# Cryptography

Lecture 10

# Announcements

- HW3 due on Wednesday, 3/6

# Agenda

- Last time:
  - MACs (K/L 4.1, 4.2, 4.3)
- This time:
  - Domain Extension for MACs (K/L 4.4) and Class Exercise solutions
  - CCA security (K/L 3.7)
  - Authenticated Encryption (K/L 4.5)

# Message Authentication Codes

Definition: A message authentication code (MAC) consists of three probabilistic polynomial-time algorithms $(Gen, Mac, Vrfy)$ such that:

1. The key-generation algorithm $Gen$ takes as input the security parameter $1^n$ and outputs a key $k$ with $|k| \geq n$.
2. The tag-generation algorithm $Mac$ takes as input a key $k$ and a message $m \in \{0,1\}^*$, and outputs a tag $t$. $t \leftarrow Mac_k(m)$.
3. The deterministic verification algorithm $Vrfy$ takes as input a key $k$, a message $m$, and a tag $t$. It outputs a bit $b$ with $b = 1$ meaning valid and $b = 0$ meaning invalid. $b \coloneqq Vrfy_k(m, t)$.

It is required that for every $n$, every key $k$ output by $Gen(1^n)$, and every $m \in \{0,1\}^*$, it holds that $Vrfy_k\big(m, Mac_k(m)\big) = 1$.

# Unforgeability for MACs

Consider a message authentication code $\Pi = (Gen, Mac, Vrfy)$, any adversary $A$, and any value $n$ for the security parameter.

Experiment $MACforge_{A,\Pi}(n)$

Adversary $A(1^n)$ — Challenger

$k \leftarrow Gen(1^n)$

$A^{Mac_k(\cdot)}$

$m'$ →

← $t'$

⋮

$(m, t)$ →

$Q$ is the set of all messages $m'$ queried by $A$

$MACforge_{A,\Pi}(n) = 1$ if both of the following hold:

1. $m \notin Q$
2. $Vrfy_k(m, t) = 1$

Otherwise, $MACforge_{A,\Pi}(n) = 0$

# Security of MACs

The message authentication experiment $MACforge_{A,\Pi}(n)$:

1. A key $k$ is generated by running $Gen(1^n)$.
2. The adversary $A$ is given input $1^n$ and oracle access to $Mac_k(\cdot)$. The adversary eventually outputs $(m, t)$. Let $Q$ denote the set of all queries that $A$ asked its oracle.
3. $A$ succeeds if and only if (1) $Vrfy_k(m, t) = 1$ and (2) $m \notin Q$. In that case, the output of the experiment is defined to be 1.

# Security of MACs

Definition: A message authentication code $\Pi = (Gen, Mac, Vrfy)$ is existentially unforgeable under an adaptive chosen message attack if for all probabilistic polynomial-time adversaries $A$, there is a negligible function $neg$ such that:

$$\Pr\left[MACforge_{A,\Pi}(n) = 1\right] \leq neg(n).$$

# Strong Unforgeability for MACs

Consider a message authentication code $\Pi = (Gen, Mac, Vrfy)$, any adversary $A$, and any value $n$ for the security parameter.

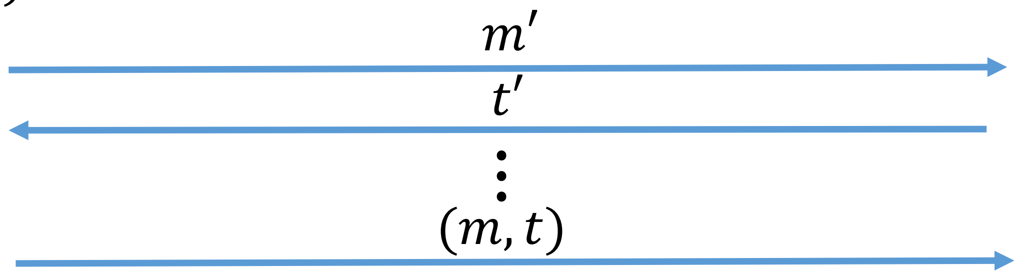
Experiment $MACsforge_{A,\Pi}(n)$

Adversary $A(1^n)$

Challenger

$k \leftarrow Gen(1^n)$

$A^{Mac_k(\cdot)}$




$Q$ is the set of all message, tag pairs $(m', t')$ queried/received by $A$

$MACsforge_{A,\Pi}(n) = 1$ if both of the following hold:

1. $m \notin Q$
2. $Vrfy_k(m, t) = 1$

Otherwise, $MACsforge_{A,\Pi}(n) = 0$

# Strong MACs

The strong message authentication experiment $MACsforge_{A,\Pi}(n)$:

1. A key $k$ is generated by running $Gen(1^n)$.
2. The adversary $A$ is given input $1^n$ and oracle access to $Mac_k(\cdot)$. The adversary eventually outputs $(m, t)$. Let $Q$ denote the set of all pairs $(m, t)$ that $A$ asked its oracle.
3. $A$ succeeds if and only if (1) $Vrfy_k(m, t) = 1$ and (2) $(m, t) \notin Q$. In that case, the output of the experiment is defined to be 1.

# Strong MACs

Definition: A message authentication code $\Pi = (Gen, Mac, Vrfy)$ is a strong MAC if for all probabilistic polynomial-time adversaries $A$, there is a negligible function $neg$ such that:

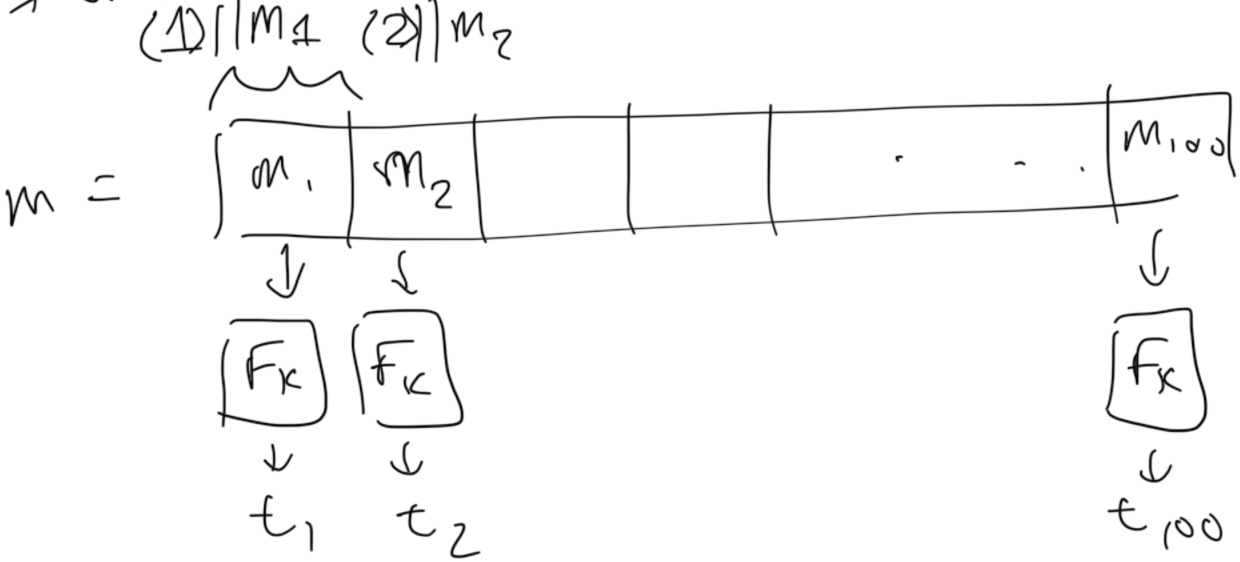
$$\Pr\left[MACsforge_{A,\Pi}(n) = 1\right] \leq neg(n).$$

# Domain Extension for MACs

Assume we have a MAC fixed length msgs

Domain: $\{0,1\}^n$ (e.g. the domain of $F_k$)

Domain Ext: use ↑ to get MACs for longer msg)




Not Secure

Not Efficient

# CBC-MAC

Let $F$ be a pseudorandom function, and fix a length function $\ell$. The basic CBC-MAC construction is as follows:

- $Mac$: on input a key $k \in \{0,1\}^n$ and a message $m$ of length $\ell(n) \cdot n$, do the following:
  1. Parse $m$ as $m = m_1, \dots, m_\ell$ where each $m_i$ is of length $n$.
  2. Set $t_0 := 0^n$. Then, for $i = 1\ to\ \ell$:

     Set $t_i := F_k(t_{i-1} \oplus m_i)$.

  Output $t_\ell$ as the tag.
- $Vrfy$: on input a key $k \in \{0,1\}^n$, a message $m$, and a tag $t$, do: If $m$ is not of length $\ell(n) \cdot n$ then output $0$. Otherwise, output $1$ if and only if $t = Mac_k(m)$.

previously: CBC-Enc

## Length Extension Attack

# CBC-MAC

1. Query oracle $m_1^* \| m_2^* \| m_3^*$
   get $t_3$

2. Query oracle on $t_3 \oplus m_4^* \| m_5^* \| m_6^*$
   get $t^*$

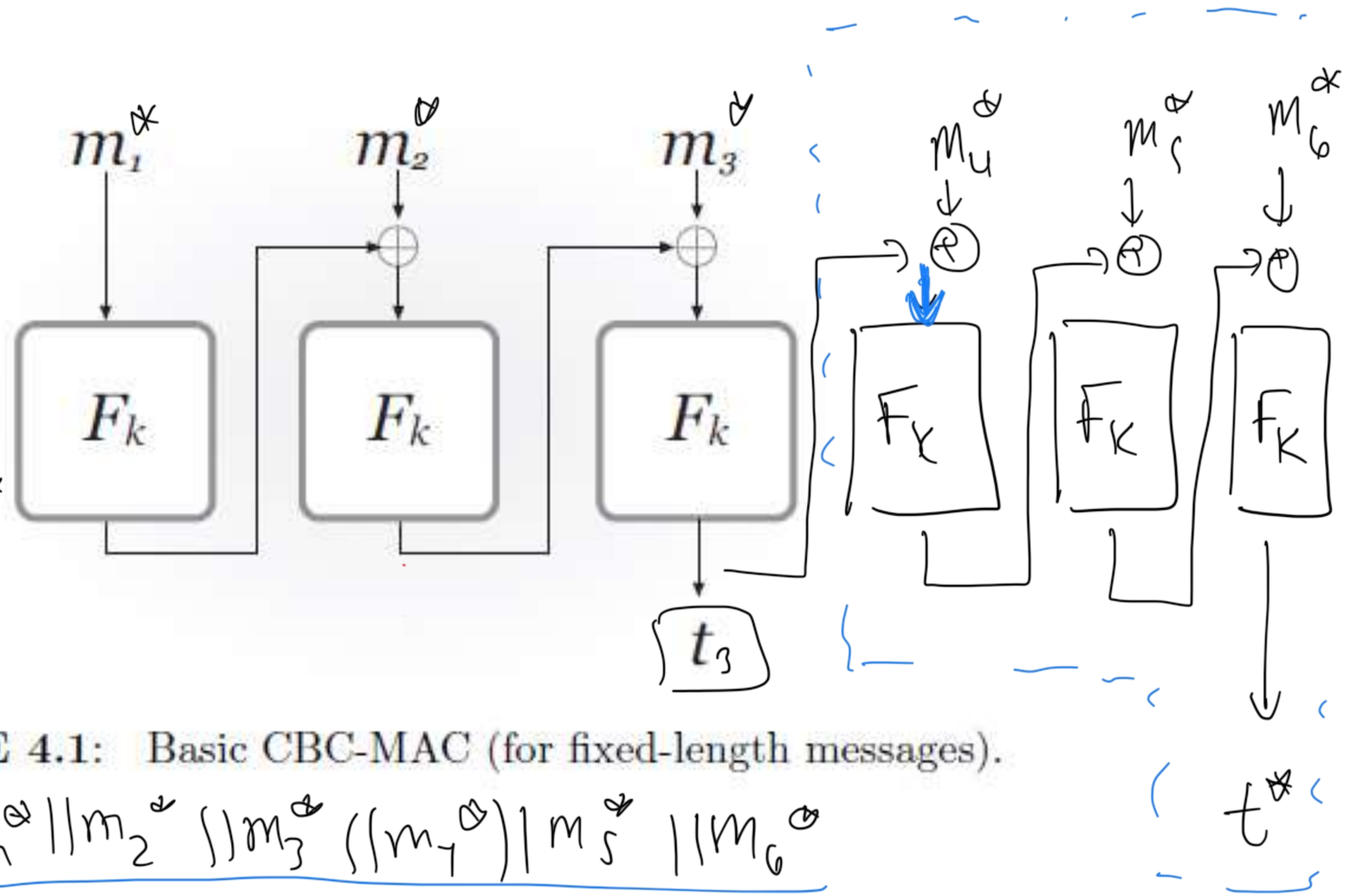



**FIGURE 4.1:** Basic CBC-MAC (for fixed-length messages).

3. Forgery: $m^* = m_1^* \| m_2^* \| m_3^* \| m_4^* \| m_5^* \| m_6^*$
   tag: $t^*$

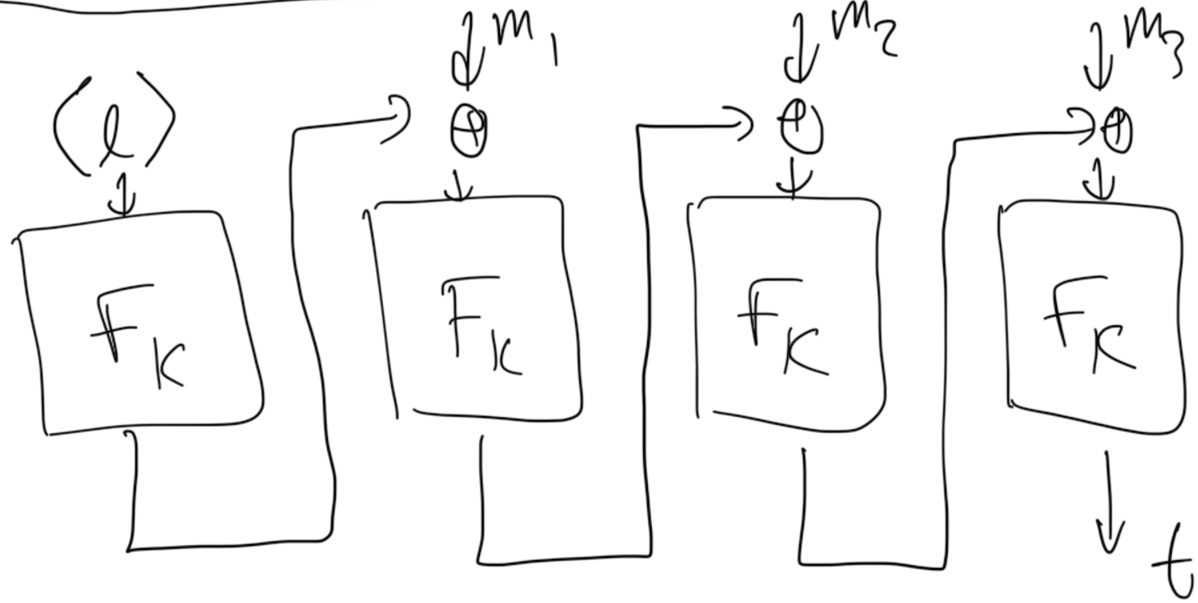
Prefix-Free encoding:
① Encode msg w/ PF encoding
② Run CBC MAC.

Encode 01101 $\longrightarrow$ $\underbrace{\langle 5 \rangle}_{\text{length}} \| \underbrace{01101}_{\text{message}}$

$m_1$

$m_2$

---




$\underbrace{m_1 \| m_2 \| m_3}_{\ell}$

CBC-MAC + Prefix-free encoding

Secure for arbitrary length msgs.

$101 \longrightarrow \langle 3 \rangle \| 101$

$10101 \longrightarrow \langle 5 \rangle \| 10101$

$110 \longrightarrow \langle 3 \rangle \| 110$

$101 \longrightarrow \langle 3 \rangle \| 101$

# Chosen Ciphertext Security

# CCA Security*

Encryption

Consider a private-key encryption scheme $\Pi = (Gen, Enc, Dec)$, any adversary $A$, and any value $n$ for the security parameter.

Experiment $PrivK_{A,\Pi}^{cca}(n)$

Adversary $A(1^n)$

Challenger

# CCA Security*

Consider a private-key encryption scheme $\Pi = (Gen, Enc, Dec)$, any adversary $A$, and any value $n$ for the security parameter.

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

Adversary $A(1^n)$

Challenger

$k \leftarrow Gen(1^n)$

# CCA Security*

Consider a private-key encryption scheme $\Pi = (Gen, Enc, Dec)$, any adversary $A$, and any value $n$ for the security parameter.

Experiment $PrivK_{A,\Pi}^{cca}(n)$

Adversary $A(1^n)$

Challenger

$k \leftarrow Gen(1^n)$

$A^{Enc_k(\cdot),Dec_k(\cdot)}$

$m'/c'$

$c'/m'$

⋮

# CCA Security*

Consider a private-key encryption scheme $\Pi = (Gen, Enc, Dec)$, any adversary $A$, and any value $n$ for the security parameter.

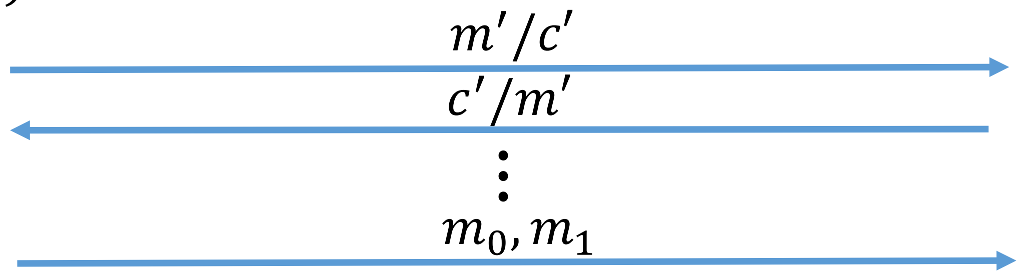
Experiment $PrivK_{A,\Pi}^{cca}(n)$

Adversary $A(1^n)$

Challenger

$k \leftarrow Gen(1^n)$

$A^{Enc_k(\cdot),Dec_k(\cdot)}$



# CCA Security*

Consider a private-key encryption scheme $\Pi = (Gen, Enc, Dec)$, any adversary $A$, and any value $n$ for the security parameter.

Experiment $PrivK_{A,\Pi}^{cca}(n)$

Adversary $A(1^n)$ | Challenger

$A^{Enc_k(\cdot),Dec_k(\cdot)}$

$k \leftarrow Gen(1^n)$

$m'/c'$ →

← $c'/m'$

⋮

$m_0, m_1$ →

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

← $c$

$c \leftarrow Enc_k(m_b)$

# CCA Security*

Consider a private-key encryption scheme $\Pi = (Gen, Enc, Dec)$, any adversary $A$, and any value $n$ for the security parameter.

Experiment $PrivK_{A,\Pi}^{cca}(n)$

Adversary $A(1^n)$

Challenger

$k \leftarrow Gen(1^n)$

$A^{Enc_k(\cdot), Dec_k(\cdot)}$

$m'/c'$ →

← $c'/m'$

⋮

$m_0, m_1$ →

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

$c \leftarrow Enc_k(m_b)$

← $c$

$A^{Enc_k(\cdot), Dec_k(\cdot)}$

$m'/c'$ →

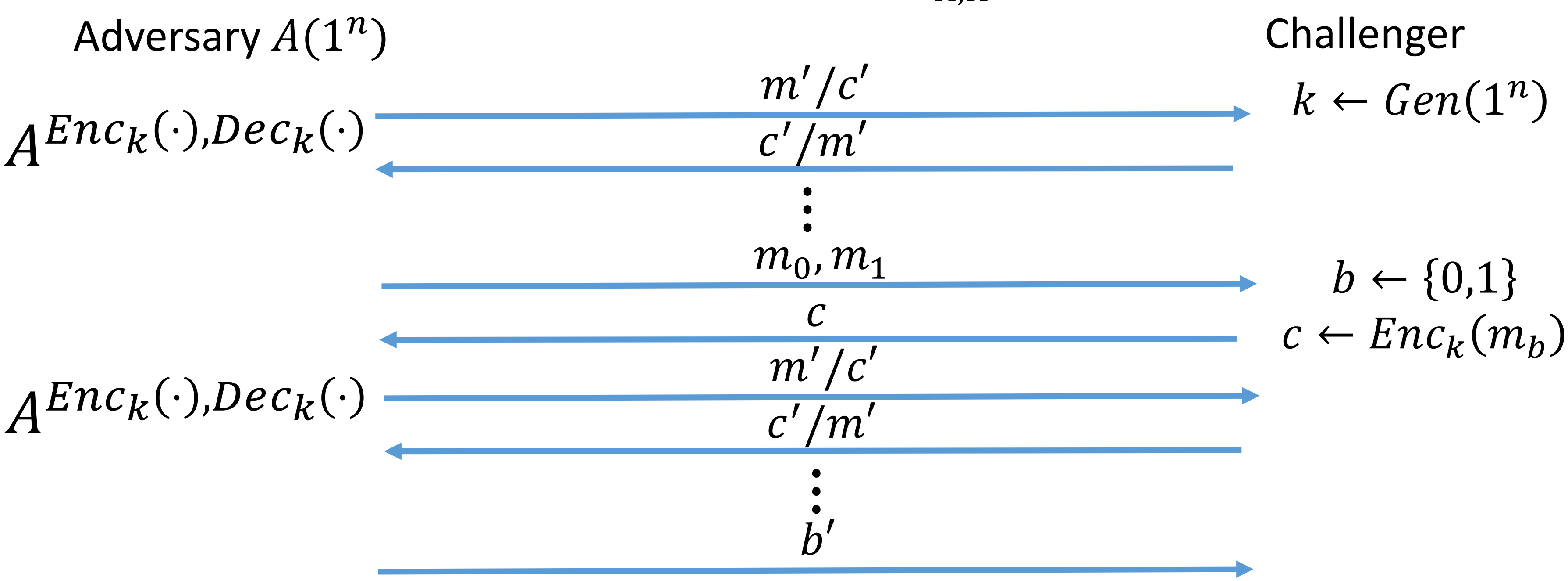
← $c'/m'$

⋮

# CCA Security*

Consider a private-key encryption scheme $\Pi = (Gen, Enc, Dec)$, any adversary $A$, and any value $n$ for the security parameter.

Experiment $PrivK_{A,\Pi}^{cca}(n)$



# CCA Security*

Consider a private-key encryption scheme $\Pi = (Gen, Enc, Dec)$, any adversary $A$, and any value $n$ for the security parameter.

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

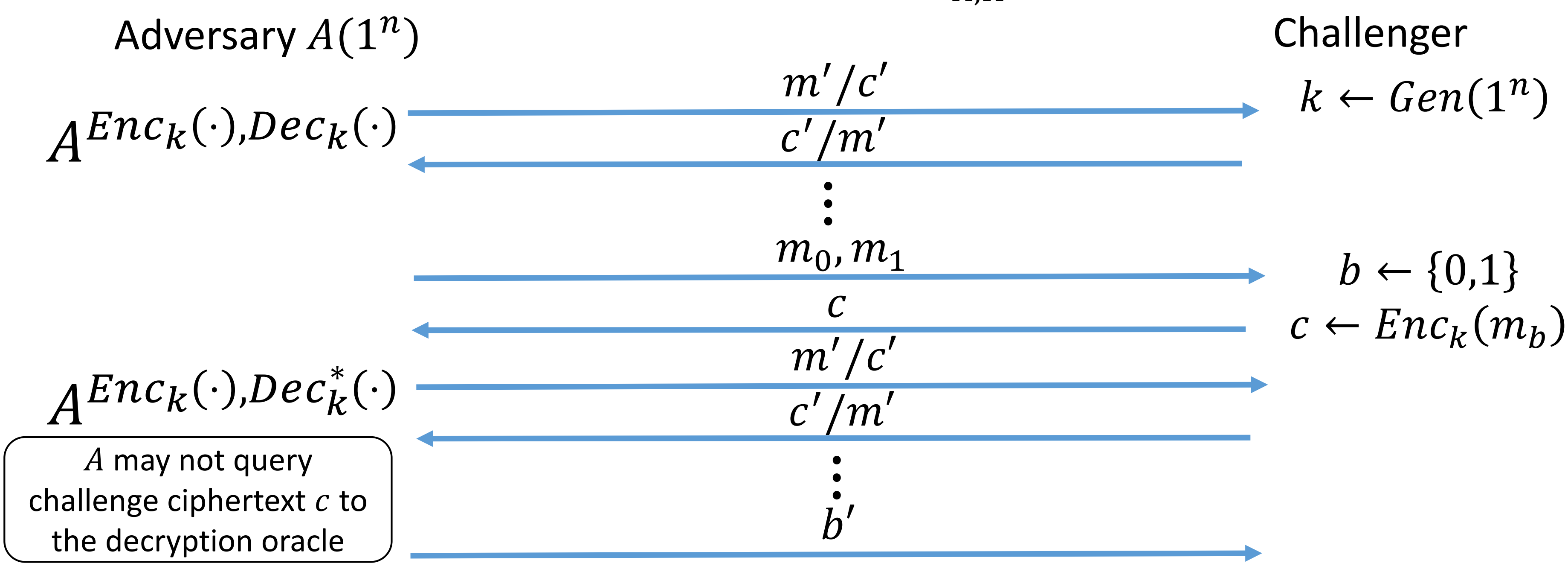
| Adversary $A(1^n)$ | | Challenger |
|---|---|---|
| $A^{Enc_k(\cdot), Dec_k(\cdot)}$ | $m'/c'$ → | $k \leftarrow Gen(1^n)$ |
| | ← $c'/m'$ | |
| | ⋮ | |
| | $m_0, m_1$ → | $b \leftarrow \{0,1\}$ |
| | ← $c$ | $c \leftarrow Enc_k(m_b)$ |
| $A^{Enc_k(\cdot), Dec_k(\cdot)}$ | $m'/c'$ → | |
| | ← $c'/m'$ | |
| | ⋮ | |
| | $b'$ → | |

$PrivK^{cca}_{A,\Pi}(n) = 1$ if $b' = b$ and $PrivK^{cca}_{A,\Pi}(n) = 0$ if $b' \neq b$.

# CCA Security

Consider a private-key encryption scheme $\Pi = (Gen, Enc, Dec)$, any adversary $A$, and any value $n$ for the security parameter.

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




$PrivK^{cca}_{A,\Pi}(n) = 1$ if $b' = b$ and $PrivK^{cca}_{A,\Pi}(n) = 0$ if $b' \neq b$.

# CCA Security

The CCA Indistinguishability Experiment $PrivK^{cca}{}_{A,\Pi}(n)$:

1. A key $k$ is generated by running $Gen(1^n)$.
2. The adversary $A$ is given input $1^n$ and oracle access to $Enc_k(\cdot)$ and $Dec_k(\cdot)$, and outputs a pair of messages $m_0, m_1$ of the same length.
3. A random bit $b \leftarrow \{0,1\}$ is chosen, and then a challenge ciphertext $c \leftarrow Enc_k(m_b)$ is computed and given to $A$.
4. The adversary $A$ continues to have oracle access to $Enc_k(\cdot)$ and $Dec_k(\cdot)$, but is not allowed to query the latter on the challenge ciphertext itself. Eventually, $A$ outputs a bit $b'$.
5. The output of the experiment is defined to be $1$ if $b' = b$, and $0$ otherwise.

# CCA Security

A private-key encryption scheme $\Pi = (Gen, Enc, Dec)$ has indistinguishable encryptions under a chosen-ciphertext attack if for all ppt adversaries $A$ there exists a negligible function $negl$ such that

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

where the probability is taken over the random coins used by $A$, as well as the random coins used in the experiment.

# Authenticated Encryption

The unforgeable encryption experiment $EncForge_{A,\Pi}(n)$:

1. Run $Gen(1^n)$ to obtain key $k$.
2. The adversary $A$ is given input $1^n$ and access to an encryption oracle $Enc_k(\cdot)$. The adversary outputs a ciphertext $c$.
3. Let $m \coloneqq Dec_k(c)$, and let $Q$ denote the set of all queries that $A$ asked its encryption oracle. The output of the experiment is $1$ if and only if (1) $m \neq \perp$ and (2) $m \notin Q$.

# Authenticated Encryption

Definition: A private-key encryption scheme $\Pi$ is unforgeable if for all ppt adversaries $A$, there is a negligible funcion $neg$ such that:

$$\Pr\left[EncForge_{A,\Pi}(n) = 1\right] \leq neg(n).$$

Definition: A private-key encryption scheme is an authenticated encryption scheme if it is CCA-secure and unforgeable.

# Generic Constructions

# Encrypt-and-authenticate

Encryption and message authentication are computed independently in parallel.

$$c \leftarrow Enc_{k_E}(m) \qquad t \leftarrow Mac_{k_M}(m)$$

$$\langle c, t \rangle$$

Is this secure?

# Encrypt-and-authenticate

Encryption and message authentication are computed independently in parallel.

$$c \leftarrow Enc_{k_E}(m) \qquad t \leftarrow Mac_{k_M}(m)$$

$$\langle c, t \rangle$$

Is this secure? NO! Tag can leak info on $m$

# Authenticate-then-encrypt

Here a MAC tag $t$ is first computed, and then the message and tag are encrypted together.

$$t \leftarrow Mac_{k_M}(m) \qquad c \leftarrow Enc_{k_E}(m||t)$$

$c$ is sent

Is this secure?

# Authenticate-then-encrypt

Here a MAC tag $t$ is first computed, and then the message and tag are encrypted together.

$$t \leftarrow Mac_{k_M}(m) \qquad c \leftarrow Enc_{k_E}(m||t)$$

$c$ is sent

Is this secure? NO! Encryption scheme may not be CCA-secure.

# Encrypt-then-authenticate

The message $m$ is first encrypted and then a MAC tag is computed over the result

$$c \leftarrow Enc_{k_E}(m) \quad t \leftarrow Mac_{k_M}(c)$$

$$\langle c, t \rangle$$

Is this secure?

# Encrypt-then-authenticate

The message $m$ is first encrypted and then a MAC tag is computed over the result

$$c \leftarrow Enc_{k_E}(m) \quad t \leftarrow Mac_{k_M}(c)$$

$$\langle c, t \rangle$$

Is this secure? YES! As long as the MAC is strongly secure.