Cryptology

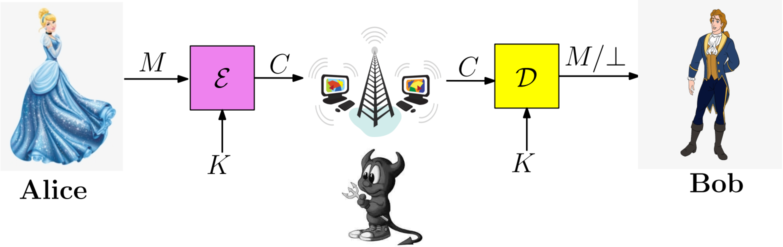
Sabyasachi Karati

Assistant Professor
Cryptology and Security Research Unit (C.S.R.U)
R. C. Bose Centre for Cryptology and Security
Indian Statistical Institute (ISI)
Kolkata, India

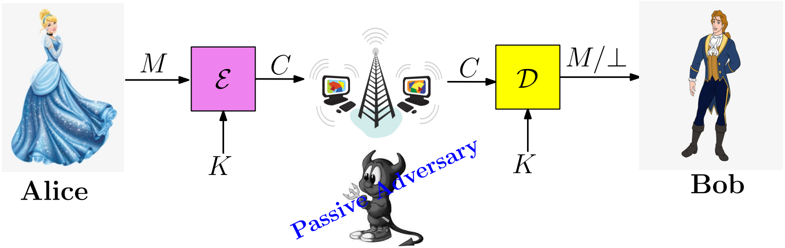
Lecture 09

Message Authentication Code

Secure Communication



Secure Communication



Alice

M

$\mathcal{E}$

K

C

Passive Adversary

C

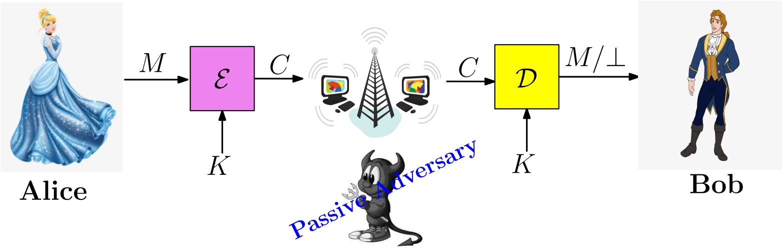
$\mathcal{D}$

K

$M/\perp$

Bob

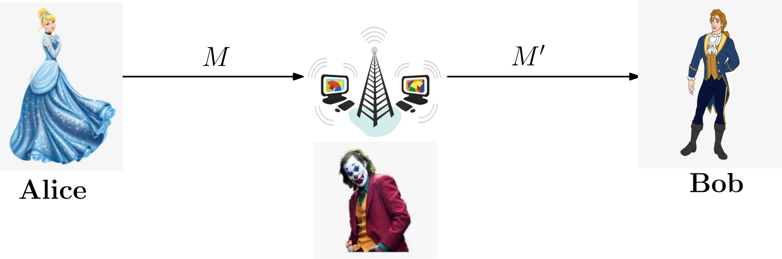
Secure Communication



Attack Models

1. Passive Adversary
2. Adversary must not learn anything about the message.

Message Integrity (or Message Authentication)



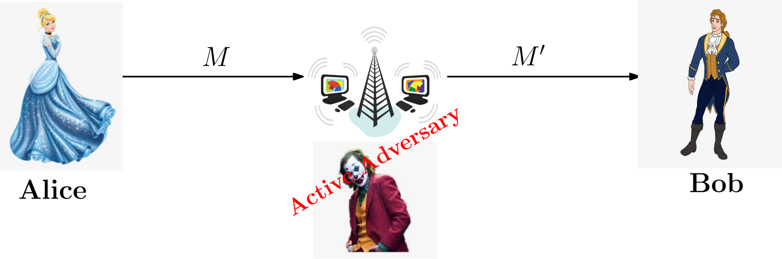
M

M'

Alice

Bob

Message Integrity (or Message Authentication)



M

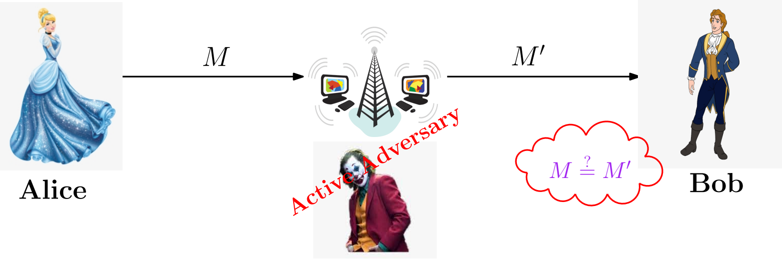
M'

Alice

Active Adversary

Bob

Message Integrity (or Message Authentication)



M

M'

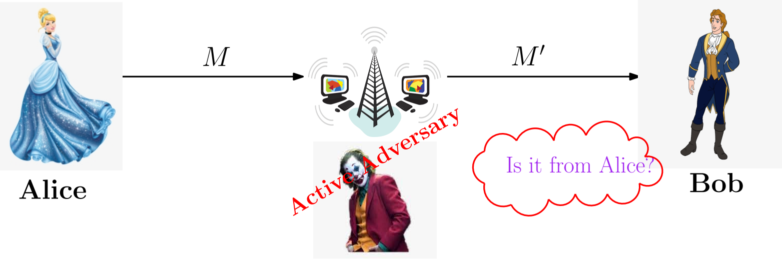
Active Adversary

$M \overset{?}{=} M'$

Alice

Bob

Message Integrity (or Message Authentication)



Alice

M

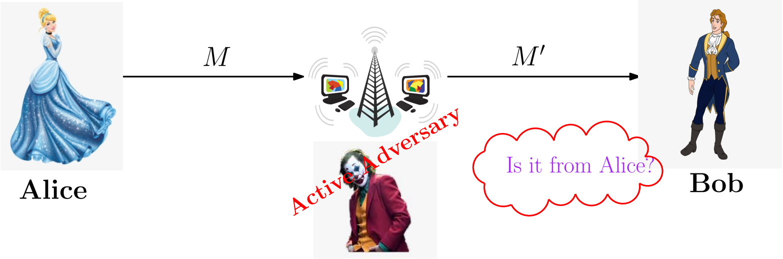
Active Adversary

M'

Is it from Alice?

Bob

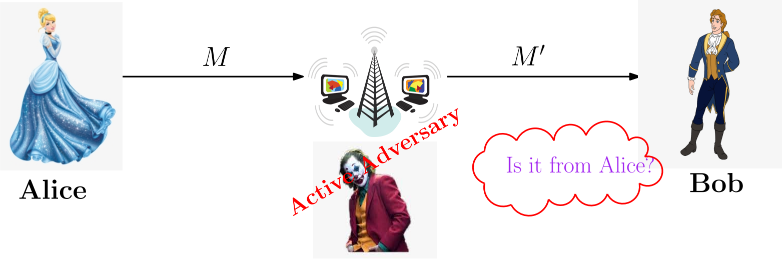
Message Integrity (or Message Authentication)



Attack Models

1. Active Adversary
2. Message may be public.

Message Integrity (or Message Authentication)



Attack Models

1. Active Adversary
2. Message may be public.

Question

How to provide Message Integrity?

Encryption Does Not Provide Integrity?

A note on Encryption

- Encryption does not (in general) provide any integrity.

Encryption Does Not Provide Integrity?

A note on Encryption

- Encryption does not (in general) provide any integrity.
- Encryption should never be used with the intent of achieving message authentication unless it is specifically designed with that purpose in mind (Authenticated encryption).

Encryption Does Not Provide Integrity?

A note on Encryption

- Encryption does not (in general) provide any integrity.
- Encryption should never be used with the intent of achieving message authentication unless it is specifically designed with that purpose in mind (Authenticated encryption).

A Incorrect reasoning

- Ciphertext hides the message.
- An adversary cannot modify an encrypted message in any meaningful way.
- All the encryption schemes that we have seen thus far do not provide message integrity.

Encryption Does Not Provide Integrity?

Stream Cipher

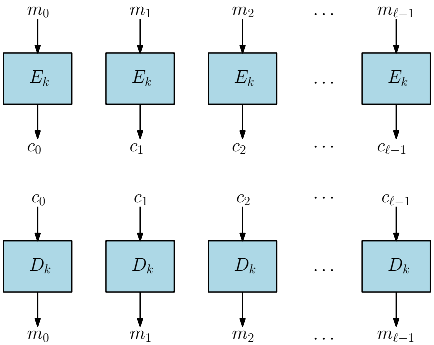
- We have already seen that we can change a part of a message in a meaningful way:

Encryption Does Not Provide Integrity?

Stream Cipher

- We have already seen that we can change a part of a message in a meaningful way:
  - If we know the text, and
  - Its position.

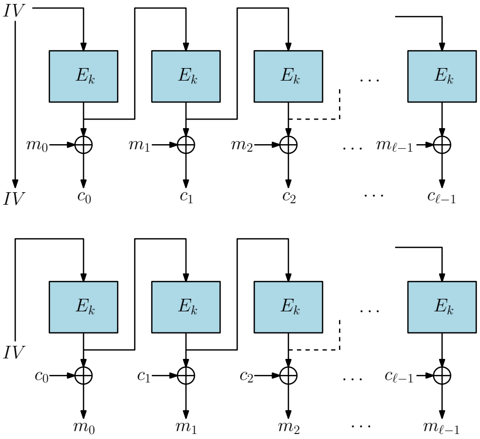
Encryption Does Not Provide Integrity?



Block Cipher (ECB mode)

- Changing one bit in cipher text means changing only one data block.

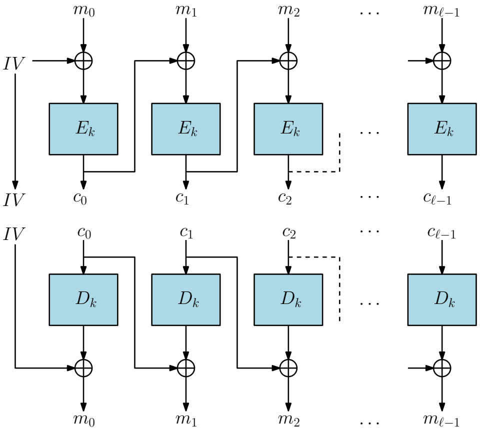
Encryption Does Not Provide Integrity?



Block Cipher (OFB mode)

- Similar to stream cipher.

Encryption Does Not Provide Integrity?



Block Cipher (CBC mode)

- Change in the IV of cipher text only changes the first data block.

How to provide Message Integrity?

Requirement

Providing message integrity between two communicating parties requires that the sending party has a secret key unknown to the adversary. Because adversary

- knows the algorithm,
- knows the message, and
- need nothing extra to compute a tag.

How to provide Message Integrity?

Requirement

Providing message integrity between two communicating parties requires that the sending party has a secret key unknown to the adversary. Because adversary

- knows the algorithm,
- knows the message, and
- need nothing extra to compute a tag.

Keyless Integrity Mechanism

- Often used in communication designed not for security
- CRC32 in Ethernet,
- 16-bit checksum in TCP.
- These keyless integrity mechanisms are designed to detect random transmission errors, not malicious errors.

Message Authentication Code

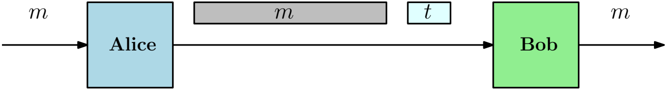
Message Authentication Code

A MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$ is a pair of efficient algorithms, $\mathcal{S}$ and $\mathcal{V}$, where $\mathcal{S}$ is called a signing algorithm and $\mathcal{V}$ is called a verification algorithm. Algorithm $\mathcal{S}$ is used to generate tags and algorithm $\mathcal{V}$ is used to verify tags.

Message Authentication Code

Message Authentication Code

A MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$ is a pair of efficient algorithms, $\mathcal{S}$ and $\mathcal{V}$, where $\mathcal{S}$ is called a signing algorithm and $\mathcal{V}$ is called a verification algorithm. Algorithm $\mathcal{S}$ is used to generate tags and algorithm $\mathcal{V}$ is used to verify tags.



Generate tag t

$t \xleftarrow{R} \mathcal{S}(k, m)$

Verify Message-tag pair (m, t)

$\mathcal{V}(k, m, t) \overset{?}{=} \texttt{accept}$

Message Authentication Code

We say that a MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$ is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$.

- $\mathcal{K}$ is a finite key space
- $\mathcal{M}$ is a finite message space
- $\mathcal{T}$ is a finite tag space,

Message Authentication Code

Message Authentication Code

We say that a MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$ is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$.

- $\mathcal{K}$ is a finite key space
- $\mathcal{M}$ is a finite message space
- $\mathcal{T}$ is a finite tag space,
- $\mathcal{S}$ is a probabilistic algorithm that is invoked as $t \xleftarrow{R} \mathcal{S}(k, m)$, where k is a key, m is a message, and the output t is called a tag.

Message Authentication Code

Message Authentication Code

We say that a MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$ is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$.

- $\mathcal{K}$ is a finite key space
- $\mathcal{M}$ is a finite message space
- $\mathcal{T}$ is a finite tag space,
- $\mathcal{S}$ is a probabilistic algorithm that is invoked as $t \overset{R}{\longleftarrow} \mathcal{S}(k, m)$, where k is a key, m is a message, and the output t is called a tag.
- $\mathcal{V}$ is a deterministic algorithm that is invoked as $r \longleftarrow \mathcal{V}(k, m, t)$, where k is a key, m is a message, t is a tag, and the output r is either `accept` or `reject`.

Message Authentication Code

Message Authentication Code

We say that a MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$ is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$.

- $\mathcal{K}$ is a finite key space
- $\mathcal{M}$ is a finite message space
- $\mathcal{T}$ is a finite tag space,
- $\mathcal{S}$ is a probabilistic algorithm that is invoked as $t \overset{R}{\longleftarrow} \mathcal{S}(k, m)$, where k is a key, m is a message, and the output t is called a tag.
- $\mathcal{V}$ is a deterministic algorithm that is invoked as $r \longleftarrow \mathcal{V}(k, m, t)$, where k is a key, m is a message, t is a tag, and the output r is either `accept` or `reject`.
- Correctness: We require that tags generated by $\mathcal{S}$ are always accepted by $\mathcal{V}$; that is, the MAC must satisfy the following correctness property: for all keys k and all messages m,

$$\Pr[\mathcal{V}(k, m, \mathcal{S}(k, m)) = \texttt{accept}] = 1.$$

Message Authentication Code

Message Authentication Code

- Deterministic MAC
  - For a given key k, and a given message m, there is a unique valid tag for m under k.

Message Authentication Code

Message Authentication Code

- Deterministic MAC
  - For a given key k, and a given message m, there is a unique valid tag for m under k.
- Randomized MAC
  - For a given key k, and a given message m, the output of $\mathcal{S}(k,m)$ may be one of many possible valid tags.

Message Authentication Code

Message Authentication Code

- Deterministic MAC
  - For a given key k, and a given message m, there is a unique valid tag for m under k.
- Randomized MAC
  - For a given key k, and a given message m, the output of $\mathcal{S}(k,m)$ may be one of many possible valid tags.
  - Are not necessary to achieve security.

Message Authentication Code

Message Authentication Code

- Deterministic MAC
  - For a given key k, and a given message m, there is a unique valid tag for m under k.
- Randomized MAC
  - For a given key k, and a given message m, the output of $\mathcal{S}(k,m)$ may be one of many possible valid tags.
  - Are not necessary to achieve security.
  - Yield better efficiency/security trade-offs.

Security of Randomized MAC

Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$, defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

Security of Randomized MAC

Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$, defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

1. The challenger picks a random $k \xleftarrow{R} \mathcal{K}$.

Security of Randomized MAC

Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$, defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

1. The challenger picks a random $k \overset{R}{\longleftarrow} \mathcal{K}$.
2. $\mathcal{A}$ queries the challenger several times. For $i = 1, 2, \ldots$, the i-th signing query is a message $m_i \in \mathcal{M}$. Given m_i, the challenger computes a tag $t_i \overset{R}{\longleftarrow} \mathcal{S}(k, m_i)$, and then gives t_i to $\mathcal{A}$.

Security of Randomized MAC

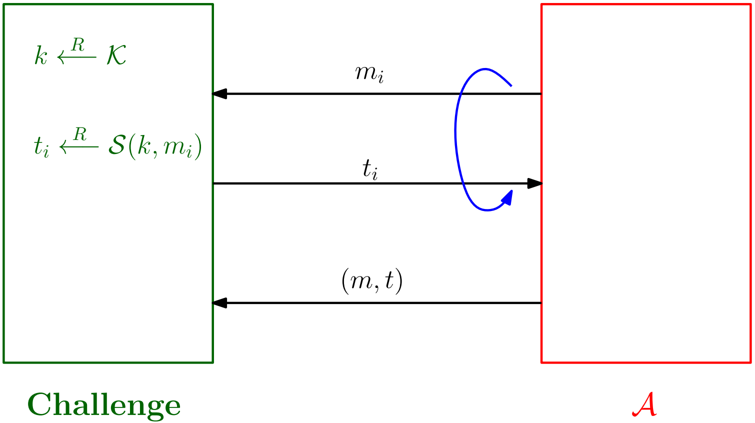
Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$, defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

1. The challenger picks a random $k \xleftarrow{R} \mathcal{K}$.
2. $\mathcal{A}$ queries the challenger several times. For $i = 1, 2, \ldots$, the i-th signing query is a message $m_i \in \mathcal{M}$. Given m_i, the challenger computes a tag $t_i \xleftarrow{R} \mathcal{S}(k, m_i)$, and then gives t_i to $\mathcal{A}$.
3. Eventually $\mathcal{A}$ outputs a candidate forgery pair $(m, t) \in \mathcal{M} \times \mathcal{T}$ that is not among the signed pairs, i.e.,

$$(m, t) \notin \{(m_1, t_1), (m_2, t_2), \ldots\}.$$

Security of Randomized MAC


$k \xleftarrow{R} \mathcal{K}$
$t_i \xleftarrow{R} \mathcal{S}(k, m_i)$
m_i
t_i
(m, t)
Challenge
$\mathcal{A}$

MAC Attcak Game

Security of Randomized MAC

Advantage of $\mathcal{A}$

- We say that $\mathcal{A}$ wins the above game if (m, t) is a valid pair under k (i.e., $\mathcal{V}(k, m, t) = \texttt{accept}$).

Security of Randomized MAC

Advantage of $\mathcal{A}$

- We say that $\mathcal{A}$ wins the above game if (m, t) is a valid pair under k (i.e., $\mathcal{V}(k, m, t) = \texttt{accept}$).
- In the adversary wins the Attack Game, the pair (m, t) that it sends to the challenger is called an existential forgery.

Security of Randomized MAC

Advantage of $\mathcal{A}$

- We say that $\mathcal{A}$ wins the above game if (m,t) is a valid pair under k (i.e., $\mathcal{V}(k,m,t) = \texttt{accept}$).
- In the adversary wins the Attack Game, the pair (m,t) that it sends to the challenger is called an existential forgery.
- We define $\mathcal{A}$'s advantage with respect to $\mathcal{I}$, denoted $\mathsf{MACadv}[\mathcal{A},\mathcal{I}]$, as the probability that $\mathcal{A}$ wins the game.

Security of Randomized MAC

Advantage of $\mathcal{A}$

- We say that $\mathcal{A}$ wins the above game if (m, t) is a valid pair under k (i.e., $\mathcal{V}(k, m, t) = \texttt{accept}$).
- In the adversary wins the Attack Game, the pair (m, t) that it sends to the challenger is called an existential forgery.
- We define $\mathcal{A}$'s advantage with respect to $\mathcal{I}$, denoted $\mathsf{MACadv}[\mathcal{A}, \mathcal{I}]$, as the probability that $\mathcal{A}$ wins the game.
- We say that $\mathcal{A}$ is a Q-query MAC adversary if $\mathcal{A}$ issues at most Q signing queries.

Security of Randomized MAC

Advantage of $\mathcal{A}$

- We say that $\mathcal{A}$ wins the above game if (m,t) is a valid pair under k (i.e., $\mathcal{V}(k,m,t) = \texttt{accept}$).
- In the adversary wins the Attack Game, the pair (m,t) that it sends to the challenger is called an existential forgery.
- We define $\mathcal{A}$'s advantage with respect to $\mathcal{I}$, denoted $\mathsf{MACadv}[\mathcal{A},\mathcal{I}]$, as the probability that $\mathcal{A}$ wins the game.
- We say that $\mathcal{A}$ is a Q-query MAC adversary if $\mathcal{A}$ issues at most Q signing queries.

Secure MAC

A MAC system $\mathcal{I}$ is secure if for all efficient adversaries $\mathcal{A}$, the value $\mathsf{MACadv}[\mathcal{A},\mathcal{I}]$ is negligible.

Security of Deterministic MAC

Attack Game

- $\mathcal{A}$ wins the Attack Game if it produce a valid message-tag pair (m,t) for some new message
$$m \notin \{m_1, m_2, \ldots\}.$$

Security of Deterministic MAC

Attack Game

- $\mathcal{A}$ wins the Attack Game if it produce a valid message-tag pair (m,t) for some new message
$$m \notin \{m_1, m_2, \ldots\}.$$
- Security in this case just means that S is unpredictable.

MAC Security with Verification Queries

Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$, defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

MAC Security with Verification Queries

Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$, defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

1. The challenger picks a random $k \xleftarrow{R} \mathcal{K}$.

MAC Security with Verification Queries

Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$, defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

1. The challenger picks a random $k \xleftarrow{R} \mathcal{K}$.
2. Signing query: For $i = 1, 2, \ldots$, the i-th signing query consists of a message $m_i \in \mathcal{M}$. Given m_i, the challenger computes a tag $t_i \xleftarrow{R} \mathcal{S}(k, m_i)$, and then gives t_i to $\mathcal{A}$.

MAC Security with Verification Queries

Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathcal{I} = (\mathcal{S}, \mathcal{V})$, defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

1. The challenger picks a random $k \xleftarrow{R} \mathcal{K}$.
2. Signing query: For $i = 1, 2, \ldots$, the i-th signing query consists of a message $m_i \in \mathcal{M}$. Given m_i, the challenger computes a tag $t_i \xleftarrow{R} \mathcal{S}(k, m_i)$, and then gives t_i to $\mathcal{A}$.
3. Verification query: For $j = 1, 2, \ldots$, the j-th verification query consists of a message-tag pair $(\hat{m}_j, \hat{t}_j) \in \mathcal{M} \times \mathcal{T}$ that is not among the previously signed pairs, i.e.,

   $$(\hat{m}_j, \hat{t}_j) \notin \{(m_1, t_1), (m_2, t_2), \ldots\}.$$

   The challenger responds to $\mathcal{A}$ with $\mathcal{V}(k, \hat{m}, \hat{t})$.

MAC Security with Verification Queries

Advantage of $\mathcal{A}$

- We say that $\mathcal{A}$ wins the above game if the challenger ever responds to a verification query with `accept`.

MAC Security with Verification Queries

Advantage of $\mathcal{A}$

- We say that $\mathcal{A}$ wins the above game if the challenger ever responds to a verification query with `accept`.
- We define $\mathcal{A}$'s advantage with respect to $\mathcal{I}$, denoted $\mathsf{MAC^{vq}adv}[\mathcal{A}, \mathcal{I}]$, as the probability that A wins the game.

MAC Security with Verification Queries

Advantage of $\mathcal{A}$

- We say that $\mathcal{A}$ wins the above game if the challenger ever responds to a verification query with `accept`.
- We define $\mathcal{A}$'s advantage with respect to $\mathcal{I}$, denoted $\mathsf{MAC^{vq}adv}[\mathcal{A}, \mathcal{I}]$, as the probability that A wins the game.
- This extra power of verification query does not help the adversary.

MAC Security with Verification Queries

Advantage of $\mathcal{A}$

- We say that $\mathcal{A}$ wins the above game if the challenger ever responds to a verification query with `accept`.
- We define $\mathcal{A}$'s advantage with respect to $\mathcal{I}$, denoted $\mathsf{MAC^{vq}adv}[\mathcal{A}, \mathcal{I}]$, as the probability that A wins the game.
- This extra power of verification query does not help the adversary.

Secure MAC

A MAC system $\mathcal{I}$ is secure if for all efficient adversaries $\mathcal{A}$, the value $\mathsf{MAC^{vq}adv}[\mathcal{A}, \mathcal{I}]$ is negligible.

MAC Security with Verification Queries

Theorem

If $\mathcal{I}$ is a secure MAC system, then it is also secure in the presence of verification queries.

MAC Security with Verification Queries

Theorem

If $\mathcal{I}$ is a secure MAC system, then it is also secure in the presence of verification queries.
In particular, for every MAC adversary $\mathcal{A}$ that attacks $\mathcal{I}$ as in MAC Attack Game with verification query, and which makes at most Q_v verification queries and at most Q_s signing queries, there exists a Q_s-query MAC adversary $\mathcal{B}$ that attacks $\mathcal{I}$ as in MAC Attack Game without verification query, where $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine, such that

$$\mathsf{MAC}^{\mathsf{vq}}\mathsf{adv}[\mathcal{A},\mathcal{I}] \leqslant \mathsf{MACadv}[\mathcal{B},\mathcal{I}] \cdot Q_v.$$

MAC Security with Verification Queries

Proof

- Adversary $\mathcal{B}$ plays the role of challenger to $\mathcal{A}$ in Attack Game with verification query.
- Adversary $\mathcal{B}$ plays the role of adversary in Attack Game without verification query.

MAC Security with Verification Queries

Proof

- Adversary $\mathcal{B}$ plays the role of challenger to $\mathcal{A}$ in Attack Game with verification query.
- Adversary $\mathcal{B}$ plays the role of adversary in Attack Game without verification query.

Construction of $\mathcal{B}$

Initialization:

$$\omega \xleftarrow{R} \{1, \ldots, Q_v\}.$$

MAC Security with Verification Queries

Proof

- Adversary $\mathcal{B}$ plays the role of challenger to $\mathcal{A}$ in Attack Game with verification query.
- Adversary $\mathcal{B}$ plays the role of adversary in Attack Game without verification query.

Construction of $\mathcal{B}$

```
Initialization:
    $\omega \overset{R}{\longleftarrow} \{1, \ldots, Q_v\}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    forward $m_i$ to the MAC challenger, obtaining the tag $t_i$
    send $t_i$ to $\mathcal{A}$
```

MAC Security with Verification Queries

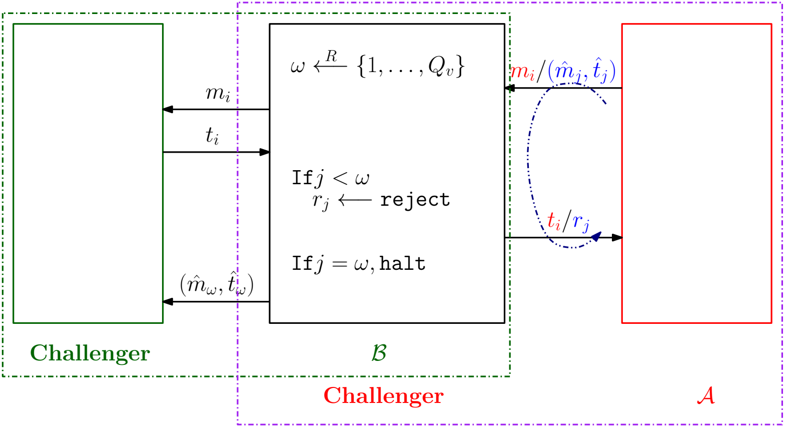
Proof

- Adversary $\mathcal{B}$ plays the role of challenger to $\mathcal{A}$ in Attack Game with verification query.
- Adversary $\mathcal{B}$ plays the role of adversary in Attack Game without verification query.

Construction of $\mathcal{B}$

```
Initialization:
    $\omega \xleftarrow{R} \{1, \ldots, Q_v\}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    forward $m_i$ to the MAC challenger, obtaining the tag $t_i$
    send $t_i$ to $\mathcal{A}$
Upon receiving a verification query $(\hat{m}_j, \hat{t}_j) \in \mathcal{M} \times \mathcal{T}$ from $\mathcal{A}$ do:
    if $j = \omega$
        then output $(\hat{m}_j, \hat{t}_j)$ as a candidate forgery pair and halt
        else send reject to $\mathcal{A}$.
```

MAC Security with Verification Queries


$\omega \xleftarrow{R} \{1,\ldots,Q_v\}$
m_i
t_i
If $j < \omega$
$r_j \longleftarrow$ reject
If $j = \omega$, halt
$(\hat{m}_\omega, \hat{t}_\omega)$
$m_i/(\hat{m}_j, \hat{t}_j)$
t_i/r_j
Challenger
$\mathcal{B}$
Challenger
$\mathcal{A}$

MAC Security with Verification Queries

Game 0: MAC attack game with verification between challenger and $\mathcal{A}$

Initialization:

$$k \xleftarrow{R} \mathcal{K}.$$

MAC Security with Verification Queries

Game 0: MAC attack game with verification between challenger and $\mathcal{A}$

```
Initialization:
    $k \xleftarrow{R} \mathcal{K}.$
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    $t_i \xleftarrow{R} \mathcal{S}(k, m_i)$
    send $t_i$ to $\mathcal{A}$
```

MAC Security with Verification Queries

Game 0: MAC attack game with verification between challenger and $\mathcal{A}$

```
Initialization:
    $k \xleftarrow{R} \mathcal{K}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    $t_i \xleftarrow{R} \mathcal{S}(k, m_i)$
    send $t_i$ to $\mathcal{A}$
Upon receiving a verification query $(\hat{m}_j, \hat{t}_j) \in \mathcal{M} \times \mathcal{T}$ from $\mathcal{A}$ do:
    $r_j \longleftarrow \mathcal{V}(k, \hat{m}_j, \hat{t}_j)$
    send $r$ to $\mathcal{A}$.
```

MAC Security with Verification Queries

Game 0: MAC attack game with verification between challenger and $\mathcal{A}$

```
Initialization:
    $k \xleftarrow{R} \mathcal{K}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    $t_i \xleftarrow{R} \mathcal{S}(k, m_i)$
    send $t_i$ to $\mathcal{A}$
Upon receiving a verification query $(\hat{m}_j, \hat{t}_j) \in \mathcal{M} \times \mathcal{T}$ from $\mathcal{A}$ do:
    $r_j \longleftarrow \mathcal{V}(k, \hat{m}_j, \hat{t}_j)$
    send $r$ to $\mathcal{A}$.
```

$\mathcal{A}$ wins

Let W_0 be the event that in Game 0, $r_j = \texttt{accept}$ for some j.

$$\Pr[W_0] = \mathsf{MAC}^{\mathsf{vq}}\mathsf{adv}[\mathcal{A}, \mathcal{I}].$$

MAC Security with Verification Queries

Game 1

Initialization:

$$k \xleftarrow{R} \mathcal{K}.$$

MAC Security with Verification Queries

Game 1

Initialization:

$k \xleftarrow{R} \mathcal{K}.$

Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:

$t_i \xleftarrow{R} \mathcal{S}(k, m_i)$

send t_i to $\mathcal{A}$

MAC Security with Verification Queries

Game 1

```
Initialization:
    $k \xleftarrow{R} \mathcal{K}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    $t_i \xleftarrow{R} \mathcal{S}(k, m_i)$
    send $t_i$ to $\mathcal{A}$
Upon receiving a verification query $(\hat{m}_j, \hat{t}_j) \in \mathcal{M} \times \mathcal{T}$ from $\mathcal{A}$ do:
    $r_j \longleftarrow \mathcal{V}(k, \hat{m}_j, \hat{t}_j)$
    send reject to $\mathcal{A}$.
```

MAC Security with Verification Queries

Game 1

```
Initialization:
    $k \xleftarrow{R} \mathcal{K}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    $t_i \xleftarrow{R} \mathcal{S}(k, m_i)$
    send $t_i$ to $\mathcal{A}$
Upon receiving a verification query $(\hat{m}_j, \hat{t}_j) \in \mathcal{M} \times \mathcal{T}$ from $\mathcal{A}$ do:
    $r_j \longleftarrow \mathcal{V}(k, \hat{m}_j, \hat{t}_j)$
    send reject to $\mathcal{A}$.
```

$\mathcal{A}$ wins

- Let W_1 be the event that in Game 1, $r_j = \texttt{accept}$ for some j.

MAC Security with Verification Queries

Game 1

```
Initialization:
    $k \overset{R}{\longleftarrow} \mathcal{K}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    $t_i \overset{R}{\longleftarrow} \mathcal{S}(k, m_i)$
    send $t_i$ to $\mathcal{A}$
Upon receiving a verification query $(\hat{m}_j, \hat{t}_j) \in \mathcal{M} \times \mathcal{T}$ from $\mathcal{A}$ do:
    $r_j \longleftarrow \mathcal{V}(k, \hat{m}_j, \hat{t}_j)$
    send reject to $\mathcal{A}$.
```

$\mathcal{A}$ wins

- Let W_1 be the event that in Game 1, $r_j = \texttt{accept}$ for some j.
- Even though the challenger does not notify $\mathcal{A}$ that W_1 occurs, both Games 0 and 1 proceed identically until this event happens.

MAC Security with Verification Queries

Game 1

```
Initialization:
    $k \xleftarrow{R} \mathcal{K}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    $t_i \xleftarrow{R} \mathcal{S}(k, m_i)$
    send $t_i$ to $\mathcal{A}$
Upon receiving a verification query $(\hat{m}_j, \hat{t}_j) \in \mathcal{M} \times \mathcal{T}$ from $\mathcal{A}$ do:
    $r_j \longleftarrow \mathcal{V}(k, \hat{m}_j, \hat{t}_j)$
    send reject to $\mathcal{A}$.
```

$\mathcal{A}$ wins

- Let W_1 be the event that in Game 1, $r_j =$ `accept` for some j.
- Even though the challenger does not notify $\mathcal{A}$ that W_1 occurs, both Games 0 and 1 proceed identically until this event happens.
- Therefore, events W_0 and W_1 are really the same.

$$\Pr[W_0] = \Pr[W_1].$$

MAC Security with Verification Queries

Game 2

Initialization:

$$k \xleftarrow{R} \mathcal{K}.$$

MAC Security with Verification Queries

Game 2

Initialization:

$k \xleftarrow{R} \mathcal{K}.$

$\omega \xleftarrow{R} \{1,\ldots,Q_v\}.$

MAC Security with Verification Queries

Game 2

```
Initialization:
    $k \xleftarrow{R} \mathcal{K}$.
    $\omega \xleftarrow{R} \{1, \ldots, Q_v\}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    $t_i \xleftarrow{R} \mathcal{S}(k, m_i)$
    send $t_i$ to $\mathcal{A}$
```

MAC Security with Verification Queries

Game 2

```
Initialization:
    $k \xleftarrow{R} \mathcal{K}$.
    $\omega \xleftarrow{R} \{1,\ldots,Q_v\}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    $t_i \xleftarrow{R} \mathcal{S}(k,m_i)$
    send $t_i$ to $\mathcal{A}$
Upon receiving a verification query $(\hat{m}_j,\hat{t}_j) \in \mathcal{M}\times\mathcal{T}$ from $\mathcal{A}$ do:
    $r_j \longleftarrow \mathcal{V}(k,\hat{m}_j,\hat{t}_j)$
    send reject to $\mathcal{A}$.
```

MAC Security with Verification Queries

Game 2

```
Initialization:
    $k \xleftarrow{R} \mathcal{K}$.
    $\omega \xleftarrow{R} \{1,\ldots,Q_v\}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    $t_i \xleftarrow{R} \mathcal{S}(k,m_i)$
    send $t_i$ to $\mathcal{A}$
Upon receiving a verification query $(\hat{m}_j,\hat{t}_j) \in \mathcal{M}\times\mathcal{T}$ from $\mathcal{A}$ do:
    $r_j \longleftarrow \mathcal{V}(k,\hat{m}_j,\hat{t}_j)$
    send reject to $\mathcal{A}$.
```

$\mathcal{A}$ wins

- Let W_2 be the event that in Game 2, $r_\omega = \texttt{accept}$. $\Pr[W_2] \geqslant \Pr[W_1]/Q_v$.

MAC Security with Verification Queries

Game 2

```
Initialization:
    $k \xleftarrow{R} \mathcal{K}$.
    $\omega \xleftarrow{R} \{1, \ldots, Q_v\}$.
Upon receiving a signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$ do:
    $t_i \xleftarrow{R} \mathcal{S}(k, m_i)$
    send $t_i$ to $\mathcal{A}$
Upon receiving a verification query $(\hat{m}_j, \hat{t}_j) \in \mathcal{M} \times \mathcal{T}$ from $\mathcal{A}$ do:
    $r_j \longleftarrow \mathcal{V}(k, \hat{m}_j, \hat{t}_j)$
    send reject to $\mathcal{A}$.
```

$\mathcal{A}$ wins

- Let W_2 be the event that in Game 2, $r_\omega = \texttt{accept}$. $\Pr[W_2] \geqslant \Pr[W_1]/Q_v$.
- By construction,

$$\mathsf{MACadv}[\mathcal{B}, \mathcal{I}] = \Pr[W_2] \geqslant \Pr[W_1]/Q_v = \Pr[W_0]/Q_v = \mathsf{MAC}^{\mathsf{vq}}\mathsf{adv}[\mathcal{A}, \mathcal{I}]/Q_v.$$

Constructing MACs from PRFs

A MAC system $\mathcal{I}$ (fixed length)

- Let F be a PRF defined over $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$.

Constructing MACs from PRFs

A MAC system $\mathcal{I}$ (fixed length)

- Let F be a PRF defined over $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$.
- Key space: $\mathcal{K}$, message space: $\mathcal{X}$, and tag space: $\mathcal{Y}$.

Constructing MACs from PRFs

A MAC system $\mathcal{I}$ (fixed length)

- Let F be a PRF defined over $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$.
- Key space: $\mathcal{K}$, message space: $\mathcal{X}$, and tag space: $\mathcal{Y}$.
- $\mathcal{S}(k,m)$: For a $k \in \mathcal{K}$ and $m \in \mathcal{X}$,

$$\mathcal{S}(k,m) := F(k,m).$$

Constructing MACs from PRFs

A MAC system $\mathcal{I}$ (fixed length)

- Let F be a PRF defined over $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$.
- Key space: $\mathcal{K}$, message space: $\mathcal{X}$, and tag space: $\mathcal{Y}$.
- $\mathcal{S}(k,m)$: For a $k \in \mathcal{K}$ and $m \in \mathcal{X}$,

$$\mathcal{S}(k,m) := F(k,m).$$

- $\mathcal{V}(k,m,t)$: For a $k \in \mathcal{K}$, $m \in \mathcal{X}$ and $t \in \mathcal{Y}$

$$\mathcal{V}(k,m,t) := \begin{cases} \texttt{accept}, & \text{if } F(k,m) = t \\ \texttt{reject}, & \text{otherwise} \end{cases} .$$

Constructing MACs from PRFs

A MAC system $\mathcal{I}$ (fixed length)

- Let F be a PRF defined over $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$.
- Key space: $\mathcal{K}$, message space: $\mathcal{X}$, and tag space: $\mathcal{Y}$.
- $\mathcal{S}(k,m)$: For a $k \in \mathcal{K}$ and $m \in \mathcal{X}$,

$$\mathcal{S}(k,m) := F(k,m).$$

- $\mathcal{V}(k,m,t)$: For a $k \in \mathcal{K}$, $m \in \mathcal{X}$ and $t \in \mathcal{Y}$

$$\mathcal{V}(k,m,t) := \begin{cases} \texttt{accept}, & \text{if } F(k,m) = t \\ \texttt{reject}, & \text{otherwise} \end{cases} .$$

Security

- Any PRF with a large (i.e., super-poly) output space is unpredictable.

Constructing MACs from PRFs

A MAC system $\mathcal{I}$ (fixed length)

- Let F be a PRF defined over $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$.
- Key space: $\mathcal{K}$, message space: $\mathcal{X}$, and tag space: $\mathcal{Y}$.
- $\mathcal{S}(k, m)$: For a $k \in \mathcal{K}$ and $m \in \mathcal{X}$,

$$\mathcal{S}(k, m) := F(k, m).$$

- $\mathcal{V}(k, m, t)$: For a $k \in \mathcal{K}$, $m \in \mathcal{X}$ and $t \in \mathcal{Y}$

$$\mathcal{V}(k, m, t) := \begin{cases} \texttt{accept}, & \text{if } F(k, m) = t \\ \texttt{reject}, & \text{otherwise} \end{cases} .$$

Security

- Any PRF with a large (i.e., super-poly) output space is unpredictable.
- Therefore, the above construction yields a secure MAC.

Constructing MACs from PRFs

Theorem

Let F be a secure PRF defined over $(\mathcal{K},\mathcal{X},\mathcal{Y})$, where $|\mathcal{Y}|$ is super-poly. Then the deterministic MAC system $\mathcal{I}$ derived from F is a secure MAC.

Constructing MACs from PRFs

Theorem

Let F be a secure PRF defined over $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$, where $|\mathcal{Y}|$ is super-poly. Then the deterministic MAC system $\mathcal{I}$ derived from F is a secure MAC.
In particular, for every Q-query MAC adversary $\mathcal{A}$ that attacks $\mathcal{I}$ as in MAC Attack Game (without verification query), there exists a $(Q+1)$-query PRF adversary $\mathcal{B}$ that attacks F as in PRF Indistinguishability, where $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine, such that

$$\mathsf{MACadv}[\mathcal{A}, \mathcal{I}] \leqslant \mathsf{PRFadv}[\mathcal{B}, F] + \frac{1}{|\mathcal{Y}|}.$$

Domain Extension to Variable-Length Messages

- The constructions will be presented here are not very efficient and is unlikely to be used in practice.

Domain Extension to Variable-Length Messages

- The constructions will be presented here are not very efficient and is unlikely to be used in practice.
- Far more efficient constructions of secure MACs are known.

Domain Extension to Variable-Length Messages

- The constructions will be presented here are not very efficient and is unlikely to be used in practice.
- Far more efficient constructions of secure MACs are known.
- We study it for its simplicity and generality.

Domain Extension to Variable-Length Messages

- The constructions will be presented here are not very efficient and is unlikely to be used in practice.
- Far more efficient constructions of secure MACs are known.
- We study it for its simplicity and generality.

Assumptions

- $\mathcal{I} = (\mathcal{S}, \mathcal{V})$: A secure fixed-length MAC for messages of length n.

Domain Extension to Variable-Length Messages

- The constructions will be presented here are not very efficient and is unlikely to be used in practice.
- Far more efficient constructions of secure MACs are known.
- We study it for its simplicity and generality.

Assumptions

- $\mathcal{I} = (\mathcal{S}, \mathcal{V})$: A secure fixed-length MAC for messages of length n.
- We parse the message m to be authenticated as a sequence of blocks $m := (m_1, \ldots, m_d)$, where $|m_i| = n$ for all i.

Domain Extension to Variable-Length Messages

Construction 1

- Compute the tag as $t \longleftarrow \mathcal{S}\left(k, \bigoplus_{i=1}^{d} m_i\right)$.

Domain Extension to Variable-Length Messages

Construction 1

- Compute the tag as $t \longleftarrow \mathcal{S}\left(k, \bigoplus_{i=1}^{d} m_i\right)$.
- Advantage:
    - Easy and efficient to compute.

Domain Extension to Variable-Length Messages

Construction 1

- Compute the tag as $t \longleftarrow \mathcal{S}\left(k, \bigoplus_{i=1}^{d} m_i\right)$.
- Advantage:
  - Easy and efficient to compute.
- Attack:
  - an adversary can forge a valid tag on a new message by changing the original message so that the XOR of the blocks does not change.

Domain Extension to Variable-Length Messages

Construction 2

- Compute the tag as $t := (t_1, t_2, \ldots, t_d)$, where $t_i \longleftarrow \mathcal{S}(k, m_i)$.

Domain Extension to Variable-Length Messages

Construction 2

- Compute the tag as $t := (t_1, t_2, \ldots, t_d)$, where $t_i \longleftarrow \mathcal{S}(k, m_i)$.
- Advantage:
  - Prevents an adversary from sending any previously unauthenticated block without being detected.

Domain Extension to Variable-Length Messages

Construction 2

- Compute the tag as $t := (t_1, t_2, \ldots, t_d)$, where $t_i \longleftarrow \mathcal{S}(k, m_i)$.
- Advantage:
  - Prevents an adversary from sending any previously unauthenticated block without being detected.
- Attack:
  - Block reordering attack possible.
  - If $(m,t) := ((m_1, m_2), (t_1, t_2))$ is a valid message-tag pair then so is $(m', t') := ((m_2, m_1), (t_2, t_1))$.

Domain Extension to Variable-Length Messages

Construction 3

- Compute the tag as $t := (t_1, t_2, \ldots, t_d)$, where $t_i \longleftarrow \mathcal{S}(k, i \| m_i)$.

Domain Extension to Variable-Length Messages

Construction 3

- Compute the tag as $t := (t_1, t_2, \ldots, t_d)$, where $t_i \longleftarrow \mathcal{S}(k, i \| m_i)$.
- Note: Block length $|m_i|$ decreases.

Domain Extension to Variable-Length Messages

Construction 3

- Compute the tag as $t := (t_1, t_2, \ldots, t_d)$, where $t_i \longleftarrow \mathcal{S}(k, i \| m_i)$.
- Note: Block length $|m_i|$ decreases.
- Advantage:
  - Prevents block reordering attack.

Domain Extension to Variable-Length Messages

Construction 3

- Compute the tag as $t := (t_1, t_2, \ldots, t_d)$, where $t_i \longleftarrow \mathcal{S}(k, i \| m_i)$.
- Note: Block length $|m_i|$ decreases.
- Advantage:
    - Prevents block reordering attack.
- Attack:
    - Truncation attack possible.

Domain Extension to Variable-Length Messages

Construction 3

- Compute the tag as $t := (t_1, t_2, \ldots, t_d)$, where $t_i \longleftarrow \mathcal{S}(k, i \| m_i)$.
- Note: Block length $|m_i|$ decreases.
- Advantage:
  - Prevents block reordering attack.
- Attack:
  - Truncation attack possible.
    - The attacker simply drops blocks from the end of the message and also drops the corresponding blocks of the tag.

Domain Extension to Variable-Length Messages

Construction 3

- Compute the tag as $t := (t_1, t_2, \ldots, t_d)$, where $t_i \longleftarrow \mathcal{S}(k, i \| m_i)$.
- Note: Block length $|m_i|$ decreases.
- Advantage:
  - Prevents block reordering attack.
- Attack:
  - Truncation attack possible.
    - The attacker simply drops blocks from the end of the message and also drops the corresponding blocks of the tag.
  - Mix-and-match block attacks possible.

Domain Extension to Variable-Length Messages

Construction 3

- Compute the tag as $t := (t_1, t_2, \ldots, t_d)$, where $t_i \longleftarrow \mathcal{S}(k, i \| m_i)$.
- Note: Block length $|m_i|$ decreases.
- Advantage:
  - Prevents block reordering attack.
- Attack:
  - Truncation attack possible.
    - The attacker simply drops blocks from the end of the message and also drops the corresponding blocks of the tag.
  - Mix-and-match block attacks possible.
    - Adversary obtains the tags $t = (t_1, \ldots, t_d)$ and $t' = (t'_1, \ldots, t'_d)$ on the messages $m = (m_1, \ldots, m_d)$ and $m' = (m'_1, \ldots, m'_d)$, respectively.

Domain Extension to Variable-Length Messages

Construction 3

- Compute the tag as $t := (t_1, t_2, \ldots, t_d)$, where $t_i \longleftarrow \mathcal{S}(k, i \| m_i)$.
- Note: Block length $|m_i|$ decreases.
- Advantage:
  - Prevents block reordering attack.
- Attack:
  - Truncation attack possible.
    - The attacker simply drops blocks from the end of the message and also drops the corresponding blocks of the tag.
  - Mix-and-match block attacks possible.
    - Adversary obtains the tags $t = (t_1, \ldots, t_d)$ and $t' = (t'_1, \ldots, t'_d)$ on the messages $m = (m_1, \ldots, m_d)$ and $m' = (m'_1, \ldots, m'_d)$, respectively.
    - Constructs message-tag pair $(\hat{m}, \hat{t}) := ((m_1, m'_2, m_3, m'_4, \ldots), (t_1, t'_2, t_3, t'_4, \ldots))$.

Domain Extension to Variable-Length Messages

- Can be prevented by including a random message identifier along with each block to prevent blocks from different messages from being combined.

Domain Extension to Variable-Length Messages

- Can be prevented by including a random message identifier along with each block to prevent blocks from different messages from being combined.
- Below, we construct a fixed-length MAC for messages of length n.

Domain Extension to Variable-Length Messages

- Can be prevented by including a random message identifier along with each block to prevent blocks from different messages from being combined.
- Below, we construct a fixed-length MAC for messages of length n.

Domain Extension to Variable-Length Messages

- Can be prevented by including a random message identifier along with each block to prevent blocks from different messages from being combined.
- Below, we construct a fixed-length MAC for messages of length n.

Construction 4: Signing

- $S'(k, m)$:
  - Inputs: a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of (nonzero) length $\ell < 2^{n/4}$.

Domain Extension to Variable-Length Messages

- Can be prevented by including a random message identifier along with each block to prevent blocks from different messages from being combined.
- Below, we construct a fixed-length MAC for messages of length n.

Construction 4: Signing

- $\mathcal{S}'(k,m)$:
  - Inputs: a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of (nonzero) length $\ell < 2^{n/4}$.
  - Parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$.

Domain Extension to Variable-Length Messages

- Can be prevented by including a random message identifier along with each block to prevent blocks from different messages from being combined.
- Below, we construct a fixed-length MAC for messages of length n.

Construction 4: Signing

- $\mathcal{S}'(k,m)$:
    - Inputs: a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of (nonzero) length $\ell < 2^{n/4}$.
    - Parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$.
    - Pad the final block with 0's to make it a multiple of $n/4$.

Domain Extension to Variable-Length Messages

- Can be prevented by including a random message identifier along with each block to prevent blocks from different messages from being combined.
- Below, we construct a fixed-length MAC for messages of length n.

Construction 4: Signing

- $\mathcal{S}'(k, m)$:
  - Inputs: a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of (nonzero) length $\ell < 2^{n/4}$.
  - Parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$.
  - Pad the final block with 0's to make it a multiple of $n/4$.
  - Choose a uniform identifier $r \xleftarrow{R} \{0,1\}^{n/4}$.

Domain Extension to Variable-Length Messages

- Can be prevented by including a random message identifier along with each block to prevent blocks from different messages from being combined.
- Below, we construct a fixed-length MAC for messages of length n.

Construction 4: Signing

- $\mathcal{S}'(k, m)$:
    - Inputs: a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of (nonzero) length $\ell < 2^{n/4}$.
    - Parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$.
    - Pad the final block with 0's to make it a multiple of $n/4$.
    - Choose a uniform identifier $r \xleftarrow{R} \{0,1\}^{n/4}$.
    - For $i = 1, \ldots, d$, Compute $t_i \leftarrow \mathcal{S}(k, r||\ell||i||m_i)$.

Domain Extension to Variable-Length Messages

- Can be prevented by including a random message identifier along with each block to prevent blocks from different messages from being combined.
- Below, we construct a fixed-length MAC for messages of length n.

Construction 4: Signing

- $\mathcal{S}'(k, m)$:
    - Inputs: a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of (nonzero) length $\ell < 2^{n/4}$.
    - Parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$.
    - Pad the final block with 0's to make it a multiple of $n/4$.
    - Choose a uniform identifier $r \xleftarrow{R} \{0,1\}^{n/4}$.
    - For $i = 1, \ldots, d$, Compute $t_i \leftarrow \mathcal{S}(k, r||\ell||i||m_i)$.
    - The tag $t := (r, t_1, \ldots, t_d)$, where i, ℓ are encoded as strings of length $n/4$ (as $i, \ell < 2^{n/4}$).

Domain Extension to Variable-Length Messages

Construction 4: Verification

- $\mathcal{V}'(k, m, t)$:
  - Inputs: key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^*$ of length $\ell < 2^{n/4}$, and a tag $t = (r, t_1, \ldots, t_{d'})$.

Domain Extension to Variable-Length Messages

Construction 4: Verification

- $\mathcal{V}'(k, m, t)$:
  - Inputs: key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^*$ of length $\ell < 2^{n/4}$, and a tag $t = (r, t_1, \ldots, t_{d'})$.
  - Parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$.

Domain Extension to Variable-Length Messages

Construction 4: Verification

- $\mathcal{V}'(k, m, t)$:
  - Inputs: key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^*$ of length $\ell < 2^{n/4}$, and a tag $t = (r, t_1, \ldots, t_{d'})$.
  - Parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$.
  - Pad the final block with 0's to make it a multiple of $n/4$.

Domain Extension to Variable-Length Messages

Construction 4: Verification

- $\mathcal{V}'(k,m,t)$:
  - Inputs: key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^*$ of length $\ell < 2^{n/4}$, and a tag $t = (r, t_1, \ldots, t_{d'})$.
  - Parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$.
  - Pad the final block with 0's to make it a multiple of $n/4$.
  - Output 1 if and only if
    1. $d' = d$, and
    2. $\mathcal{V}(k, r||\ell||i||m_i, t_i) = 1$, $\forall\ 1 \leq i \leq d$.

Domain Extension to Variable-Length Messages

Construction 4: Theorem

If $\mathcal{I}$ is a secure fixed-length MAC for messages of length n, then Construction 4 is a secure MAC for arbitrary-length messages.

Domain Extension to Variable-Length Messages

Construction 4: Theorem

If $\mathcal{I}$ is a secure fixed-length MAC for messages of length n, then Construction 4 is a secure MAC for arbitrary-length messages.

Construction 4: Comments

- Asymptotically, $2^{n/4}$ is an exponential bound.

Domain Extension to Variable-Length Messages

Construction 4: Theorem

If $\mathcal{I}$ is a secure fixed-length MAC for messages of length n, then Construction 4 is a secure MAC for arbitrary-length messages.

Construction 4: Comments

- Asymptotically, $2^{n/4}$ is an exponential bound.
- Honest parties will not authenticate messages that long.

Domain Extension to Variable-Length Messages

Construction 4: Theorem

If $\mathcal{I}$ is a secure fixed-length MAC for messages of length n, then Construction 4 is a secure MAC for arbitrary-length messages.

Construction 4: Comments

- Asymptotically, $2^{n/4}$ is an exponential bound.
- Honest parties will not authenticate messages that long.
- Any polynomial-time adversary cannot submit messages that long to its MAC oracle.

Domain Extension to Variable-Length Messages

Construction 4: Theorem

If $\mathcal{I}$ is a secure fixed-length MAC for messages of length n, then Construction 4 is a secure MAC for arbitrary-length messages.

Construction 4: Comments

- Asymptotically, $2^{n/4}$ is an exponential bound.
- Honest parties will not authenticate messages that long.
- Any polynomial-time adversary cannot submit messages that long to its MAC oracle.
- In practice, when n is fixed, one must ensure that $2^{n/4}$ is acceptable.

Domain Extension to Variable-Length Messages

Construction 4: Comments (cont.)

- A secure MAC for arbitrary-length messages from a pseudorandom function taking inputs of fixed length n.

Domain Extension to Variable-Length Messages

Construction 4: Comments (cont.)

- A secure MAC for arbitrary-length messages from a pseudorandom function taking inputs of fixed length n.
- Thus secure MACs can be constructed from block ciphers.

Domain Extension to Variable-Length Messages

Construction 4: Comments (cont.)

- A secure MAC for arbitrary-length messages from a pseudorandom function taking inputs of fixed length n.
- Thus secure MACs can be constructed from block ciphers.
- But, Construction 4 is extremely inefficient:

Domain Extension to Variable-Length Messages

Construction 4: Comments (cont.)

- A secure MAC for arbitrary-length messages from a pseudorandom function taking inputs of fixed length n.
- Thus secure MACs can be constructed from block ciphers.
- But, Construction 4 is extremely inefficient:
  - $4d$ evaluations of block cipher are required to compute a tag on a message of length dn.
  - The tag is more than $4dn$ bits long.

Domain Extension to Variable-Length Messages

Construction 4: Comments (cont.)

- A secure MAC for arbitrary-length messages from a pseudorandom function taking inputs of fixed length n.
- Thus secure MACs can be constructed from block ciphers.
- But, Construction 4 is extremely inefficient:
  - $4d$ evaluations of block cipher are required to compute a tag on a message of length dn.
  - The tag is more than $4dn$ bits long.
- Fortunately, far more efficient constructions exists.

Domain Extension to Variable-Length Messages

Construction 4: Comments (cont.)

- A secure MAC for arbitrary-length messages from a pseudorandom function taking inputs of fixed length n.
- Thus secure MACs can be constructed from block ciphers.
- But, Construction 4 is extremely inefficient:
  - $4d$ evaluations of block cipher are required to compute a tag on a message of length dn.
  - The tag is more than $4dn$ bits long.
- Fortunately, far more efficient constructions exists.
- We explore one such construction based on block ciphers.

Domain Extension to Variable-Length Messages

Construction 4: Comments (cont.)

- A secure MAC for arbitrary-length messages from a pseudorandom function taking inputs of fixed length n.
- Thus secure MACs can be constructed from block ciphers.
- But, Construction 4 is extremely inefficient:
  - $4d$ evaluations of block cipher are required to compute a tag on a message of length dn.
  - The tag is more than $4dn$ bits long.
- Fortunately, far more efficient constructions exists.
- We explore one such construction based on block ciphers.

Given a secure PRF on short inputs construct a secure PRF on long inputs.

CBC-MAC

CBC-MAC: Basic

- It is a standardized MAC, used widely in practice.

CBC-MAC

CBC-MAC: Basic

- It is a standardized MAC, used widely in practice.
- We present here a basic version of CBC-MAC.

CBC-MAC

CBC-MAC: Basic

- It is a standardized MAC, used widely in practice.
- We present here a basic version of CBC-MAC.
- **Caution:** This basic scheme is not secure in the general case when messages of different lengths may be authenticated.

CBC-MAC: Fixed Length

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC: Fixed Length

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$: Construction of Signing $\mathcal{S}_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$ and a message m of length $\ell(n) \cdot n$.

CBC-MAC: Fixed Length

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$: Construction of Signing $\mathcal{S}_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$ and a message m of length $\ell(n) \cdot n$.
    1. Parse m as $m = (m_1, \ldots, m_\ell)$ where each m_i is of length n.

CBC-MAC: Fixed Length

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$: Construction of Signing $\mathcal{S}_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$ and a message m of length $\ell(n) \cdot n$.

1. Parse m as $m = (m_1, \ldots, m_\ell)$ where each m_i is of length n.
2. Set $t_0 \longleftarrow 0^n$.

CBC-MAC: Fixed Length

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$: Construction of Signing $\mathcal{S}_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$ and a message m of length $\ell(n) \cdot n$.
  1. Parse m as $m = (m_1, \ldots, m_\ell)$ where each m_i is of length n.
  2. Set $t_0 \longleftarrow 0^n$.
  3. For $i = 1$ to ℓ, $t_i \longleftarrow F(k, t_{i-1} \oplus m_i)$.

CBC-MAC: Fixed Length

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$: Construction of Signing $\mathcal{S}_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$ and a message m of length $\ell(n) \cdot n$.
  1. Parse m as $m = (m_1, \ldots, m_\ell)$ where each m_i is of length n.
  2. Set $t_0 \longleftarrow 0^n$.
  3. For $i = 1$ to ℓ, $t_i \longleftarrow F(k, t_{i-1} \oplus m_i)$.
- Output t_ℓ as the tag.

CBC-MAC: Fixed Length

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$: Construction of Verification $\mathcal{V}_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$, a message m of length $\ell(n) \cdot n$, and tag t.

CBC-MAC: Fixed Length

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$: Construction of Verification $\mathcal{V}_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$, a message m of length $\ell(n) \cdot n$, and tag t.

  1. If $|m| \neq \ell(n) \cdot n$ then output 0.

CBC-MAC: Fixed Length

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$: Construction of Verification $\mathcal{V}_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$, a message m of length $\ell(n) \cdot n$, and tag t.
  1. If $|m| \neq \ell(n) \cdot n$ then output 0.
  2. Parse m as $m = (m_1, \ldots, m_\ell)$ where each m_i is of length n.
  3. Set $t'_0 := 0^n$.
  4. For $i = 1$ to ℓ, $t'_i \longleftarrow F(k, t_{i-1} \oplus m_i)$.

CBC-MAC: Fixed Length

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$: Construction of Verification $\mathcal{V}_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$, a message m of length $\ell(n) \cdot n$, and tag t.
  1. If $|m| \neq \ell(n) \cdot n$ then output 0.
  2. Parse m as $m = (m_1, \ldots, m_\ell)$ where each m_i is of length n.
  3. Set $t'_0 := 0^n$.
  4. For $i = 1$ to ℓ, $t'_i \longleftarrow F(k, t_{i-1} \oplus m_i)$.
- Output $t \overset{?}{=} t'_\ell$.

CBC-MAC: Fixed Length

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$: Construction of Verification $\mathcal{V}_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$, a message m of length $\ell(n) \cdot n$, and tag t.
    1. If $|m| \neq \ell(n) \cdot n$ then output 0.
    2. Parse m as $m = (m_1, \ldots, m_\ell)$ where each m_i is of length n.
    3. Set $t'_0 := 0^n$.
    4. For $i = 1$ to ℓ, $t'_i \longleftarrow F(k, t_{i-1} \oplus m_i)$.
- Output $t \overset{?}{=} t'_\ell$.

Theorem

Let ℓ be a polynomial. If F is a pseudorandom function, then CBC-MAC construction $\mathcal{I}_{\mathsf{CBC}}$ is a secure MAC for messages of length $\ell(n) \cdot n$.

CBC-MAC: Fixed Length

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$: Construction of Verification $\mathcal{V}_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$, a message m of length $\ell(n) \cdot n$, and tag t.
  1. If $|m| \neq \ell(n) \cdot n$ then output 0.
  2. Parse m as $m = (m_1, \ldots, m_\ell)$ where each m_i is of length n.
  3. Set $t'_0 := 0^n$.
  4. For $i = 1$ to ℓ, $t'_i \longleftarrow F(k, t_{i-1} \oplus m_i)$.
- Output $t \overset{?}{=} t'_\ell$.

Theorem

Let ℓ be a polynomial. If F is a pseudorandom function, then CBC-MAC construction $\mathcal{I}_{\mathsf{CBC}}$ is a secure MAC for messages of length $\ell(n) \cdot n$.

CBC-MAC $\mathcal{I}_{\mathsf{CBC}}$ is much more efficient than Construction 4

- Requires d block-cipher evaluations for a message of length dn.
- Tag length is n.

CBC-MAC: Arbitrary-length messages

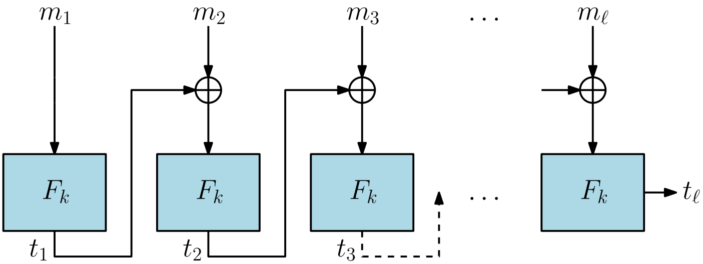


Figure: MAC-CBC

CBC-MAC: Arbitrary-length messages

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC: Arbitrary-length messages

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC $\mathcal{I}_{\mathsf{CBC}} - 2$: Construction of Signing $\mathcal{S}'_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$ and a message m of length $v \cdot n$ where $1 \leqslant v \leqslant \ell(n)$.

CBC-MAC: Arbitrary-length messages

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC $\mathcal{I}_{\mathsf{CBC}} - 2$: Construction of Signing $\mathcal{S}'_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$ and a message m of length $v \cdot n$ where $1 \leqslant v \leqslant \ell(n)$.

  1. Parse m as $m = (m_1, \ldots, m_v)$ where each m_i is of length n.

CBC-MAC: Arbitrary-length messages

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC $\mathcal{I}_{\mathsf{CBC}} - 2$: Construction of Signing $\mathcal{S}'_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$ and a message m of length $v \cdot n$ where $1 \leqslant v \leqslant \ell(n)$.
  1. Parse m as $m = (m_1, \ldots, m_v)$ where each m_i is of length n.
  2. Set $t_0 \longleftarrow 0^n$.

CBC-MAC: Arbitrary-length messages

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC $\mathcal{I}_{\mathsf{CBC}} - 2$: Construction of Signing $\mathcal{S}'_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$ and a message m of length $v \cdot n$ where $1 \leqslant v \leqslant \ell(n)$.
  1. Parse m as $m = (m_1, \ldots, m_v)$ where each m_i is of length n.
  2. Set $t_0 \longleftarrow 0^n$.
  3. For $i = 1$ to v, $t_i \longleftarrow F(k, t_{i-1} \oplus m_i)$.

CBC-MAC: Arbitrary-length messages

- Let F be a pseudorandom function, and fix a length function $\ell > 0$.

CBC-MAC $\mathcal{I}_{\mathsf{CBC}} - 2$: Construction of Signing $\mathcal{S}'_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$ and a message m of length $v \cdot n$ where $1 \leqslant v \leqslant \ell(n)$.
  1. Parse m as $m = (m_1, \ldots, m_v)$ where each m_i is of length n.
  2. Set $t_0 \longleftarrow 0^n$.
  3. For $i = 1$ to v, $t_i \longleftarrow F(k, t_{i-1} \oplus m_i)$.
- Output t_v as the tag.

CBC-MAC: Arbitrary-length messages

CBC-MAC $\mathcal{I}_{\mathsf{CBC}} - 2$: Construction of Verification $\mathcal{V}'_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$, a message m of length $v \cdot n$ where $1 \leqslant v \leqslant \ell(n)$, and tag t.

CBC-MAC: Arbitrary-length messages

CBC-MAC $\mathcal{I}_{\mathsf{CBC}} - 2$: Construction of Verification $\mathcal{V}'_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$, a message m of length $v \cdot n$ where $1 \leqslant v \leqslant \ell(n)$, and tag t.

  1. If $|m| > \ell(n) \cdot n$ then output 0.

CBC-MAC: Arbitrary-length messages

CBC-MAC $\mathcal{I}_{\mathsf{CBC}} - 2$: Construction of Verification $\mathcal{V}'_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$, a message m of length $v \cdot n$ where $1 \leqslant v \leqslant \ell(n)$, and tag t.

  1. If $|m| > \ell(n) \cdot n$ then output 0.
  2. Parse m as $m = (m_1, \ldots, m_{v'})$ where each m_i is of length n.
  3. Set $t'_0 := 0^n$.
  4. For $i = 1$ to v', $t'_i \longleftarrow F(k, t_{i-1} \oplus m_i)$.

CBC-MAC: Arbitrary-length messages

CBC-MAC $\mathcal{I}_{\mathsf{CBC}} - 2$: Construction of Verification $\mathcal{V}'_{\mathsf{CBC}}$

- Inputs: Key $k \in \{0,1\}^n$, a message m of length $v \cdot n$ where $1 \leqslant v \leqslant \ell(n)$, and tag t.

  1. If $|m| > \ell(n) \cdot n$ then output 0.
  2. Parse m as $m = (m_1, \ldots, m_{v'})$ where each m_i is of length n.
  3. Set $t'_0 := 0^n$.
  4. For $i = 1$ to v', $t'_i \longleftarrow F(k, t_{i-1} \oplus m_i)$.

- Output $t \overset{?}{=} t'_{v'}$.

CBC-MAC

Prefix-free Secure PRF

Let F be a PRF defined over $(\mathcal{K}, \mathcal{X}^{\leqslant \ell}, \mathcal{Y})$. We say that a PRF adversary $\mathcal{A}$ playing PRF Indistinguishability Attack Game with respect to F is a prefix-free adversary if all of its queries are non-empty strings over $\mathcal{X}$ of length at most ℓ, no one of which is a proper prefix of another. We denote $\mathcal{A}$'s advantage in winning the game by $\mathsf{PRF^{pf}adv}[\mathcal{A}, F]$. Further, let us say that F is a prefix-free secure PRF if $\mathsf{PRF^{pf}adv}[\mathcal{A}, F]$ is negligible for all efficient, prefix-free adversaries $\mathcal{A}$.

CBC-MAC

Prefix-free Secure PRF

Let F be a PRF defined over $(\mathcal{K}, \mathcal{X}^{\leqslant \ell}, \mathcal{Y})$. We say that a PRF adversary $\mathcal{A}$ playing PRF Indistinguishability Attack Game with respect to F is a prefix-free adversary if all of its queries are non-empty strings over $\mathcal{X}$ of length at most ℓ, no one of which is a proper prefix of another. We denote $\mathcal{A}$'s advantage in winning the game by $\mathsf{PRF}^{\mathsf{pf}}\mathsf{adv}[\mathcal{A}, F]$. Further, let us say that F is a prefix-free secure PRF if $\mathsf{PRF}^{\mathsf{pf}}\mathsf{adv}[\mathcal{A}, F]$ is negligible for all efficient, prefix-free adversaries $\mathcal{A}$.

Theorem

Let F be a secure PRF defined over $(\mathcal{K}, \mathcal{X}, \mathcal{X})$ where $\mathcal{X} = \{0,1\}^n$ and $|\mathcal{X}| = 2^n$ is super-poly. Then for any poly-bounded value ℓ, we have that $\mathcal{I}_{\mathsf{CBC}} - 2$ is a prefix-free secure PRF defined over $(\mathcal{K}, \mathcal{X}^{\leqslant \ell}, \mathcal{X})$.
In particular, for every prefix-free PRF adversary $\mathcal{A}$ that attacks $\mathcal{I}_{\mathsf{CBC}} - 2$ as in PRF Indistinguishability Attack Game, and issues at most Q queries, there exists a PRF adversary $\mathcal{B}$ that attacks F as in PRF Indistinguishability Attack Game, where $\mathcal{B}$ calls $\mathcal{A}$ as subroutine, such that

$$\mathsf{PRF}^{\mathsf{pf}}\mathsf{adv}[\mathcal{A}, \mathcal{I}_{\mathsf{CBC}} - 2] \leqslant \mathsf{PRFadv}[\mathcal{B}, F] + \frac{(Q\ell)^2}{2|\mathcal{X}|}.$$

End