bittau, Mark Handley, and Joshua Lackey. The final nail in WEP's cofin. In IEEE Symposium on Security and Privacy, pages 386-400. IEEE Computer Society, 2006.
- 2. Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: the insecurity of 802.11. In ACM MobiCom 2001, pages 180-189. ACM Press, 2001.
- 3. Rafik Chaabouni. Break WEP faster with statistical analysis. Technical report, EPFL, LASEC, June 2006. http://lasecwww.epfl.ch/pub/lasec/doc/cha06.pdf.
- Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of RC4. In Serge Vaudenay and Amr M. Youssef, editors, Selected Areas in Cryptography 2001, volume 2259 of Lecture Notes in Computer Science, pages 1-24. Springer, 2001.
- Andreas Klein. Attacks on the RC4 stream cipher. submitted to Designs, Codes and Cryptography, 2007.
- KoreK. chopchop (experimental WEP attacks). http://www.netstumbler.org/showthread.php?t=12489, 2004.
- KoreK. Next generation of WEP attacks? http://www.netstumbler.org/showpost.php?p=93942&postcount=35, 2004.
- Subhamoy Maitra and Goutam Paul. Many keystream bytes of RC4 leak secret key information. Cryptology ePrint Archive, Report 2007/261, 2007. http://eprint.iacr.org/.
- Toshihiro Ohigashi, Hidenori Kuwakado, and Masakatu Morii. A key recovery attack on WEP with less packets. to be published, 2007.
- Yuko Ozasa, Yoshiaki Fujikawa, Toshihiro Ohigashi, Hidenori Kuwakado, and Masakatu Morii. A study on the Tews, Weinmann, Pyshkin attack against WEP. In IEICE Tech. Rep., volume 107 of ISEC2007-47, pages 17{21, Hokkaido, July 2007. Thu, Jul 19, 2007 - Fri, Jul 20 : Future University-Hakodate (ISEC, SITE, IPSJ-CSEC).
- Adam Stubblefield, John Ioannidis, and Aviel D. Rubin. A key recovery attack on the 802.11b wired equivalent privacy protocol (WEP). ACM Transactions on Information and System Security, 7(2):319{332, May 2004.
- Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin, ‘Breaking 104 bit WEP in less than 60 seconds’, Cryptology ePrint Archive, Report 2007/120, 2007. http://eprint.iacr.org/.

Questions?