CSE5014 CRYPTOGRAPHY AND NETWORK SECURITY

Dr. Qi Wang

Department of Computer Science and Engineering
Office: Room413, CoE South Tower
Email: wangqi@sustech.edu.cn

Stream ciphers

- As we defined, PRGs are *limited*
  - They have fixed-length output
  - They produce output in "one shot"

- In practice, PRGs are based on *stream ciphers*
  - Can be viewed as producing an "infinite" stream of pseudorandom bits, on demand
  - More flexible, more efficient

Stream ciphers

- Pair of efficient, deterministic algorithms (Init, GetBits)

Stream ciphers

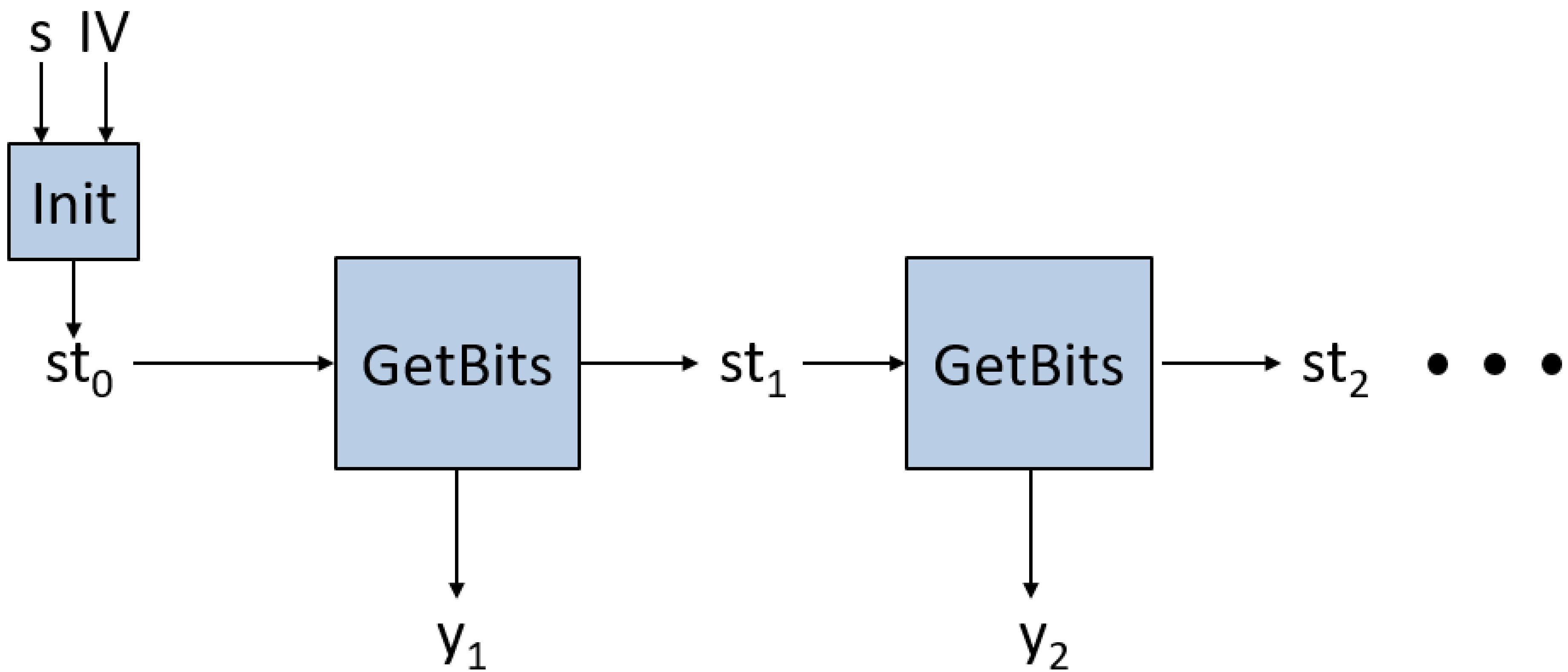
- Pair of efficient, deterministic algorithms (Init, GetBits)
  - Init takes a seed s_0 (and optional IV), and outputs initial state st_0
  - GetBits takes the current state st and outputs a bit y along with updated state st'

Stream ciphers

- Pair of efficient, deterministic algorithms (Init, GetBits)
  - Init takes a seed s_0 (and optional IV),
    and outputs initial state st_0
  - GetBits takes the current state st and
    outputs a bit y along with updated state st'
    - In practice, y would be a block rather than a bit

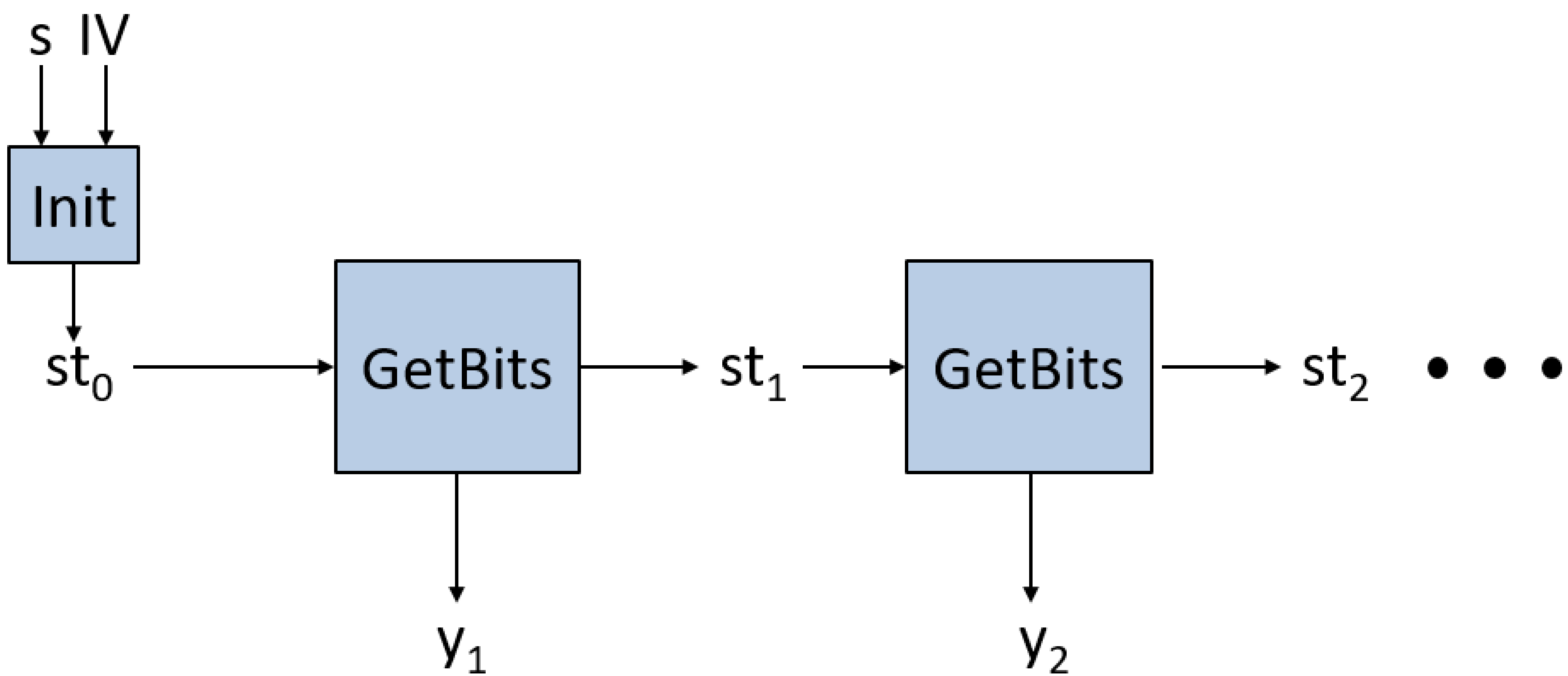
Stream ciphers

- Can use (Init, GetBits) to generate any desired number of output bits from an initial seed



Stream ciphers

- A *stream cipher* is *secure* (informally) if the output stream generated from a uniform seed is pseudorandom
  - I.e., regardless of how long the output stream is (so long as it is polynomial)



Modes of operation

- Stream-cipher modes of operation
  - Synchronized
  - Unsynchronized

Modes of operation

- Stream-cipher modes of operation
  - Synchronized
  - Unsynchronized

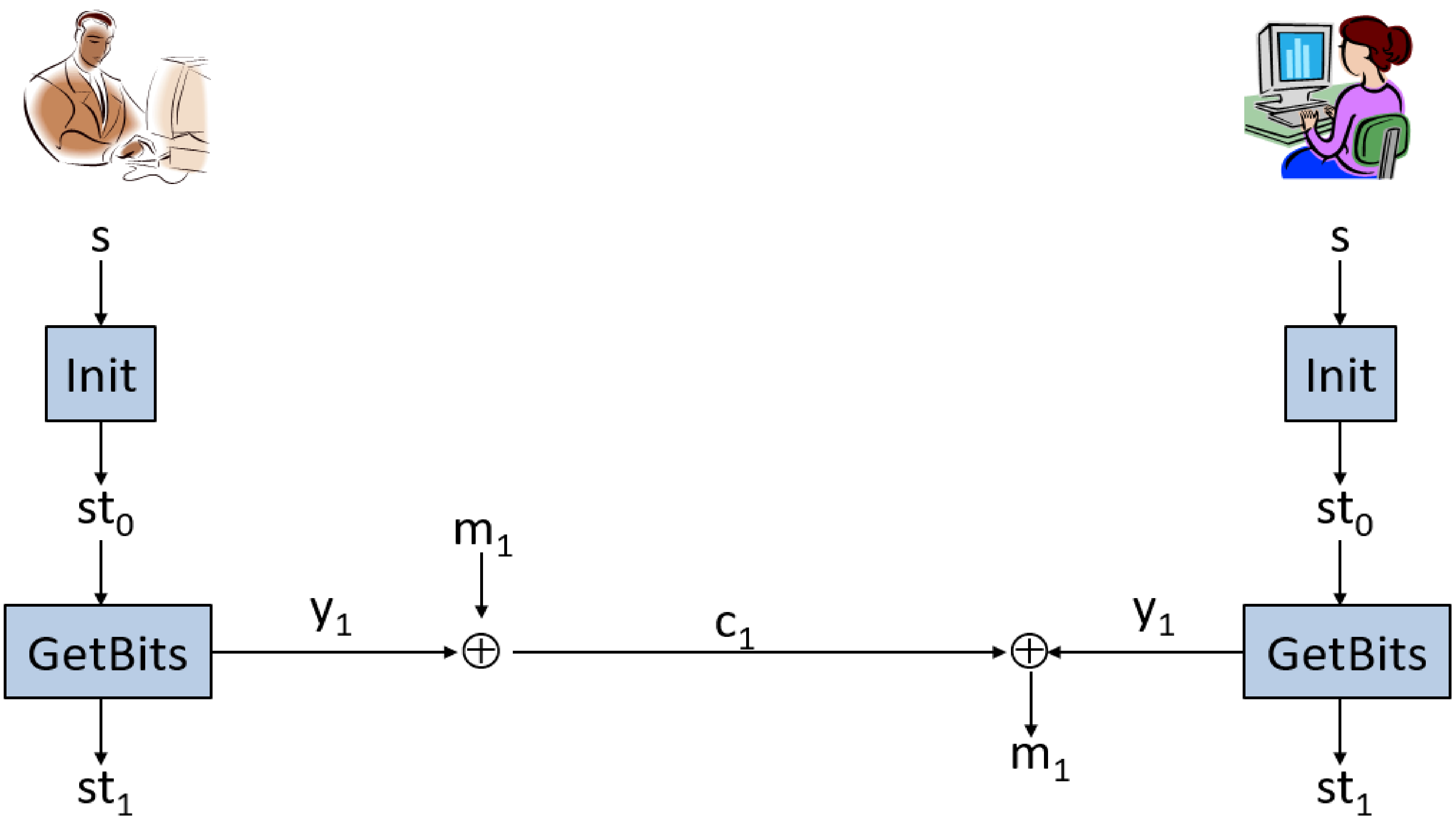
- Synchronized mode
  - Sender and receiver maintain state (they are stateful), and must be *synchronized*

Modes of operation

- Stream-cipher modes of operation
  - Synchronized
  - Unsynchronized

- Synchronized mode
  - Sender and receiver maintain state (they are stateful), and must be *synchronized*
  - Makes sense in the context of a limited-time communication session where messages are received in order, without being lost

Synchronized mode



Unsynchronized mode

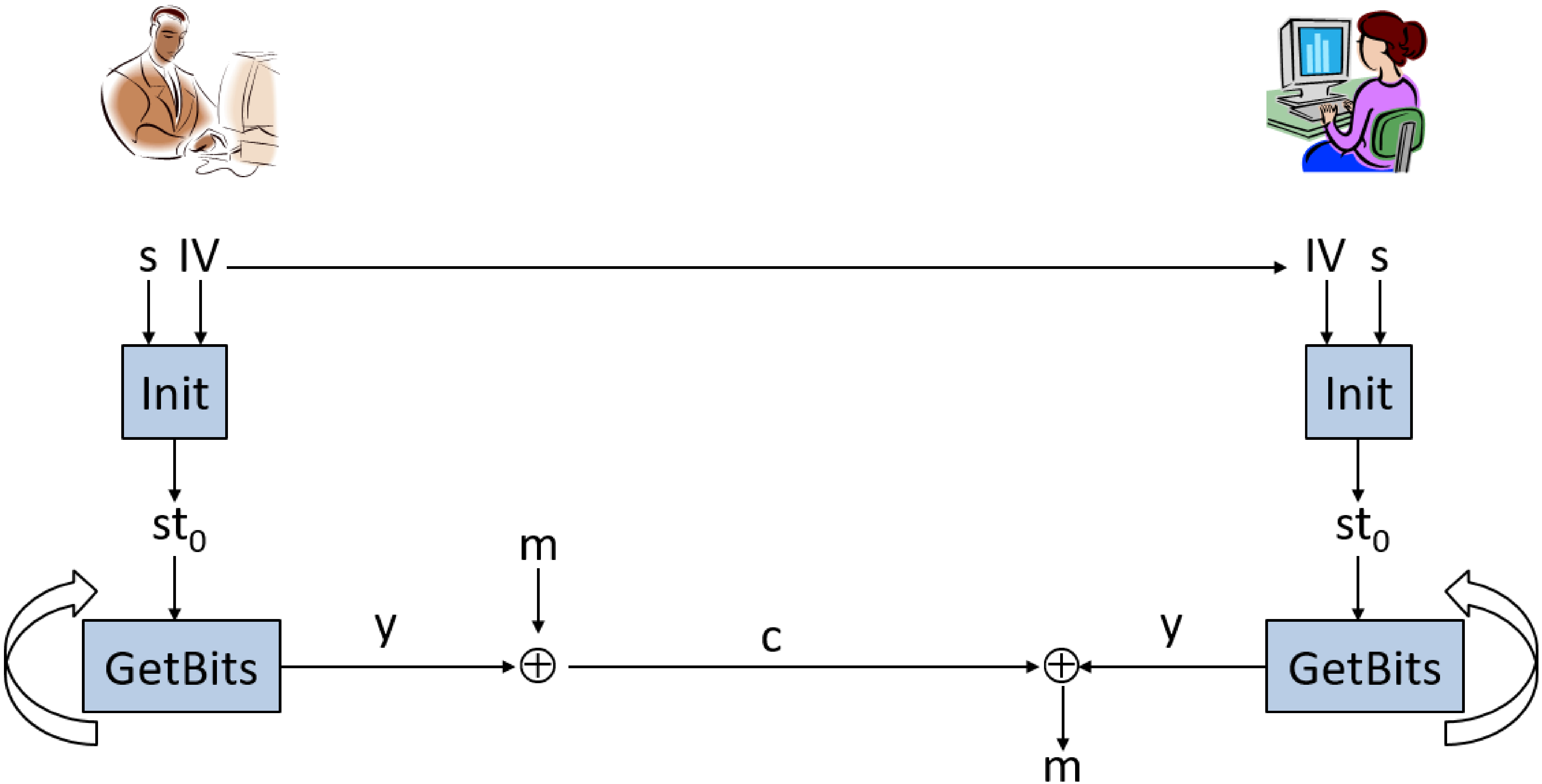
- Choose random IV to encrypt next message

Unsynchronized mode

- Choose random IV to encrypt next message
- Similar to the first CPA-secure scheme we have seen
  - But "natively" handles arbitrary-length messages with better ciphertext expansion

Unsynchronized mode

- Choose random *IV* to encrypt next message



Unsynchronized mode

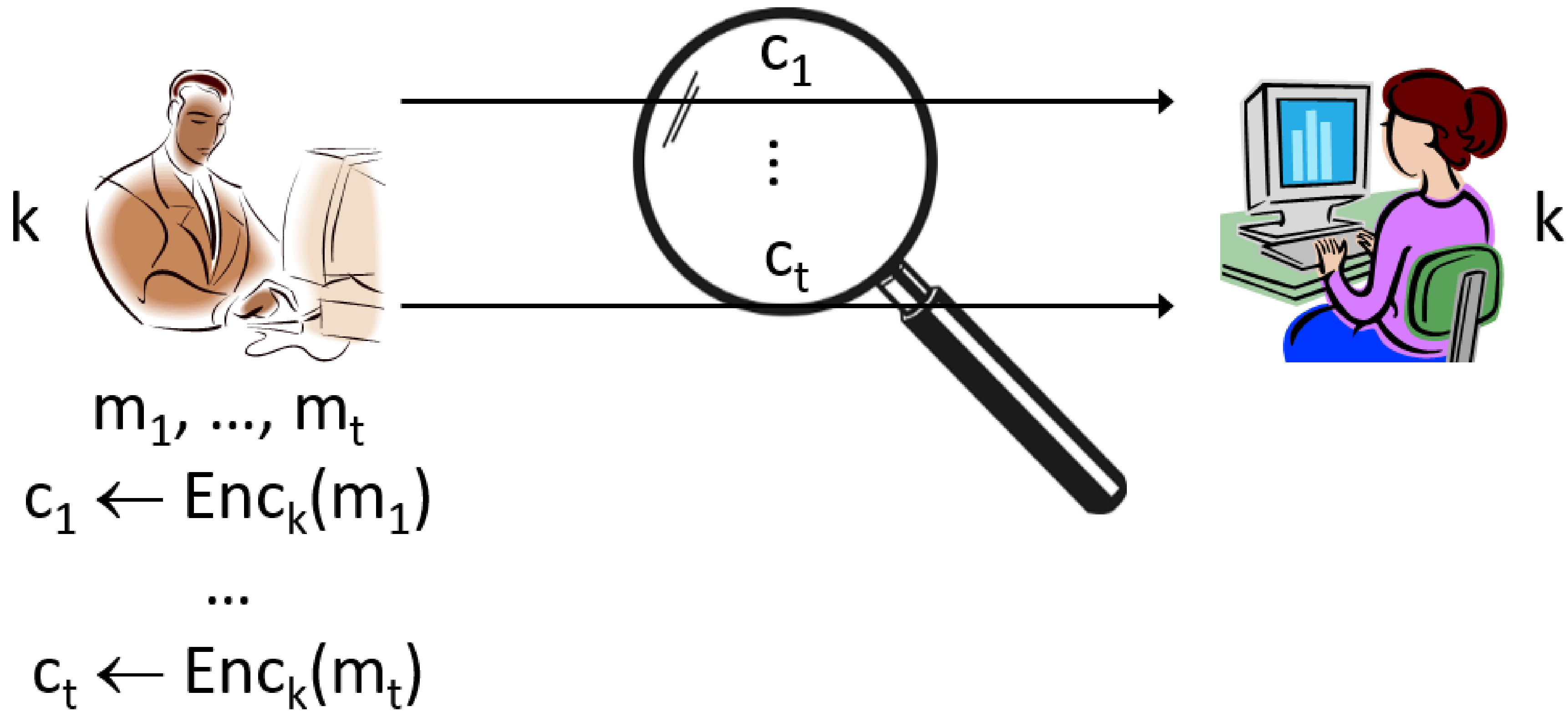
- Note that for security, we require the stream cipher to be a *PRF*
  - I.e., for fixed seed s, the output of the stream cipher when using *different* IVs should all look *uniform* and *independent*
  - The ciphertext $\langle IV, G_\infty(s, IV, 1^{|m|}) \oplus m \rangle$
    $F_k(IV) := G_\infty(k, IV, 1^\ell)$ is a PRF

Passive & active attack

- So far, we have been assuming only a *passive*, *eavesdropping* attacker
  - Even if it can carry out *chosen-plaintext attacks* (CPA)

Passive & active attack

- So far, we have been assuming only a *passive*, *eavesdropping* attacker
  - Even if it can carry out *chosen-plaintext attacks* (CPA)

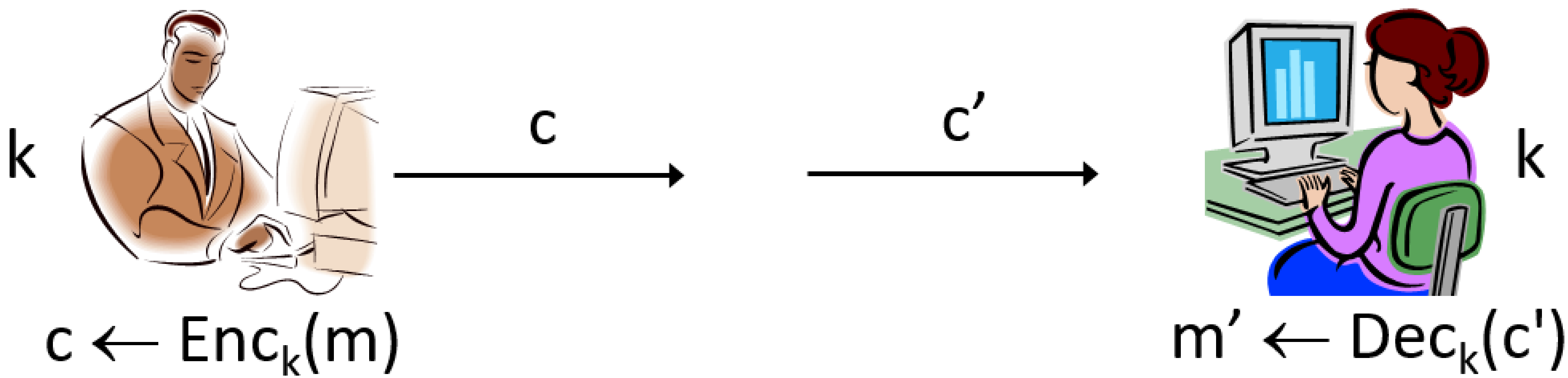


Passive & active attack

- So far, we have been assuming only a *passive*, *eavesdropping* attacker
  - Even if it can carry out *chosen-plaintext attacks* (CPA)
- What if the attacker can be *active*?
  - E.g., interfering with the communication channel

Passive & active attack

- So far, we have been assuming only a *passive*, *eavesdropping* attacker
  - Even if it can carry out *chosen-plaintext attacks* (CPA)
- What if the attacker can be *active*?
  - E.g., interfering with the communication channel



Malleability

- (Informal): A scheme is *malleable* if it is possible to modify a ciphertext and thereby cause a *predictable* change to the plaintext

Malleability

- (Informal): A scheme is *malleable* if it is possible to modify a ciphertext and thereby cause a *predictable* change to the plaintext

- *Malleability* can be dangerous!
  - E.g., encrypted bank transactions

Malleability

- (Informal): A scheme is *malleable* if it is possible to modify a ciphertext and thereby cause a *predictable* change to the plaintext

- *Malleability* can be dangerous!
  – E.g., encrypted bank transactions

- All the schemes we have seen so far are *malleable*
  – E.g., the one-time pad ...

Malleability

- (Informal): A scheme is *malleable* if it is possible to modify a ciphertext and thereby cause a *predictable* change to the plaintext

- *Malleability* can be dangerous!
  - E.g., encrypted bank transactions

- All the schemes we have seen so far are *malleable*
  - E.g., the one-time pad ...

k  $\xrightarrow{c_1c_2\ldots c_n}$ $\xrightarrow{c_1c_2\ldots c'_n}$  k

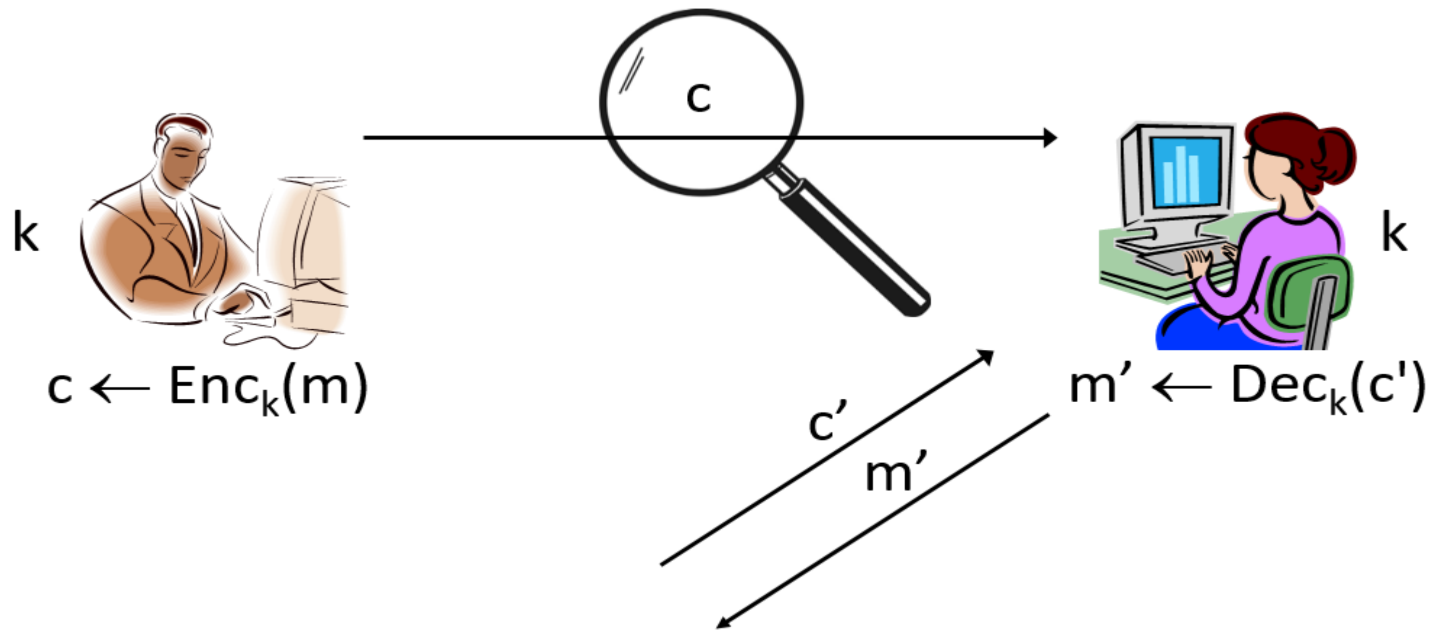
$c := (m_1m_2\ldots m_n)\oplus k$ $m_1m_2\ldots m'_n := (c_1c_2\ldots c'_n)\oplus k$

Passive & active attack

- So far, we have been assuming only a *passive*, *eavesdropping* attacker
  - Even if it can carry out *chosen-plaintext attacks*
- What if the attacker can be *active*?
  - E.g., "impersonating" the sender; injecting communication on the channel

Passive & active attack

- So far, we have been assuming only a *passive*, *eavesdropping* attacker
  - Even if it can carry out *chosen-plaintext attacks*
- What if the attacker can be *active*?
  - E.g., "impersonating" the sender; injecting communication on the channel



Chosen-ciphertext attacks

- Models settings in which the attacker can influence what gets *decrypted*, and observe the effects
  - How to model?

Chosen-ciphertext attacks

- Models settings in which the attacker can influence what gets *decrypted*, and observe the effects
  - How to model?
- Allow attackers to submit ciphertexts of its choice (with one restriction) to the receiver, and learn the corresponding plaintext
  - In addition to being able to carry out a *chosen-plaintext attack*

CCA-security

- Define a randomized experiment $PrivCCA_{A,\Pi}(n)$:
  1. $k \leftarrow Gen(1^n)$
  2. $A(1^n)$ interacts with an *encryption oracle* $Enc_k(\cdot)$, and a *decryption oracle* $Dec_k(\cdot)$, and then outputs m_0, m_1 of the same length
  3. $b \leftarrow \{0, 1\}$, $c \leftarrow Enc_k(m_b)$, give c to A
  4. A can continue to interact with $Enc_k(\cdot)$, $Dec_k(\cdot)$, but may not request decryption of c
  5. A outputs b'; A succeeds if $b = b'$, and experiment evaluates to 1 in this case

CCA-security

- Define a randomized experiment $PrivCCA_{A,\Pi}(n)$:
  1. $k \leftarrow Gen(1^n)$
  2. $A(1^n)$ interacts with an *encryption oracle* $Enc_k(\cdot)$, and a *decryption oracle* $Dec_k(\cdot)$, and then outputs m_0, m_1 of the same length
  3. $b \leftarrow \{0,1\}$, $c \leftarrow Enc_k(m_b)$, give c to A
  4. A can continue to interact with $Enc_k(\cdot)$, $Dec_k(\cdot)$, but may not request decryption of c
  5. A outputs b'; A succeeds if $b = b'$, and experiment evaluates to 1 in this case

Definition 6.1 Π is *secure against chosen-ciphertext attacks (CCA-secure)* if for all PPT attackers A, there is a *negligible* function ϵ such that

$$\Pr[PrivCCA_{A,\Pi}(n) = 1] \leq 1/2 + \epsilon(n)$$

Chosen-ciphertext attacks and malleability

- If a scheme is *malleable*, then it cannot be *CCA-secure*
  - Modify c, submit the modified ciphertext c' to the decryption oracle and determine original message based on the result

Chosen-ciphertext attacks and malleability

- If a scheme is *malleable*, then it cannot be *CCA-secure*
  – Modify c, submit the modified ciphertext c' to the decryption oracle and determine original message based on the result
- *CCA-security* implies *non-malleability*

Chosen-ciphertext attacks and malleability

- If a scheme is *malleable*, then it cannot be *CCA-secure*
  - Modify c, submit the modified ciphertext c' to the decryption oracle and determine original message based on the result

- *CCA-security* implies *non-malleability*

  $Gen(1^n)$: choose a uniform key $k \in \{0,1\}^n$
  $Enc_k(m)$, for $|m| = |k|$
  – Choose uniform $r \in \{0,1\}^n$ (*nonce/ initialization vector*)
  – Output ciphertext $\langle r, F_k(r) \oplus m \rangle$
  $Dec_k(c_1, c_2)$: output $c_2 \oplus F_k(c_1)$

  Theorem 5.1 If F is a pseudorandom function, then this scheme is *CPA-secure*.

CCA-security

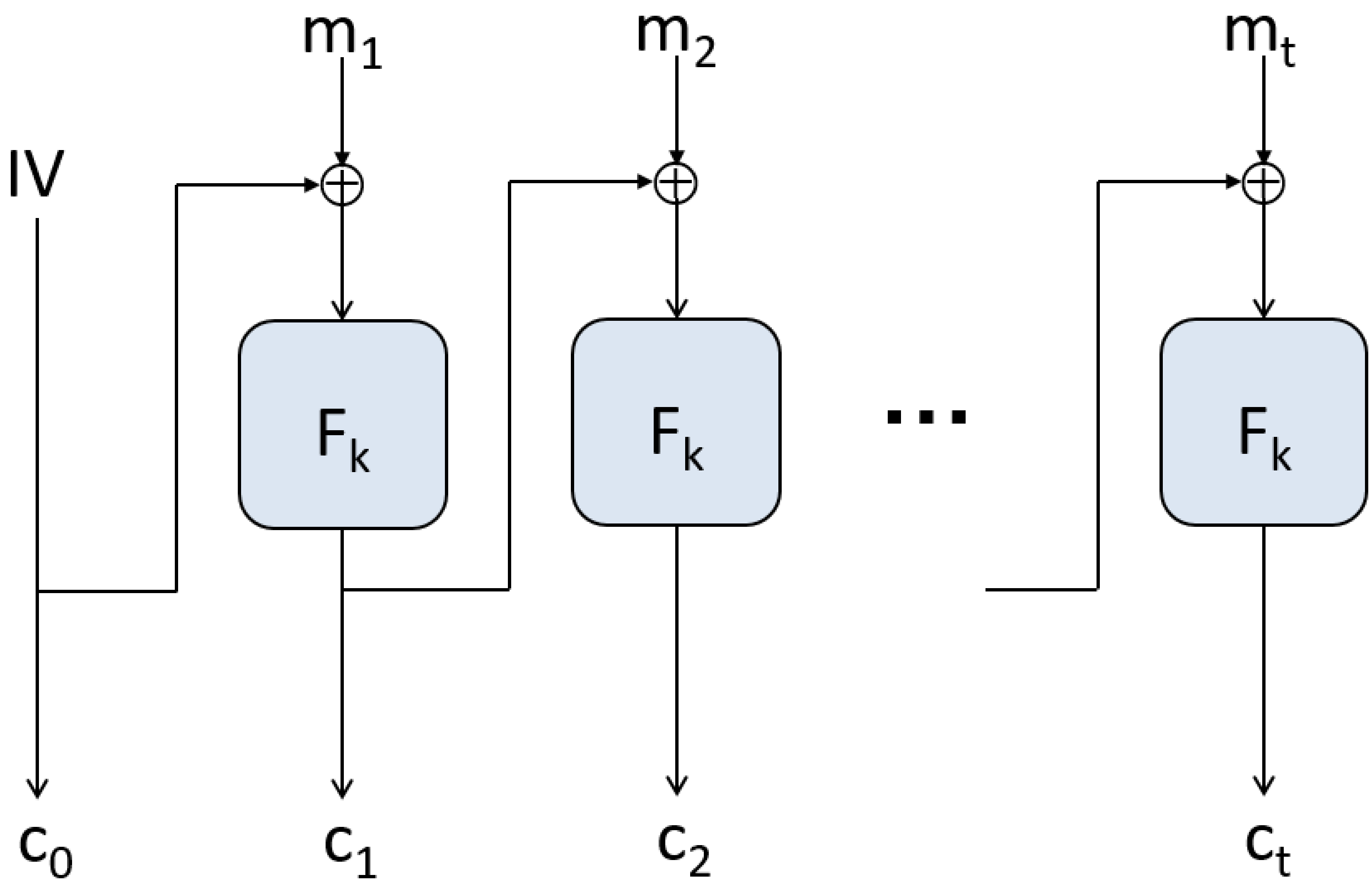
- In the definition of *CCA-security*, the attacker can obtain the decryption of any ciphertext of its choice (besides the challenge ciphertext)
  - Is this realistic?

CCA-security

- In the definition of *CCA-security*, the attacker can obtain the decryption of any ciphertext of its choice (besides the challenge ciphertext)
  - Is this realistic?

- We show a scenario where:
  - *One bit* about decrypted ciphertexts is leaked
  - The scenario occurs in the real world!
  - This can be exploited to learn the entire plaintext!

CBC mode



Arbitrary-length messages

- Message $\rightarrow$ encoded data $\rightarrow$ ciphertext

Arbitrary-length messages

- Message $\rightarrow$ encoded data $\rightarrow$ ciphertext
- PKCS #5 encoding:
  - Assume message is an integeral # of bytes
  - Let L be the block length (in bytes) of the cipher
  - Let $b \geq 1$ be # of bytes that need to be appended to the message to get length a multiple of L
    - $1 \leq b \leq L$; note $b \neq 0$
  - Append b (encoded in 1 byte), b times
    - I.e., if 3 bytes of padding are needed, append 0x030303

Decryption?

- To Decrypt:
  - Use CBC-mode decryption to obtain encoded data
  - Say, the final byte of encoded data has value b

Decryption?

- To Decrypt:
  - Use CBC-mode decryption to obtain encoded data
  - Say, the final byte of encoded data has value b
    - If $b = 0$ or $b > L$, return "error"
    - If final b bytes of encoded data are not all equal to b, return "error"
    - Otherwise, strip off the final b bytes of the encoded data, and output what remains as the message

Example ($L = 8$)

AB	01	$4F$	21	00	$7C$

Example ($L = 8$)

AB	01	$4F$	21	00	$7C$	02	02

Example ($L = 8$)

AB	01	$4F$	21	00	$7C$	02	02

$$\downarrow$$

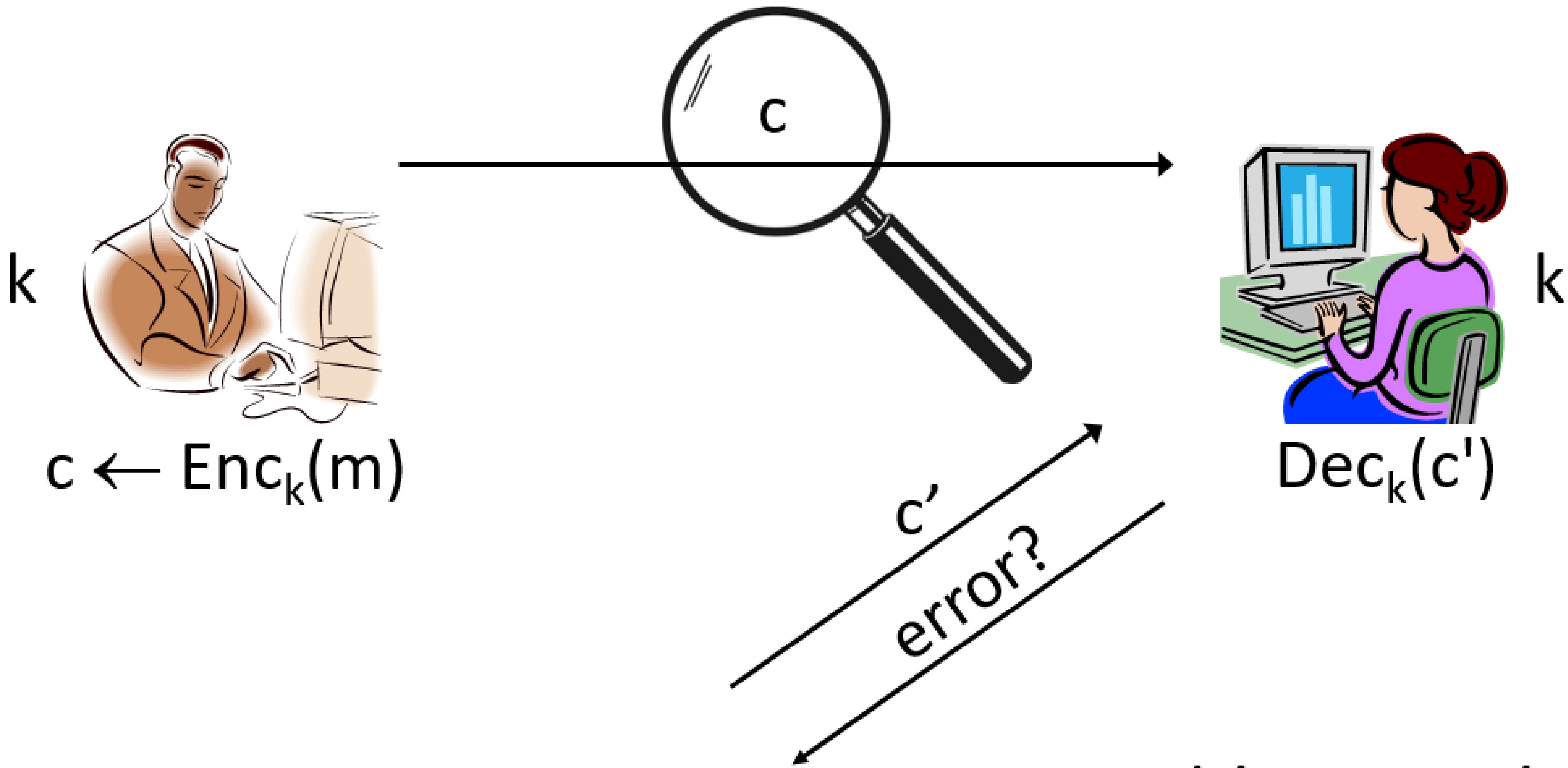
AB	01	$4F$	21	00	$7C$	02	02

Example ($L = 8$)

AB	01	$4F$	21	00	$7C$	02	02

AB	01	$4F$	21	00	$7C$

Padding oracles



Padding oracle!

Padding oracles

- Padding oracles are frequently present in, e.g., web applications
- Even if an error is not explicitly returned, an attacker might be able to detect differences in timing, behavior, etc.

Padding oracles

- Padding oracles are frequently present in, e.g., web applications
- Even if an error is not explicitly returned, an attacker might be able to detect differences in timing, behavior, etc.
- Main idea of the attack
  - Consider a two-block ciphertext IV, c
    - Encoded data $= F_k^{-1}(c) \oplus IV$

Padding oracles

- Padding oracles are frequently present in, e.g., web applications
- Even if an error is not explicitly returned, an attacker might be able to detect differences in timing, behavior, etc.
- Main idea of the attack
  - Consider a two-block ciphertext IV, c
    - Encoded data $= F_k^{-1}(c) \oplus IV$
  - Main observation: If an attacker modifies the ith byte of IV, this causes a predictable change (only) to the ith byte of the encoded data

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:

XX	XX	XX	XX	XX	XX	XX	XX

$\oplus$

IV:

AB	01	4F	21	00	7C	02	9E

$=$

Encoded data:

XX	XX	XX	XX	XX	XX	XX	XX

"Success"

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	XX
				$\oplus$				
IV:		01	4F	21	00	7C	02	9E
				$=$				
Encoded data:	XX	XX	XX	XX	XX	XX	XX	XX

"Success"

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	XX

$$\oplus$$

IV:		01	4F	21	00	7C	02	9E

$$=$$

Encoded data:		XX	XX	XX	XX	XX	XX	XX

"Success"

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	XX

$$\oplus$$

IV:			4F	21	00	7C	02	9E

$$=$$

Encoded data:			XX	XX	XX	XX	XX	XX

"Success"

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	XX
				$\oplus$				
IV:				21	00	7C	02	9E
				$=$				
Encoded data:				XX	XX	XX	XX	XX

"Success" $\longrightarrow$ "Error"

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	XX

$$\oplus$$

IV:				21	00	7C	02	9E

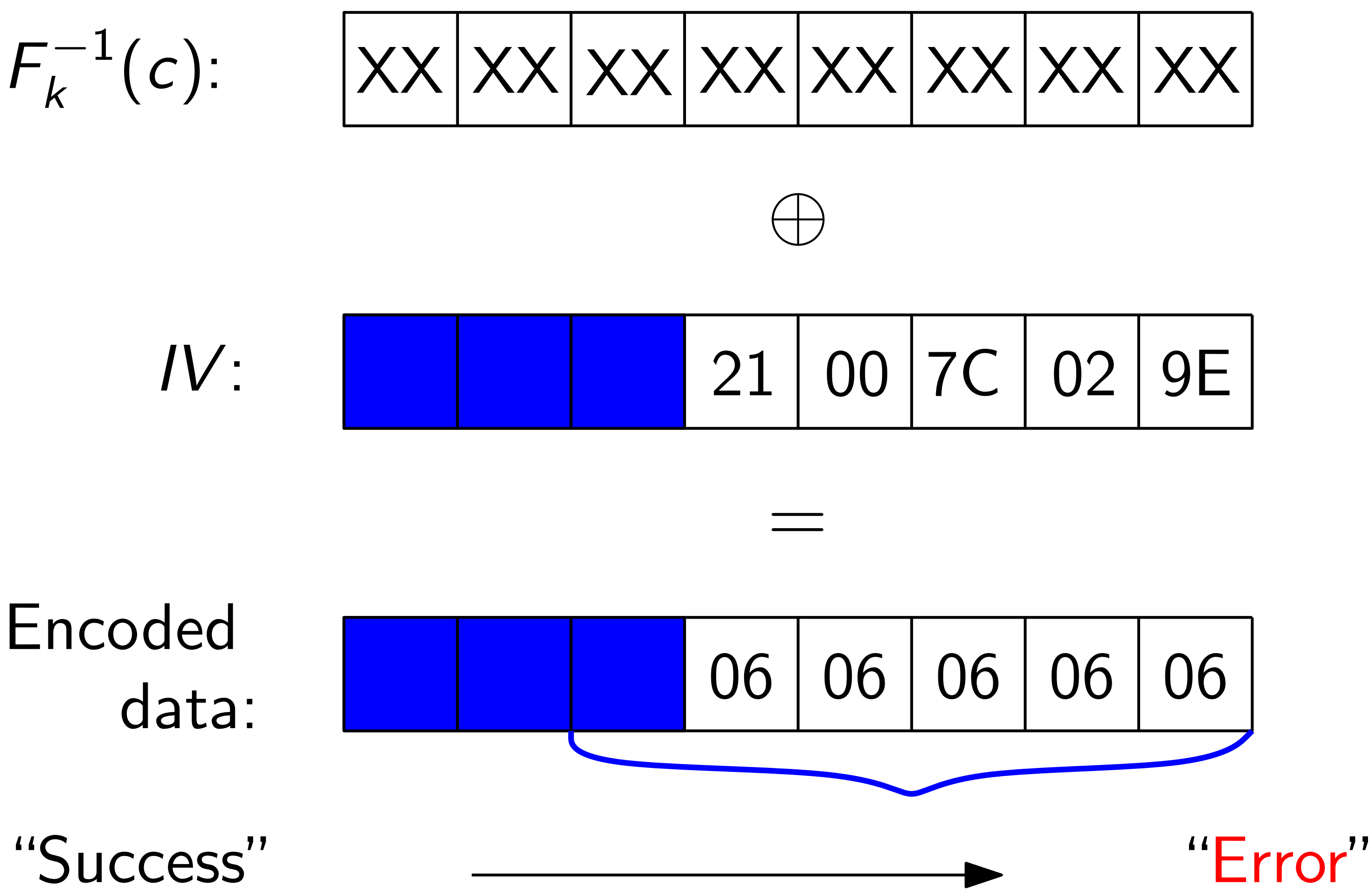
$$=$$

Encoded data:				XX	XX	XX	XX	XX

"Success" $\longrightarrow$ "Error"

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$



Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

0x9E $\oplus$ 0x06

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	98
				$\oplus$				
IV:				21	00	7C	02	9E
				$=$				
Encoded data:				06	06	06	06	06

“Success” $\longrightarrow$ “Error”

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	98
				$\oplus$				
IV:	AB	01	4F	21	00	7C	02	9E
				$=$				
Encoded data:	XX	XX	06	06	06	06	06	06

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	98
$\oplus$								$0x98 \oplus 0x07$
IV:	AB	01	4F	21	00	7C	02	9F
$=$								
Encoded data:	XX	XX	06	06	06	06	06	06

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	98
				$\oplus$				
IV:	AB	01	4F	21	00	7C	02	9F
				$=$				
Encoded data:	XX	XX	06	06	06	06	06	07

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:

XX	XX	XX	XX	XX	XX	XX	98

$\oplus$ $\qquad$ 0x02 $\oplus$ 0x06 $\oplus$ 0x07

IV:

AB	01	4F	21	00	7C	02	9F

$=$

Encoded data:

XX	XX	06	06	06	06	06	07

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	98

$\oplus$

$0x02 \oplus 0x06 \oplus 0x07$

IV:	AB	01	4F	21	00	7C	03	9F

$=$

Encoded data:	XX	XX	06	06	06	06	06	07

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	98
				$\oplus$				
IV:	AB	01	4F	21	00	7C	03	9F
				$=$				
Encoded data:	XX	XX	06	06	06	06	07	07

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	98
				$\oplus$				
IV:	AB	01	4E	20	01	7D	03	9F
				$=$				
Encoded data:	XX	XX	07	07	07	07	07	07

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:

XX	XX	XX	XX	XX	XX	XX	98

$\oplus$

IV:

AB	00	4E	20	01	7D	03	9F

$=$

Encoded data:

XX	XX	07	07	07	07	07	07

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	98
				$\oplus$				
IV:	AB	01	4E	20	01	7D	03	9F
				$=$				
Encoded data:	XX	XX	07	07	07	07	07	07

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	98
				$\oplus$				
IV:	AB	02	4E	20	01	7D	03	9F
				$=$				
Encoded data:	XX	XX	07	07	07	07	07	07

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	98
				$\oplus$				
IV:	AB	41	4E	20	01	7D	03	9F
				$=$				
Encoded data:	XX	XX	07	07	07	07	07	07

"Success"

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:

XX	XX	XX	XX	XX	XX	XX	98

$\oplus$

IV:

AB	41	4E	20	01	7D	03	9F

$=$

Encoded data:

XX	07	07	07	07	07	07	07

"Success"

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:	XX	XX	XX	XX	XX	XX	XX	98

$$\oplus$$

IV:	AB	41	4E	20	01	7D	03	9F

$$=$$

Encoded data:	XX	07	07	07	07	07	07	07

$$\text{XX} \oplus \text{0x41} = \text{0x07}$$

"Success"

$$\Rightarrow \text{XX} = \text{0x41} \oplus \text{0x07}$$

Padding-oracle attack

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c)$:

XX	XX	XX	XX	XX	XX	XX	98

$$\oplus$$

IV:

AB	41	4E	20	01	7D	03	9F

$$=$$

Encoded data:

XX	07	07	07	07	07	07	07

"Success"

$$\mathrm{XX} \oplus \mathrm{0x41} = \mathrm{0x07}$$
$$\Rightarrow \mathrm{XX} = \mathrm{0x41} \oplus \mathrm{0x07}$$
$$\Rightarrow \text{plaintext byte} = \mathrm{XX} \oplus \mathrm{0x01} = \mathrm{0x47}$$

Attack complexity

- $\leq L$ tries to learn the $\#$ of padding bytes (b)

- $\leq 2^8 = 256$ tries to learn each plaintext byte

CCA-security: a summary

- *Chosen-ciphertext attacks* represent a significant, real-world threat
- Modern encryption schemes are designed to be *CCA-secure*
- None of the schemes we have seen so far are *CCA-secure*

Secrecy vs. integrity

- So far we have been concerned with ensuring *secrecy* of communication

Secrecy vs. integrity

- So far we have been concerned with ensuring *secrecy* of communication

- What about *integrity*?
  – I.e., ensuring that a received message originated from the intended party, and was not modified
    – Even if an attacker controls the channel!
  – Standard error-correction techniques are not enough!
    – The right tool is a *message authentication code*

Secrecy vs. integrity

k 

m, t $\longrightarrow$ m', t' $\longrightarrow$

 k

m

t = $Mac_k(m)$

$Vrfy_k(m', t') = 1$?

Secrecy vs. integrity

- *Secrecy* and *integrity* are *orthogonal* concerns
  - Possible to have either one without the other
  - Sometimes you might want one without the other
  - Most often, both are needed

Secrecy vs. integrity

- *Secrecy* and *integrity* are *orthogonal* concerns
  - Possible to have either one without the other
  - Sometimes you might want one without the other
  - Most often, both are needed

- Encryption does not (in general) provide any integrity
  - Integrity is even stronger than *non-malleabity*
  - None of the schemes we have seen so far provide any integrity

Message authentication code (MAC)

- A *message authentication code* is defined by three PPT algorithms (*Gen*, *Mac*, *Vrfy*):
  - *Gen*: take as input 1^n; outputs k. (assume $|k| \geq n$.)
  - *Mac*: take as input key k and message $m \in \{0,1\}^*$; outputs *tag* t: $t := Mac_k(m)$
  - *Vrfy*: takes key k, message m, and tag t as input; outputs 1 ("accept") or 0 ("reject")

Message authentication code (MAC)

- A *message authentication code* is defined by three PPT algorithms (Gen, Mac, $Vrfy$):
  - Gen: take as input 1^n; outputs k. (assume $|k| \geq n$.)
  - Mac: take as input key k and message $m \in \{0,1\}^*$; outputs *tag* t: $t := Mac_k(m)$
  - $Vrfy$: takes key k, message m, and tag t as input; outputs 1 ("accept") or 0 ("reject")

For all m and all k output by Gen,

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

Security

- Threat model
  - "Adaptive chosen-message attack"
  - Assume the attacker can induce the sender to authenticate *messages of the attacker's choice*

Security

- Threat model
  - "Adaptive chosen-message attack"
  - Assume the attacker can induce the sender to authenticate *messages of the attacker's choice*
- Security goal
  - "Existential unforgeability"
  - Attacker should be unable to forge a valid tag on *any* message not previously authenticated by the sender

MAC

k

$t_1 := \mathrm{Mac}_k(m_1)$

$t_2 := \mathrm{Mac}_k(m_2)$

…

$t_i := \mathrm{Mac}_k(m_i)$

$\xrightarrow{m_1,\, t_1}$

$\xrightarrow{m_2,\, t_2}$

⋮

$\xrightarrow{m_i,\, t_i}$

$\xrightarrow{m',\, t'}$

k

$\mathrm{Vrfy}_k(m', t')$??

Formal definition

- Fix A, Π. Define a randomized experiment $Forge_{A,\Pi}(n)$:
  1. $k \leftarrow Gen(1^n)$
  2. $A(1^n)$ interacts with an *oracle* $Mac_k(\cdot)$; let M be the set of messages submitted to this oracle
  3. A outputs (m, t)
  4. A *succeeds*, and the experiment evaluates to 1, if $Vrfy_k(m, t) = 1$ and $m \notin M$

Formal definition

- Fix A, Π. Define a randomized experiment $Forge_{A,\Pi}(n)$:
  1. $k \leftarrow Gen(1^n)$
  2. $A(1^n)$ interacts with an *oracle* $Mac_k(\cdot)$; let M be the set of messages submitted to this oracle
  3. A outputs (m, t)
  4. A *succeeds*, and the experiment evaluates to 1, if $Vrfy_k(m, t) = 1$ and $m \notin M$

Definition 6.2 Π is *secure* if for all PPT attackers A, there is a *negligible* function ϵ such that

$$\Pr[Forge_{A,\Pi}(n) = 1] \leq \epsilon(n)$$

Security

- Is the definition too strong?
  - We don't want to make any assumptions about what the sender might authenticate
  - We don't want to make any assumptions about what forgeries are "meaningful"

Security

- Is the definition too strong?
  - We don't want to make any assumptions about what the sender might authenticate
  - We don't want to make any assumptions about what forgeries are "meaningful"

- An *MAC* satisfying this definition can be used anywhere integrity is needed

Replay attacks

- Replay attacks are not prevented
  - No *stateless* mechanism can prevent them
- Replay attacks are often a significant real-world concern
- Need to protect against replay attacks at a higher level
  - Decision about what to do with a replayed message is application-dependent

A fixed-length MAC

- Intuition: we need a keyed function *Mac* such that:
  - Given $Mac_k(m_1)$, $Mac_k(m_2)$, $\ldots$,
  - It is infeasible to predict the value $Mac_k(m)$ for any $m \notin \{m_1, m_2, \ldots\}$

A fixed-length MAC

- Intuition: we need a keyed function *Mac* such that:
  - Given $Mac_k(m_1)$, $Mac_k(m_2)$, $\ldots$,
  - It is infeasible to predict the value $Mac_k(m)$ for any $m \notin \{m_1, m_2, \ldots\}$

- Let *Mac* be a PRF!

Construction

- Let F be a length-preserving PRF (aka block cipher)

Construction

- Let F be a length-preserving PRF (aka block cipher)

- Construct the following MAC Π:
  - Gen: choose a uniform key k for F
  - $Mac_k(m)$: output $F_k(m)$
  - $Vrfy_k(m, t)$: output 1 iff $F_k(m) = t$

Construction

- Let F be a length-preserving PRF (aka block cipher)

- Construct the following MAC Π:
  - Gen: choose a uniform key k for F
  - $Mac_k(m)$: output $F_k(m)$
  - $Vrfy_k(m, t)$: output 1 iff $F_k(m) = t$

- **Theorem 6.3** Π is a *secure* MAC

Next Lecture

- proof, authenticated encryption ...