Symmetric Key Cryptography : Hash and MAC

Avijit Dutta

Cryptology and Security Research Unit
Indian Statistical Institute, Kolkata, India

Outline

Definition

A Hash Function $H : \{0,1\}^m \rightarrow \{0,1\}^n$ such that $m >> n$.

Definition

A Hash Function $H : \{0,1\}^m \rightarrow \{0,1\}^n$ such that $m >> n$.

Definition

A Hash Function Family $\mathcal{H} := \{H_k : k \in \mathcal{K}\}$ such that each $H_k : \{0,1\}^m \rightarrow \{0,1\}^n$

Definition

A Hash Function $H : \{0,1\}^m \rightarrow \{0,1\}^n$ such that $m >> n$.

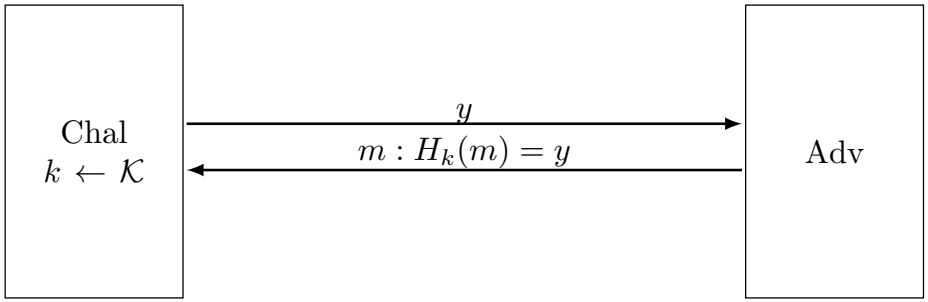
Definition

A Hash Function Family $\mathcal{H} := \{H_k : k \in \mathcal{K}\}$ such that each $H_k : \{0,1\}^m \rightarrow \{0,1\}^n$

Choosing a hash function $h \xleftarrow{\$} \mathcal{H}$ is equivalen to choose $k \xleftarrow{\$} \mathcal{K}$ and then set $h \leftarrow H_k$

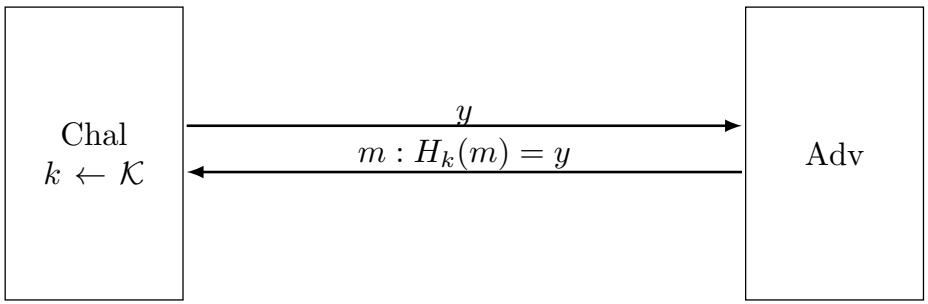
Properties of Hash Function

Pre-Image Resistant



Properties of Hash Function

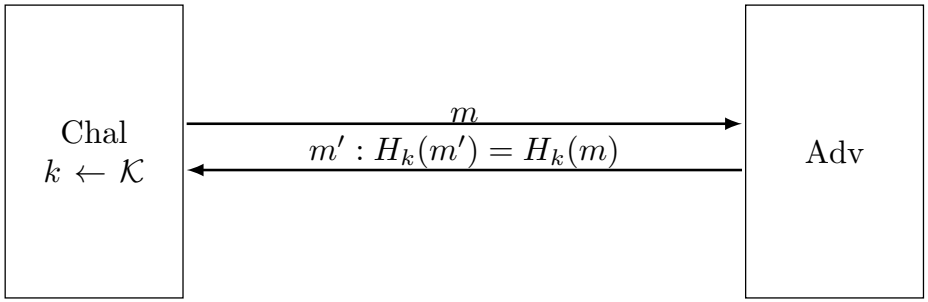
Pre-Image Resistant



ϵ-Pre-Image Secure $\Rightarrow \forall$ efficient $\mathcal{A}$, Pr[$\mathcal{A}$ wins the game] $\leq \epsilon$

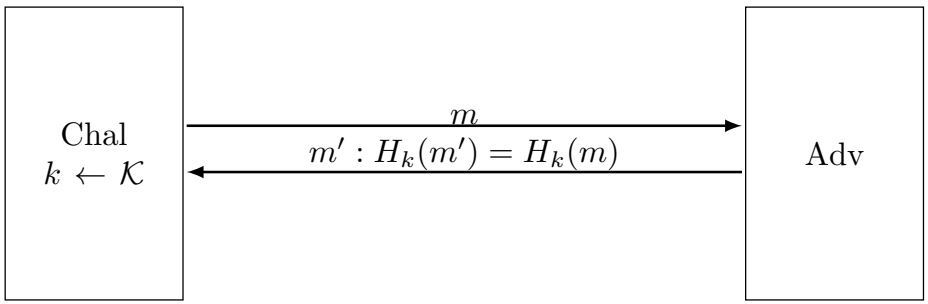
Properties of Hash Function

Second-Pre-Image Resistant



Properties of Hash Function

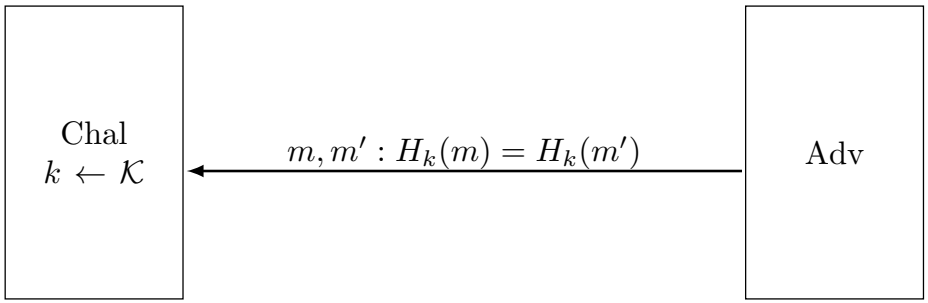
Second-Pre-Image Resistant



ϵ-Second-Pre-Image Secure $\Rightarrow \forall$ efficient $\mathcal{A}$, Pr[$\mathcal{A}$ wins the game] $\leq \epsilon$

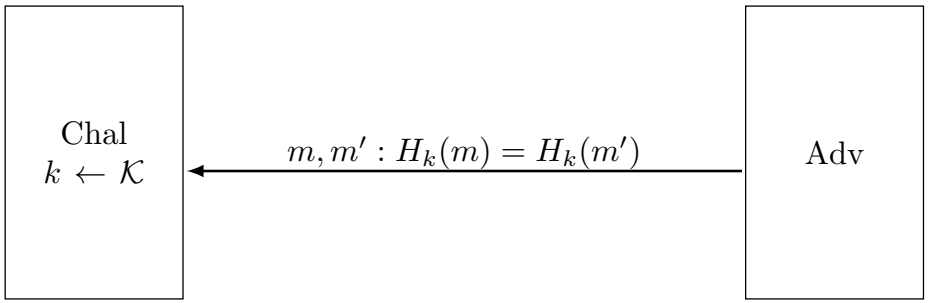
Properties of Hash Function

Collision Resistant



Properties of Hash Function

Collision Resistant



ϵ-Collision Resistance $\Rightarrow \forall$ efficient $\mathcal{A}$, Pr[$\mathcal{A}$ wins the game] $\leq \epsilon$

Properties of Hash Function

Facts

- A Good Hash Function should be 2^{-n} pre-image secure

Properties of Hash Function

Facts

- A Good Hash Function should be 2^{-n} pre-image secure
- A Good Hash Function should be 2^{-n} second-pre-image secure

Properties of Hash Function

Facts

- A Good Hash Function should be 2^{-n} pre-image secure
- A Good Hash Function should be 2^{-n} second-pre-image secure
- The success probability of finding a collision in a hash function is at least $\frac{q^2}{2^n}$ (Birthday Bound !)

Birthday Bound

Problem Statement

How many people must there be in a room before there is a 50% chance that two of them were boron on the same day of the year ?

Birthday Bound

Problem Statement

How many people must there be in a room before there is a 50% chance that two of them were boron on the same day of the year ?

Let us assume # of days is n.

- If $N = n + 1$, probability of collision is 1 (Pigeonhole Principle)

Birthday Bound

Problem Statement

How many people must there be in a room before there is a 50% chance that two of them were boron on the same day of the year ?

Let us assume # of days is n.

- If $N = n + 1$, probability of collision is 1 (Pigeonhole Principle)
- To ensure probability $\frac{1}{2}$, is the number $n/2$?

Birthday Bound

Problem Statement

How many people must there be in a room before there is a 50% chance that two of them were boron on the same day of the year ?

Let us assume # of days is n.

- If $N = n + 1$, probability of collision is 1 (Pigeonhole Principle)
- To ensure probability $\frac{1}{2}$, is the number $n/2$?
- The real answer is surprisingly small. Its about $\Omega(\sqrt{n})$

Collision Attack in Hash Function

Let $H : \{0,1\}^* \rightarrow \{0,1\}^n$ be a hash function.

Collision Finding Algorithm

1. Choose $2^{n/2}$ random message $m \in \{0,1\}^*$ $(m_1, m_2, \ldots, m_{2^{n/2}})$ message will be distinct with high probability.
2. for each of these messages compute $t \leftarrow H(m)$
3. if $\exists i \neq j$ such that $t_i = t_j$ return 1. else go to step 1

Collision Attack in Hash Function

Let $H : \{0,1\}^* \rightarrow \{0,1\}^n$ be a hash function.

Collision Finding Algorithm

1. Choose $2^{n/2}$ random message $m \in \{0,1\}^*$ $(m_1, m_2, \ldots, m_{2^{n/2}})$ message will be distinct with high probability.
2. for each of these messages compute $t \leftarrow H(m)$
3. if $\exists i \neq j$ such that $t_i = t_j$ return 1. else go to step 1

Analogy : $\{0,1\}^n$ is the set of days in the birthday paradox setting

Relation between PR, 2-PR, and CR

Relation between PR, 2-PR, and CR

$\text{CR} \Rightarrow \text{2-PR} \Rightarrow \text{PR}$

Relation between PR, 2-PR, and CR

CR $\Rightarrow$ 2-PR $\Rightarrow$ PR

Proof.

Proof in the contrapositive way.

- If $\mathcal{A}$ attacks PR then $\exists \mathcal{B}$ attacks 2-PR

Relation between PR, 2-PR, and CR

CR $\Rightarrow$ 2-PR $\Rightarrow$ PR

Proof.

Proof in the contrapositive way.

- If $\mathcal{A}$ attacks PR then $\exists \mathcal{B}$ attacks 2-PR
- If $\mathcal{B}$ attacks 2-PR then $\exists \mathcal{C}$ attacks CR

Construction of a Hash Function

Q. How to construct a secure hash function ?

Construction of a Hash Function

Q. How to construct a secure hash function ?

The general idea of building a higher primitive is to start from a lower primitive.

Construction of a Hash Function

Q. How to construct a secure hash function ?

The general idea of building a higher primitive is to start from a lower primitive.

- To construct a hash function, the lower primitive is the Compression Function

Construction of a Hash Function

Q. How to construct a secure hash function ?

The general idea of building a higher primitive is to start from a lower primitive.

- To construct a hash function, the lower primitive is the Compression Function
- This Compression function is a fixed length primitive.
  $f : \{0,1\}^c \times \{0,1\}^b \rightarrow \{0,1\}^c$

Construction of a Hash Function

Q. How to construct a secure hash function ?

The general idea of building a higher primitive is to start from a lower primitive.

- To construct a hash function, the lower primitive is the Compression Function
- This Compression function is a fixed length primitive.
  $f : \{0,1\}^c \times \{0,1\}^b \rightarrow \{0,1\}^c$
- From a fixed length compression function, design a variable length hash function by iterating the compression function

Construction of a Hash Function

Q. How to construct a secure hash function ?

The general idea of building a higher primitive is to start from a lower primitive.

- To construct a hash function, the lower primitive is the Compression Function
- This Compression function is a fixed length primitive.
  $f : \{0,1\}^c \times \{0,1\}^b \rightarrow \{0,1\}^c$
- From a fixed length compression function, design a variable length hash function by iterating the compression function
- The most popular approach : Merkle-Damgard Technique of Iterating Hash Function.

Davis-Meyer Construction

How to construct a Compression function ?

Davis-Meyer Construction

How to construct a Compression function ?

The very old and popular method to construct a compression function out of a block-cipher is due to Davis and Meyer.

Davis-Meyer Construction

How to construct a Compression function ?

The very old and popular method to construct a compression function out of a block-cipher is due to Davis and Meyer.

Davis-Meyer Compression Function

Let $e : \{0,1\}^n \times \{0,1\}^c \to \{0,1\}^c$ be a block-cipher and thus for any $K \in \{0,1\}^n, e_K := e(K, \cdot)$ is a permutation over $\{0,1\}^c$. The Davis-Meyer Compression Function $\mathrm{DM}_e : \{0,1\}^c \times \{0,1\}^n \to \{0,1\}^c$, defined as $\mathrm{DM}_e(h, m) = e_m(h) \oplus h$.

Merkle-Damgard Hash Function

Merkle-Damgard Hash Function

Given a compression function $f : \{0,1\}^c \times \{0,1\}^b \rightarrow \{0,1\}^c$, we define the iterated hash function as follows:

Merkle-Damgard Hash Function

Given a compression function $f : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$, we define the iterated hash function as follows:

- Given a message $M \in \{0,1\}^*$, if $|M| \neq bk$ for some k, $M' = M||\langle l \rangle$ to make $|M'| = bk$. Else $M' = M$.

Merkle-Damgard Hash Function

Given a compression function $f : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$, we define the iterated hash function as follows:

- Given a message $M \in \{0,1\}^*$, if $|M| \neq bk$ for some k, $M' = M||\langle l \rangle$ to make $|M'| = bk$. Else $M' = M$.
- Spilts the message $M = (M_1, M_2, \ldots, M_\ell) \ni |M_i| = b$

Merkle-Damgard Hash Function

Given a compression function $f : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$, we define the iterated hash function as follows:

- Given a message $M \in \{0,1\}^*$, if $|M| \neq bk$ for some k, $M' = M||\langle l \rangle$ to make $|M'| = bk$. Else $M' = M$.
- Spilts the message $M = (M_1, M_2, \ldots, M_\ell) \ni |M_i| = b$
- Initialize $h_0 = IV \in \{0,1\}^c$ a fixed publicly known value.

Merkle-Damgard Hash Function

Given a compression function $f : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$, we define the iterated hash function as follows:

- Given a message $M \in \{0,1\}^*$, if $|M| \neq bk$ for some k, $M' = M||\langle l \rangle$ to make $|M'| = bk$. Else $M' = M$.
- Spilts the message $M = (M_1, M_2, \ldots, M_\ell) \ni |M_i| = b$
- Initialize $h_0 = IV \in \{0,1\}^c$ a fixed publicly known value.
- for $i = 1$ to ℓ, $h_i = f(h_{i-1}, M_i)$

Merkle-Damgard Hash Function

Given a compression function $f : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$, we define the iterated hash function as follows:

- Given a message $M \in \{0,1\}^*$, if $|M| \neq bk$ for some k, $M' = M||\langle l \rangle$ to make $|M'| = bk$. Else $M' = M$.
- Spilts the message $M = (M_1, M_2, \ldots, M_\ell) \ni |M_i| = b$
- Initialize $h_0 = IV \in \{0,1\}^c$ a fixed publicly known value.
- for $i = 1$ to ℓ, $h_i = f(h_{i-1}, M_i)$
- Return h_ℓ

Merkle-Damgard Hash Function

Theorem

Let f be a fixed compression function and H^f be the Merkle-Damgard Iterated hash Function. If f is a collision resistant function then H^f is also a collision resistant function

Merkle-Damgard Hash Function

Theorem

Let f be a fixed compression function and H^f be the Merkle-Damgard Iterated hash Function. If f is a collision resistant function then H^f is also a collision resistant function

Proof.

Proof the contrapositive. □

Merkle-Damgard Hash Function

Theorem

Let f be a fixed compression function and H^f be the Merkle-Damgard Iterated hash Function. If f is a collision resistant function then H^f is also a collision resistant function

Proof.

Proof the contrapositive. □

Remark

- To design a collision resistant hash function, it is enough to design a collision resistant compression function

Merkle-Damgard Hash Function

Theorem

Let f be a fixed compression function and H^f be the Merkle-Damgard Iterated hash Function. If f is a collision resistant function then H^f is also a collision resistant function

Proof.

Proof the contrapositive. □

Remark

- To design a collision resistant hash function, it is enough to design a collision resistant compression function
- We cannot lift the preimage resistant or second preimage resistant security to that of compression function.

Merkle-Damgard Hash Function

Note

- It was believed until 2005 that Merkle Damgard Iterated Hash Function is 2^n preimage secure. But Kelsey and Schneier [2] in Eurocrypt'05 showed $\frac{2^n}{\ell}$ attack.

Merkle-Damgard Hash Function

Note

- It was believed until 2005 that Merkle Damgard Iterated Hash Function is 2^n preimage secure. But Kelsey and Schneier [2] in Eurocrypt'05 showed $\frac{2^n}{\ell}$ attack.

Theorem

Let f be a public random function and H^f be the Merkle-Damgard Iterated Hash Function. Let $\mathcal{A}$ be a preimage adversary (q, ℓ, ϵ) breaks H^f. Then,

$$\epsilon \leq \frac{q\ell}{2^n}$$

.

Merkle-Damgard Hash Function

Note

- It was believed until 2005 that Merkle Damgard Iterated Hash Function is 2^n preimage secure. But Kelsey and Schneier [2] in Eurocrypt'05 showed $\frac{2^n}{\ell}$ attack.

Theorem

Let f be a public random function and H^f be the Merkle-Damgard Iterated Hash Function. Let $\mathcal{A}$ be a preimage adversary (q, ℓ, ϵ) breaks H^f. Then,

$$\epsilon \leq \frac{q\ell}{2^n}$$

.

Proof.

We shall look at the proof in later slide. □

Multi-Collision Attack(Joux [1])

Algorithm

- set $h_0 = IV$

Multi-Collision Attack(Joux [1])

Algorithm

- set $h_0 = IV$
- for $i = 1$ to t

  call C with h_{i-1}, C will return $B \neq B'$ such that $f(h_{i-1}, B) = f(h_{i-1}, B')$.

Multi-Collision Attack(Joux [1])

Algorithm

- set $h_0 = IV$
- for $i = 1$ to t
  call C with h_{i-1}, C will return $B \neq B'$ such that $f(h_{i-1}, B) = f(h_{i-1}, B')$.
- set $h_i = f(h_{i-1}, B)$

Multi-Collision Attack(Joux [1])

Algorithm

- set $h_0 = IV$
- for $i = 1$ to t
  call C with h_{i-1}, C will return $B \neq B'$ such that $f(h_{i-1}, B) = f(h_{i-1}, B')$.
- set $h_i = f(h_{i-1}, B)$
- Pad and output 2^t messages of the form $(b_1, \ldots, b_t, \text{Pad})$ such that each $b_i = \{B_i, B_i')$

Multi-Collision Attack(Joux [1])

Remark

- The collision finding algorithm C on f may use generic birthday attack or any other attack exploiting the weakness of f.

Multi-Collision Attack(Joux [1])

Remark

- The collision finding algorithm C on f may use generic birthday attack or any other attack exploiting the weakness of f.
- The above attack is possible when message block size is larger than chaining variable size

Multi-Collision Attack(Joux [1])

Remark

- The collision finding algorithm C on f may use generic birthday attack or any other attack exploiting the weakness of f.
- The above attack is possible when message block size is larger than chaining variable size
- In other case, one could merge two or more blocks until the merged block size becomes equal to the chaining variable size

Multi-Collision Attack(Joux [1])

Remark

- The collision finding algorithm C on f may use generic birthday attack or any other attack exploiting the weakness of f.
- The above attack is possible when message block size is larger than chaining variable size
- In other case, one could merge two or more blocks until the merged block size becomes equal to the chaining variable size

Time Complexity

$f : \{0,1\}^n \times \{0,1\}^b \rightarrow \{0,1\}^n$ then the time complexity of the above attack is $t \times 2^{n/2}$

Multi-Collision Attack(Joux [1])

Remark

- The collision finding algorithm C on f may use generic birthday attack or any other attack exploiting the weakness of f.
- The above attack is possible when message block size is larger than chaining variable size
- In other case, one could merge two or more blocks until the merged block size becomes equal to the chaining variable size

Time Complexity

$f : \{0,1\}^n \times \{0,1\}^b \rightarrow \{0,1\}^n$ then the time complexity of the above attack is $t \times 2^{n/2}$

Application

- Collision-Resistance on the Cascading Hash Function.

Multi-Collision Attack(Joux [1])

Remark

- The collision finding algorithm C on f may use generic birthday attack or any other attack exploiting the weakness of f.
- The above attack is possible when message block size is larger than chaining variable size
- In other case, one could merge two or more blocks until the merged block size becomes equal to the chaining variable size

Time Complexity

$f : \{0,1\}^n \times \{0,1\}^b \rightarrow \{0,1\}^n$ then the time complexity of the above attack is $t \times 2^{n/2}$

Application

- Collision-Resistance on the Cascading Hash Function.
- Pre-Image and Second-Pre-Image on the Cascading Hash Function.

CR, PR and 2-PR on the Cascading Hash Function

Motivation

CR, PR and 2-PR on the Cascading Hash Function

Motivation

- A Long Standing Open Problem : Is the concatenation of two independent hash values more secure than a single hash-value ?

CR, PR and 2-PR on the Cascading Hash Function

Motivation

- A Long Standing Open Problem : Is the concatenation of two independent hash values more secure than a single hash-value ?
- Cascading of hash function was proposed in the PhD thesis of B.Preneel to increase the security.

CR, PR and 2-PR on the Cascading Hash Function

Motivation

- A Long Standing Open Problem : Is the concatenation of two independent hash values more secure than a single hash-value ?
- Cascading of hash function was proposed in the PhD thesis of B.Preneel to increase the security.
- If F and G are two hash functions output n_f and n_g bits respectively, then $F||G$ seems to be $2^{n_f/2+n_g/2}$ collision resistant secure.

CR, PR and 2-PR on the Cascading Hash Function

Motivation

- A Long Standing Open Problem : Is the concatenation of two independent hash values more secure than a single hash-value ?
- Cascading of hash function was proposed in the PhD thesis of B.Preneel to increase the security.
- If F and G are two hash functions output n_f and n_g bits respectively, then $F||G$ seems to be $2^{n_f/2+n_g/2}$ collision resistant secure.
- If F and G are two hash functions output n_f and n_g bits respectively, then $F||G$ seems to be $2^{n_f+n_g}$ pre image and second pre image secure.

CR, PR and 2-PR on the Cascading Hash Function

Motivation

- A Long Standing Open Problem : Is the concatenation of two independent hash values more secure than a single hash-value ?
- Cascading of hash function was proposed in the PhD thesis of B.Preneel to increase the security.
- If F and G are two hash functions output n_f and n_g bits respectively, then $F||G$ seems to be $2^{n_f/2+n_g/2}$ collision resistant secure.
- If F and G are two hash functions output n_f and n_g bits respectively, then $F||G$ seems to be $2^{n_f+n_g}$ pre image and second pre image secure.

Joux Attack [1]

- Collision : $q \approx n_g \times 2^{n_f/2} + 2^{n_g/2}$ if $n_f \leq n_g$ (respectively $n_f \times 2^{n_g/2} + 2^{n_f/2}$ if $n_f \geq n_g$).

CR, PR and 2-PR on the Cascading Hash Function

Motivation

- A Long Standing Open Problem : Is the concatenation of two independent hash values more secure than a single hash-value ?
- Cascading of hash function was proposed in the PhD thesis of B.Preneel to increase the security.
- If F and G are two hash functions output n_f and n_g bits respectively, then $F||G$ seems to be $2^{n_f/2+n_g/2}$ collision resistant secure.
- If F and G are two hash functions output n_f and n_g bits respectively, then $F||G$ seems to be $2^{n_f+n_g}$ pre image and second pre image secure.

Joux Attack [1]

- Collision : $q \approx n_g \times 2^{n_f/2} + 2^{n_g/2}$ if $n_f \leq n_g$ (respectively $n_f \times 2^{n_g/2} + 2^{n_f/2}$ if $n_f \geq n_g$).
- Pre-Image and Second-Pre-Image : $q \approx n_g \times 2^{n_f/2} + 2^{n_f} + 2^{n_g}$ if $n_f \leq n_g$ (respectively $n_f \times 2^{n_g/2} + 2^{n_g} + 2^{n_f}$ if $n_f \geq n_g$)

Collision-Resistance on the Cascading Hash Function

Attack Algorithm (Case $n_f \leq n_g$)

Collision-Resistance on the Cascading Hash Function

Attack Algorithm (Case $n_f \leq n_g$)

- set t to $n_g/2$

Collision-Resistance on the Cascading Hash Function

Attack Algorithm (Case $n_f \leq n_g$)

- set t to $n_g/2$
- Construct 2^t multi collision for F. This would cost to $t2^{n_f/2} = n_g/2 \times 2^{n_f/2}$.

Collision-Resistance on the Cascading Hash Function

Attack Algorithm (Case $n_f \leq n_g$)

- set t to $n_g/2$
- Construct 2^t multi collision for F. This would cost to $t2^{n_f/2} = n_g/2 \times 2^{n_f/2}$.
- Since $t \geq n_g$ 2, out of $2^t = 2^{n_g/2}$ messages there exists at least a pair of message collides for H.

Collision-Resistance on the Cascading Hash Function

Attack Algorithm (Case $n_f \leq n_g$)

- set t to $n_g/2$
- Construct 2^t multi collision for F. This would cost to $t2^{n_f/2} = n_g/2 \times 2^{n_f/2}$.
- Since $t \geq n_g$ 2, out of $2^t = 2^{n_g/2}$ messages there exists at least a pair of message collides for H.
- Total query complexity $q \approx n_g/2 \times 2^{n_f/2} + 2^{n_g/2} \leq 2^{n_f/2+n_g/2}$.

Remark

Finding collision for H requires $t \times 2^t$ evaluation of the compression function of H. But due to the tree structure of 2^t messages, it requires 2^t evaluatios of the compression function of H.

Collision-Resistance on the Cascading Hash Function

Note

- In this attack G is not neccessarily to be a iterated hash function

Collision-Resistance on the Cascading Hash Function

Note

- In this attack G is not neccessarily to be a iterated hash function
- One can replace G with random oracle and the same attack will apply with the same query complexity

Collision-Resistance on the Cascading Hash Function

Note

- In this attack G is not neccessarily to be a iterated hash function
- One can replace G with random oracle and the same attack will apply with the same query complexity
- Cascading two good independent hash functions does not improve the collision resistance.

Pre-Image and Second-Pre-Image on the Cascading Hash Function

Attack Algorithm

- set t to n_g

Pre-Image and Second-Pre-Image on the Cascading Hash Function

Attack Algorithm

- set t to n_g
- Construct 2^t multi collision for F. This would cost to $t2^{n_f/2} = n_g \times 2^{n_f/2}$.

Pre-Image and Second-Pre-Image on the Cascading Hash Function

Attack Algorithm

- set t to n_g
- Construct 2^t multi collision for F. This would cost to $t2^{n_f/2} = n_g \times 2^{n_f/2}$.
- Search for an additional block that maps to target value of F. This would require 2^{n_f} call to the compression function of F.

Pre-Image and Second-Pre-Image on the Cascading Hash Function

Attack Algorithm

- set t to n_g
- Construct 2^t multi collision for F. This would cost to $t2^{n_f/2} = n_g \times 2^{n_f/2}$.
- Search for an additional block that maps to target value of F. This would require 2^{n_f} call to the compression function of F.
- Since $t \geq n_g$, out of $2^t = 2^{n_g}$ messages there exists at least one message with probability 1 that hashes to the target value of H.

Pre-Image and Second-Pre-Image on the Cascading Hash Function

Attack Algorithm

- set t to n_g
- Construct 2^t multi collision for F. This would cost to $t2^{n_f/2} = n_g \times 2^{n_f/2}$.
- Search for an additional block that maps to target value of F. This would require 2^{n_f} call to the compression function of F.
- Since $t \geq n_g$, out of $2^t = 2^{n_g}$ messages there exists at least one message with probability 1 that hashes to the target value of H.
- Return the message.

Pre-Image and Second-Pre-Image on the Cascading Hash Function

Attack Algorithm

- set t to n_g
- Construct 2^t multi collision for F. This would cost to $t2^{n_f/2} = n_g \times 2^{n_f/2}$.
- Search for an additional block that maps to target value of F. This would require 2^{n_f} call to the compression function of F.
- Since $t \geq n_g$, out of $2^t = 2^{n_g}$ messages there exists at least one message with probability 1 that hashes to the target value of H.
- Return the message.
- Total query complexity $q \approx n_g \times 2^{n_f/2} + 2^{n_f} + 2^{n_g} \leq 2^{n_f+n_g}$.

Pre-Image and Second-Pre-Image on Cascading Hash Function

Remark

- G need not be iterated hash function. It could be replaced by random oracle and the attack works !

Pre-Image and Second-Pre-Image on Cascading Hash Function

Remark

- G need not be iterated hash function. It could be replaced by random oracle and the attack works !
- Second-Pre-Image attack works in the same way.Attacker computes $F||G(M)$ and then invoke the attack algorithm with input $F||G(M)$

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

Finding 2nd preimage of a challenge message is soley based on the idea of Expandable Message

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

Finding 2nd preimage of a challenge message is soley based on the idea of Expandable Message

Expandable Message

An Expandable Message is a pattern of messages of different lengths which all yield the same intermediate hash value after processing them

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

Finding 2nd preimage of a challenge message is soley based on the idea of Expandable Message

Expandable Message

An Expandable Message is a pattern of messages of different lengths which all yield the same intermediate hash value after processing them

How to find a pair of colliding messages having unequal number of message block. e.g. Find a colliding message pair (m, m') such that $|m| = 1$ and $|m'| = \alpha$

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

Finding 2nd preimage of a challenge message is soley based on the idea of Expandable Message

Expandable Message

An Expandable Message is a pattern of messages of different lengths which all yield the same intermediate hash value after processing them

How to find a pair of colliding messages having unequal number of message block. e.g. Find a colliding message pair (m, m') such that $|m| = 1$ and $|m'| = \alpha$

- Collision Finding Algorithm : FindColl(α, h_{in})

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

Finding 2nd preimage of a challenge message is soley based on the idea of Expandable Message

Expandable Message

An Expandable Message is a pattern of messages of different lengths which all yield the same intermediate hash value after processing them

How to find a pair of colliding messages having unequal number of message block. e.g. Find a colliding message pair (m, m') such that $|m| = 1$ and $|m'| = \alpha$

- Collision Finding Algorithm : FindColl(α, h_{in})
- # of compression function call : $\alpha - 1 + 2^{n/2+1}$

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

Building Full $(k, k + 2^k - 1)$ Expandable Message

- MakeExpandableMessage(h_{in}, k)

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

Building Full $(k, k + 2^k - 1)$ Expandable Message

- MakeExpandableMessage(h_{in}, k)
- $\#$. of compression function call : $k \times 2^{n/2+1} + 2^k - 1$.

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

Building Full $(k, k + 2^k - 1)$ Expandable Message

- MakeExpandableMessage(h_{in}, k)
- $\#$. of compression function call : $k \times 2^{n/2+1} + 2^k - 1$.

Second Preimage attack

- h_{exp} be the internal hash chaining value of expandable message

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

Building Full $(k, k + 2^k - 1)$ Expandable Message

- MakeExpandableMessage(h_{in}, k)
- $\#$. of compression function call : $k \times 2^{n/2+1} + 2^k - 1$.

Second Preimage attack

- h_{exp} be the internal hash chaining value of expandable message
- Find M_{link} such that $f(h_{exp}, M_{link}) = h_j$, $k + 1 \leq j \leq 2^k + k - 1$

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

Building Full $(k, k + 2^k - 1)$ Expandable Message

- MakeExpandableMessage(h_{in}, k)
- $\#$. of compression function call : $k \times 2^{n/2+1} + 2^k - 1$.

Second Preimage attack

- h_{exp} be the internal hash chaining value of expandable message
- Find M_{link} such that $f(h_{exp}, M_{link}) = h_j$, $k + 1 \leq j \leq 2^k + k - 1$
- Choose the expandable message $M*$ of j blocks and return $M * ||M_{link}||m_{j+1}||m_{j+2}|| \ldots ||m_{2^k+k-1}$

$\frac{2^n}{\ell}$ Pre-Image Attack on Merkle Damgard Iterated Hash Function (Kelsey et al. [2].)

Building Full $(k, k + 2^k - 1)$ Expandable Message

- MakeExpandableMessage(h_{in}, k)
- #. of compression function call : $k \times 2^{n/2+1} + 2^k - 1$.

Second Preimage attack

- h_{exp} be the internal hash chaining value of expandable message
- Find M_{link} such that $f(h_{exp}, M_{link}) = h_j$, $k + 1 \leq j \leq 2^k + k - 1$
- Choose the expandable message $M*$ of j blocks and return $M * ||M_{link}||m_{j+1}||m_{j+2}|| \ldots ||m_{2^k+k-1}$

#. of compression function call : $k \times 2^{n/2+1} + 2^{n-k+1}$

Proof of $\frac{2^n}{\ell}$ Second-Pre-Image Security of Merkle Damgard Iterated Hash Function.

Theorem

Let f be a public random function and H^f be the Merkle-Damgard Iterated Hash Function. Let $\mathcal{A}$ be a second-preimage adversary (q, ℓ, ϵ) breaks H^f. Then,

$$\epsilon \leq \frac{q\ell}{2^n}$$

.

Proof.

Given M, $\mathcal{A}$ will compute $H^f(M)$. Let h_{i_0} be the chaining value, $1 \leq i_0 \leq \ell$ for computing hash value of M. Success of $\mathcal{A} \rightarrow$ Collision in H^f. Let M' be the output of $\mathcal{A} : H^f(M) = H^f(M')$. We know, Collision in $H^f \rightarrow$ Collision in $h_{i_0}, 1 \leq i_0 \leq \ell$. □

HAIFA (A Framework for Iterative Hash Function) [1]

HAIFA (A Framework for Iterative Hash Function) [1]

Given a compression function $f : \{0,1\}^s \times \{0,1\}^b \times \{0,1\}^c \rightarrow \{0,1\}^c$, we define the iterated hash function as follows:

HAIFA (A Framework for Iterative Hash Function) [1]

Given a compression function $f : \{0,1\}^s \times \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$, we define the iterated hash function as follows:

- Given a message $M \in \{0,1\}^*$, if $|M| \neq bk$ for some k, $M' = M||\langle l \rangle$ to make $|M'| = bk$. Else $M' = M$.

HAIFA (A Framework for Iterative Hash Function) [1]

Given a compression function $f : \{0,1\}^s \times \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$, we define the iterated hash function as follows:

- Given a message $M \in \{0,1\}^*$, if $|M| \neq bk$ for some k, $M' = M||\langle l \rangle$ to make $|M'| = bk$. Else $M' = M$.
- Spilts the message $M = (M_1, M_2, \ldots, M_\ell) \ni |M_i| = b$

HAIFA (A Framework for Iterative Hash Function) [1]

Given a compression function $f : \{0,1\}^s \times \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$, we define the iterated hash function as follows:

- Given a message $M \in \{0,1\}^*$, if $|M| \neq bk$ for some k, $M' = M||\langle l \rangle$ to make $|M'| = bk$. Else $M' = M$.
- Spilts the message $M = (M_1, M_2, \ldots, M_\ell) \ni |M_i| = b$
- Initialize $h_0 = IV \in \{0,1\}^c$ a fixed publicly known value.

HAIFA (A Framework for Iterative Hash Function) [1]

Given a compression function $f : \{0,1\}^s \times \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$, we define the iterated hash function as follows:

- Given a message $M \in \{0,1\}^*$, if $|M| \neq bk$ for some k, $M' = M||\langle l \rangle$ to make $|M'| = bk$. Else $M' = M$.
- Spilts the message $M = (M_1, M_2, \ldots, M_\ell) \ni |M_i| = b$
- Initialize $h_0 = IV \in \{0,1\}^c$ a fixed publicly known value.
- for $i = 1$ to ℓ, $h_i = f(\langle i \rangle_s, h_{i-1}, M_i)$

HAIFA (A Framework for Iterative Hash Function) [1]

Given a compression function $f : \{0,1\}^s \times \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$, we define the iterated hash function as follows:

- Given a message $M \in \{0,1\}^*$, if $|M| \neq bk$ for some k, $M' = M||\langle l \rangle$ to make $|M'| = bk$. Else $M' = M$.
- Spilts the message $M = (M_1, M_2, \ldots, M_\ell) \ni |M_i| = b$
- Initialize $h_0 = IV \in \{0,1\}^c$ a fixed publicly known value.
- for $i = 1$ to ℓ, $h_i = f(\langle i \rangle_s, h_{i-1}, M_i)$
- Return h_ℓ

Proof of 2^n Pre-Image Security of HAIFA [2]

Theorem

Let f be a public random function and H^f be the HAIFA Iterated Hash Function. Let $\mathcal{A}$ be a preimage adversary (q, ℓ, ϵ) breaks H^f. Then,

$$\epsilon \leq \frac{q}{2^n}$$

.

Proof of 2^n Pre-Image Security of HAIFA [2]

Theorem

Let f be a public random function and H^f be the HAIFA Iterated Hash Function. Let $\mathcal{A}$ be a preimage adversary (q, ℓ, ϵ) breaks H^f. Then,

$$\epsilon \leq \frac{q}{2^n}$$

.

Proof.

$\mathcal{A}$ is given the challenge M and therefore $\mathcal{A}$ computes $H^f(M)$, storing all intermediate chaining values $h_{i_0}, 1 \leq i_0 \leq \ell$. Suppose $\mathcal{A}$ wins and outputs M'.

Proof of 2^n Pre-Image Security of HAIFA [2]

Theorem

Let f be a public random function and H^f be the HAIFA Iterated Hash Function. Let $\mathcal{A}$ be a preimage adversary (q, ℓ, ϵ) breaks H^f. Then,

$$\epsilon \leq \frac{q}{2^n}$$

.

Proof.

$\mathcal{A}$ is given the challenge M and therefore $\mathcal{A}$ computes $H^f(M)$, storing all intermediate chaining values $h_{i_0}, 1 \leq i_0 \leq \ell$. Suppose $\mathcal{A}$ wins and outputs M'.

- when $|M| \neq |M'|$, counter at the last invocation of the compression functions are different.

Proof of 2^n Pre-Image Security of HAIFA [2]

Theorem

Let f be a public random function and H^f be the HAIFA Iterated Hash Function. Let $\mathcal{A}$ be a preimage adversary (q, ℓ, ϵ) breaks H^f. Then,

$$\epsilon \leq \frac{q}{2^n}$$

.

Proof.

$\mathcal{A}$ is given the challenge M and therefore $\mathcal{A}$ computes $H^f(M)$, storing all intermediate chaining values $h_{i_0}, 1 \leq i_0 \leq \ell$. Suppose $\mathcal{A}$ wins and outputs M'.

- when $|M| \neq |M'|$, counter at the last invocation of the compression functions are different.
- When $|M| = |M'|$, collision with the same counter value or collision in the last invocation of the compression function.

Related Notion of Hash Function

Related Notion of Hash Function

ϵ-Universal Hash Function

- A hash family $\mathcal{H}$ is called ϵ-**universal** (ϵ-U) if $\max_{x \neq x'} \Pr_{\mathbf{K}}[h_{\mathbf{K}}(x) = h_{\mathbf{K}}(x')] \leq \epsilon$.

Related Notion of Hash Function

ϵ-Universal Hash Function

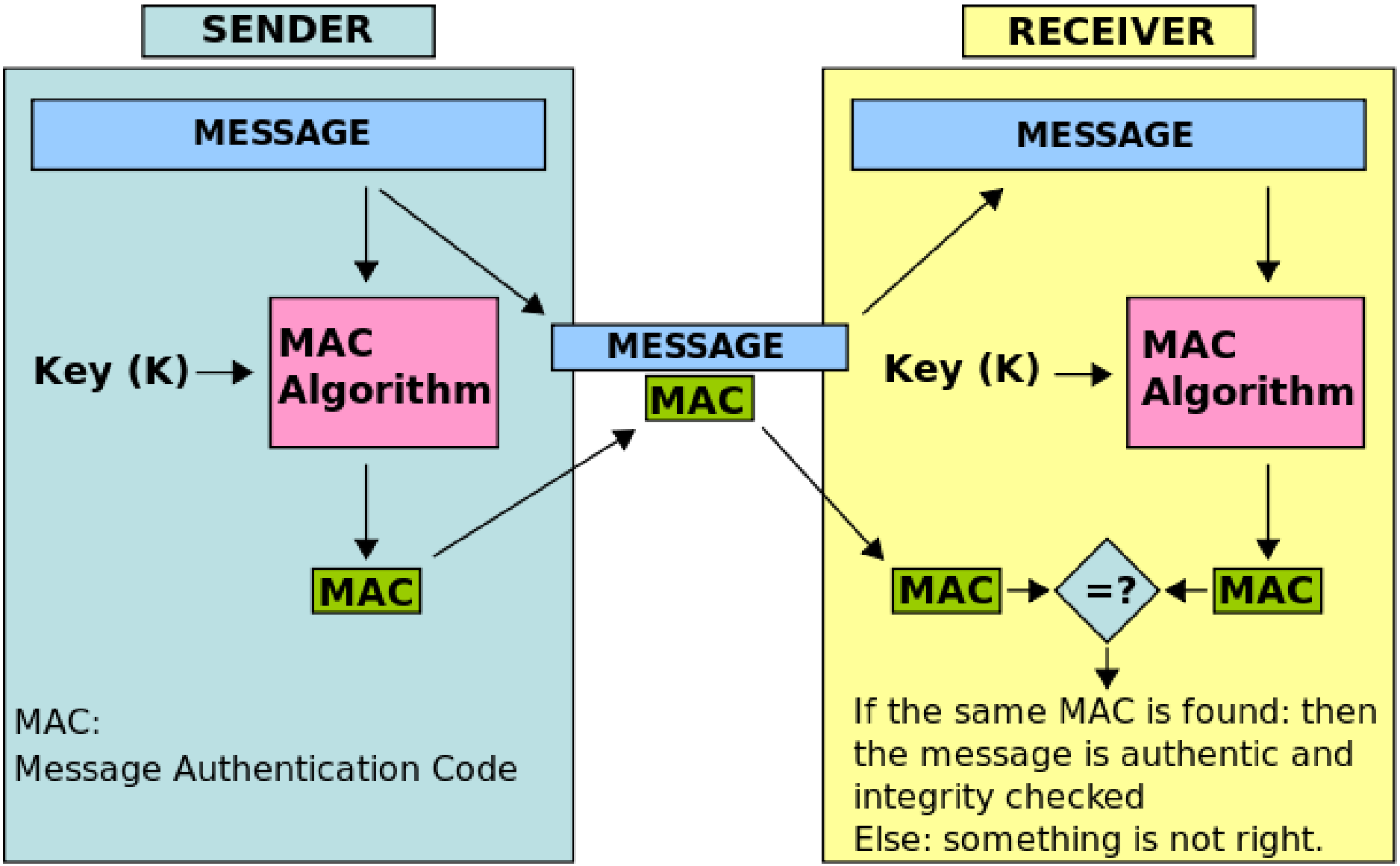
- A hash family $\mathcal{H}$ is called ϵ-**universal** (or ϵ-U) if $\max_{x \neq x'} \Pr_{\mathbf{K}}[h_{\mathbf{K}}(x) = h_{\mathbf{K}}(x')] \leq \epsilon$.

ϵ-AXU-Universal Hash Function

- A $(\mathcal{K}, \mathcal{D})$-family $\mathcal{H}$ is called ϵ-**Almost-XOR Universal** hash function, if for any two distinct x and x' in $\mathcal{D}$ and a $\delta \in \{0,1\}^n$, the δ-*differential probability* $\Pr_k[h_k(x) \oplus h_k(x') = \delta] \leq \epsilon$ where the random variable k is uniformly distributed over the set $\mathcal{K}$.

Symmetric Key Cryptography : Message Authentication Code (MAC)

MAC used in Communication



Definition of MAC

A Message Authentication Code (MAC) scheme Π, defined over $\mathcal{K}, \mathcal{M}, \mathcal{T}$ is a pair of algorithms $(\mathsf{TG}, \mathsf{VF})$ (possibly probabilistic) such that :

Definition of MAC

A Message Authentication Code (MAC) scheme Π, defined over $\mathcal{K}, \mathcal{M}, \mathcal{T}$ is a pair of algorithms $(\mathsf{TG}, \mathsf{VF})$ (possibly probabilistic) such that :

1. TF is an (possibly probabilistic) algorithm from $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$

Definition of MAC

A Message Authentication Code (MAC) scheme Π, defined over $\mathcal{K}, \mathcal{M}, \mathcal{T}$ is a pair of algorithms $(\mathsf{TG}, \mathsf{VF})$ (possibly probabilistic) such that :

1. TF is an (possibly probabilistic) algorithm from $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$
2. VG is a deterministic algorithm from $\mathcal{K} \times \mathcal{M} \times \mathcal{T} \rightarrow \{0, 1\}$

Definition of MAC

A Message Authentication Code (MAC) scheme Π, defined over $\mathcal{K}, \mathcal{M}, \mathcal{T}$ is a pair of algorithms $(\mathsf{TG}, \mathsf{VF})$ (possibly probabilistic) such that :

1. TF is an (possibly probabilistic) algorithm from $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$
2. VG is a deterministic algorithm from $\mathcal{K} \times \mathcal{M} \times \mathcal{T} \rightarrow \{0, 1\}$

Correctness Condition : $\forall k \in \mathcal{K}, \forall m \in \mathcal{M}$ and $\forall t \in \mathcal{T}$ with $\Pr[\mathsf{TG}(k, m) = t] > 0$ such that $\mathsf{VG}(k, m, t) = 1$

Definition of MAC

A Message Authentication Code (MAC) scheme Π, defined over $\mathcal{K}, \mathcal{M}, \mathcal{T}$ is a pair of algorithms $(\mathsf{TG}, \mathsf{VF})$ (possibly probabilistic) such that :

1. TF is an (possibly probabilistic) algorithm from $\mathcal{K} \times \mathcal{M} \to \mathcal{T}$
2. VG is a deterministic algorithm from $\mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \{0, 1\}$

Correctness Condition : $\forall k \in \mathcal{K}, \forall m \in \mathcal{M}$ and $\forall t \in \mathcal{T}$ with $\Pr[\mathsf{TG}(k, m) = t] > 0$ such that $\mathsf{VG}(k, m, t) = 1$

• Note: If TG is deterministic algorithm then the MAC is said to be deterministic MAC, otherwise it is called Probabilistic MAC

Security Game of MAC

- Power of the Adversary with respect to attacking MAC: *Adaptive Chosen Message Attack*

$\mathcal{A}$ queries the MAC oracle with messages m_i and the oracle returns the valid tag $t_i = F(k, m_i)$ $\forall 1 \leq i \leq q$.

Security Game of MAC

- Power of the Adversary with respect to attacking MAC: *Adaptive Chosen Message Attack*

$\mathcal{A}$ queries the MAC oracle with messages m_i and the oracle returns the valid tag $t_i = F(k, m_i)\ \forall 1 \leq i \leq q$.

- Goal of the Adversary : *Existential Forgery*

$\mathcal{A}$ produces a valid (m^*, t^*) pair such that $m^* \notin \{m_1, \ldots, m_q\}$ and $V(k, m^*, t^*) = 1$

Security Game of MAC

- Power of the Adversary with respect to attacking MAC: *Adaptive Chosen Message Attack*

$\mathcal{A}$ queries the MAC oracle with messages m_i and the oracle returns the valid tag $t_i = F(k, m_i)\ \forall 1 \leq i \leq q$.

- Goal of the Adversary : *Existential Forgery*

$\mathcal{A}$ produces a valid (m^*, t^*) pair such that $m^* \notin \{m_1, \ldots, m_q\}$ and $V(k, m^*, t^*) = 1$

Adv $\mathcal{A}$ wins if $m^* \notin \{m_1, m_2, \ldots, m_q\}$ and $Vrfy(m^*, t^*) = 1$.

Security Game of MAC

- Power of the Adversary with respect to attacking MAC: *Adaptive Chosen Message Attack*

$\mathcal{A}$ queries the MAC oracle with messages m_i and the oracle returns the valid tag $t_i = F(k, m_i)\ \forall 1 \leq i \leq q$.

- Goal of the Adversary : *Existential Forgery*

$\mathcal{A}$ produces a valid (m^*, t^*) pair such that $m^* \notin \{m_1, \ldots, m_q\}$ and $V(k, m^*, t^*) = 1$

Adv $\mathcal{A}$ wins if $m^* \notin \{m_1, m_2, \ldots, m_q\}$ and $Vrfy(m^*, t^*) = 1$.

$\mathbf{Adv}^{\mathrm{mac}}(\mathcal{A}) := \Pr[\mathcal{A}$ wins the game $]$

Security Game of MAC

- Power of the Adversary with respect to attacking MAC: *Adaptive Chosen Message Attack*

$\mathcal{A}$ queries the MAC oracle with messages m_i and the oracle returns the valid tag $t_i = F(k, m_i)$ $\forall 1 \leq i \leq q$.

- Goal of the Adversary : *Existential Forgery*

$\mathcal{A}$ produces a valid (m^*, t^*) pair such that $m^* \notin \{m_1, \ldots, m_q\}$ and $V(k, m^*, t^*) = 1$

Adv $\mathcal{A}$ wins if $m^* \notin \{m_1, m_2, \ldots, m_q\}$ and $Vrfy(m^*, t^*) = 1$.

$\mathbf{Adv}^{\mathrm{mac}}(\mathcal{A}) := \Pr[\mathcal{A} \text{ wins the game }]$

$\mathbf{Adv}^{\mathrm{mac}}(q, \ell, t) := \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{mac}}(\mathcal{A})$

More Definitions Related to MAC

- Stateful MAC : Holds the state and with each message update the state. (Nonce based MAC)(e.g XMACC)

More Definitions Related to MAC

- Stateful MAC : Holds the state and with each message update the state. (Nonce based MAC)(e.g XMACC)
- Stateless MAC : Doesn't require to hold any state (e.g, XMACR, LightMAC, CBC, HMAC)

More Definitions Related to MAC

- Stateful MAC : Holds the state and with each message update the state. (Nonce based MAC)(e.g XMACC)
- Stateless MAC : Doesn't require to hold any state (e.g, XMACR, LightMAC, CBC, HMAC)
- Single Forgery : $\mathcal{A}$ will be allowed to submit only one verification query.

More Definitions Related to MAC

- Stateful MAC : Holds the state and with each message update the state. (Nonce based MAC)(e.g XMACC)
- Stateless MAC : Doesn't require to hold any state (e.g, XMACR, LightMAC, CBC, HMAC)
- Single Forgery : $\mathcal{A}$ will be allowed to submit only one verification query.
- Multiple Forgery : $\mathcal{A}$ will be allowed to submit multiple verification query and MAC queries, verification queries are interleaved.

More Definitions Related to MAC

- Stateful MAC : Holds the state and with each message update the state. (Nonce based MAC)(e.g XMACC)
- Stateless MAC : Doesn't require to hold any state (e.g, XMACR, LightMAC, CBC, HMAC)
- Single Forgery : $\mathcal{A}$ will be allowed to submit only one verification query.
- Multiple Forgery : $\mathcal{A}$ will be allowed to submit multiple verification query and MAC queries, verification queries are interleaved.
- Weak Unforgeability : If $\mathcal{A}$ queries MAC oracle with M, then it cannot submit M to the verification oracle.

More Definitions Related to MAC

- Stateful MAC : Holds the state and with each message update the state. (Nonce based MAC)(e.g XMACC)
- Stateless MAC : Doesn't require to hold any state (e.g, XMACR, LightMAC, CBC, HMAC)
- Single Forgery : $\mathcal{A}$ will be allowed to submit only one verification query.
- Multiple Forgery : $\mathcal{A}$ will be allowed to submit multiple verification query and MAC queries, verification queries are interleaved.
- Weak Unforgeability : If $\mathcal{A}$ queries MAC oracle with M, then it cannot submit M to the verification oracle.
- Strong unforgeability : If $\mathcal{A}$ queries to MAC oracle with M, it can also submit M to the verification oracle

More Definitions Related to MAC

- Stateful MAC : Holds the state and with each message update the state. (Nonce based MAC)(e.g XMACC)
- Stateless MAC : Doesn't require to hold any state (e.g, XMACR, LightMAC, CBC, HMAC)
- Single Forgery : $\mathcal{A}$ will be allowed to submit only one verification query.
- Multiple Forgery : $\mathcal{A}$ will be allowed to submit multiple verification query and MAC queries, verification queries are interleaved.
- Weak Unforgeability : If $\mathcal{A}$ queries MAC oracle with M, then it cannot submit M to the verification oracle.
- Strong unforgeability : If $\mathcal{A}$ queries to MAC oracle with M, it can also submit M to the verification oracle

Result

UF-1 $\not\Rightarrow$ UF-M. But in Stronger Unforgeability model SUF-1 $\iff$ SUF-M due to Bellare et al. [3]

UF-1 $\not\Rightarrow$ UF-M (Bellare et al. [3])

UF-1 $\not\Rightarrow$ UF-M (Bellare et al. [3])

- $\Pi_1 = (\mathsf{KG}, \mathsf{TG}, \mathsf{VG})$ be a MAC

UF-1 $\not\Rightarrow$ UF-M (Bellare et al. [3])

- $\Pi_1 = (\mathsf{KG}, \mathsf{TG}, \mathsf{VG})$ be a MAC
- Construct $\Pi_2 = (\mathsf{KG}, \mathsf{TG}', \mathsf{VG}')$ be a MAC which is UF-1 but not UF-M

UF-1 $\not\Rightarrow$ UF-M (Bellare et al. [3])

- $\Pi_1 = (\mathsf{KG}, \mathsf{TG}, \mathsf{VG})$ be a MAC
- Construct $\Pi_2 = (\mathsf{KG}, \mathsf{TG}', \mathsf{VG}')$ be a MAC which is UF-1 but not UF-M

Tag-generation

1. On query $m, t \leftarrow \mathsf{TG}_k(m)$
2. $t' \leftarrow t || \langle 0 \rangle$
3. return t'.

Verification

1. On a query $(\widetilde{m}, \widetilde{t}')$
2. Parse $\widetilde{t}'$ into $\widetilde{t}' || \langle i \rangle$, where $i \in \{0, 1, \ldots, |k|\}$
3. $b \leftarrow \mathsf{VF}_k(\widetilde{m}, \widetilde{t})$
4. if $b = 0$ or $i = 0$, return b
5. if $b = 1$ and $i \geq 1$, return $k[i]$

UF-1 $\not\Rightarrow$ UF-M (Bellare et al. [3])

Attack

- $\mathcal{A}$ submits m, obtains $t \leftarrow \mathrm{TG}_k^{'}(m)$
- for $i = 0$ to $|k| - 1$ do
- $\mathcal{A}$ constructs $\widetilde{t} = t||\langle i \rangle$
- $\mathcal{A}$ queries $(m, \widetilde{t})$ to the verification oracle
- $\mathcal{A}$ obtains $k[i]$
- Endfor
- $\mathcal{A}$ recovers the whole key k.
- $\mathcal{A}$ constructs a new $(\widetilde{m}, \widetilde{t})$ pair and forge with probability 1.

Types of MAC

Types of MAC

- Probabilistic and Stateless MAC (e.g, XMACR, RMAC, EhtM etc.)

Types of MAC

- Probabilistic and Stateless MAC (e.g, XMACR, RMAC, EhtM etc.)
- Deterministic and Stateless MAC (e.g CBC-MAC, PMAC, LightMAC, PCS etc)

Types of MAC

- Probabilistic and Stateless MAC (e.g, XMACR, RMAC, EhtM etc.)
- Deterministic and Stateless MAC (e.g CBC-MAC, PMAC, LightMAC, PCS etc)
- Deterministic and Stateful MAC (e.g XMACC, Carter-Wegman MAC)

Types of MAC

- Probabilistic and Stateless MAC (e.g, XMACR, RMAC, EhtM etc.)
- Deterministic and Stateless MAC (e.g CBC-MAC, PMAC, LightMAC, PCS etc)
- Deterministic and Stateful MAC (e.g XMACC, Carter-Wegman MAC)
- Probabilistic and Stateful MAC (not useful in practice)

Types of MAC

- Probabilistic and Stateless MAC (e.g, XMACR, RMAC, EhtM etc.)
- Deterministic and Stateless MAC (e.g CBC-MAC, PMAC, LightMAC, PCS etc)
- Deterministic and Stateful MAC (e.g XMACC, Carter-Wegman MAC)
- Probabilistic and Stateful MAC (not useful in practice)

Note: We only discuss about Case (2).

How to design a secure-MAC

Can a secure Hash Function be used as a MAC ?

How to design a secure-MAC

Can a secure Hash Function be used as a MAC ?

Theorem

Let $F : K \times X \to Y$ *be a secure PRF. We define* $MAC = (S, V)$ *as follows :* $S(k, m) = F(k, m)$ *and* $V(k, m, t) = 1$ *if* $t = S(k, m)$ *else* $V(k, m, t) = 0$. *Then,* $\mathbf{Adv}_F^{mac}(q, \ell, t) \leq \mathbf{Adv}_F^{prf}(q, \ell, t) + \frac{1}{|Y|}$.

How to design a secure-MAC

Can a secure Hash Function be used as a MAC ?

Theorem

Let $F : K \times X \to Y$ *be a secure PRF. We define* $MAC = (S, V)$ *as follows :* $S(k, m) = F(k, m)$ *and* $V(k, m, t) = 1$ *if* $t = S(k, m)$ *else* $V(k, m, t) = 0$. *Then,* $\mathbf{Adv}^{mac}_F(q, \ell, t) \leq \mathbf{Adv}^{prf}_F(q, \ell, t) + \frac{1}{|Y|}$.

Proof.

Proof is by *Reduction Game* □

How to design a secure-MAC

Can a secure Hash Function be used as a MAC ?

Theorem

Let $F : K \times X \to Y$ *be a secure PRF. We define* $MAC = (S, V)$ *as follows :* $S(k, m) = F(k, m)$ *and* $V(k, m, t) = 1$ *if* $t = S(k, m)$ *else* $V(k, m, t) = 0$. *Then,* $\mathbf{Adv}^{mac}_F(q, \ell, t) \leq \mathbf{Adv}^{prf}_F(q, \ell, t) + \frac{1}{|Y|}$.

Proof.

Proof is by *Reduction Game* □

Corollary

To design a secure MAC, it is enough to design a secure PRF. That means if the distinguishability advantage of PRF is negligible then as $\frac{1}{|Y|}$ *is negligible the MAC advantage would be negligible*

Are these constructions secure PRF ?

Secure PRF ?/ Secure MAC ?

Let f, g be two random functions and h be a hash function. We ask the following :

Are these constructions secure PRF ?

Secure PRF ?/ Secure MAC ?

Let f, g be two random functions and h be a hash function. We ask the following :

- $F_f(r, m) = f(r) \oplus m$

Are these constructions secure PRF ?

Secure PRF ?/ Secure MAC ?

Let f, g be two random functions and h be a hash function. We ask the following :

- $F_f(r, m) = f(r) \oplus m$
- $F_{f,g}(r, m) = f(r) \oplus g(m)$

Are these constructions secure PRF ?

Secure PRF ?/ Secure MAC ?

Let f, g be two random functions and h be a hash function. We ask the following :

- $F_f(r, m) = f(r) \oplus m$
- $F_{f,g}(r, m) = f(r) \oplus g(m)$
- $F_{f,h}(r, m) = f(r) \oplus h(m)$

Are these constructions secure PRF ?

Secure PRF ?/ Secure MAC ?

Let f, g be two random functions and h be a hash function. We ask the following :

- $F_f(r, m) = f(r) \oplus m$
- $F_{f,g}(r, m) = f(r) \oplus g(m)$
- $F_{f,h}(r, m) = f(r) \oplus h(m)$
- $F_{f,g,h}(r, m) = f(r) \oplus g(h(m))$

Are these constructions secure PRF ?

Secure PRF ?/ Secure MAC ?

Let f, g be two random functions and h be a hash function. We ask the following :

- $F_f(r, m) = f(r) \oplus m$
- $F_{f,g}(r, m) = f(r) \oplus g(m)$
- $F_{f,h}(r, m) = f(r) \oplus h(m)$
- $F_{f,g,h}(r, m) = f(r) \oplus g(h(m))$
- $F_{f,g}(r, m) = f(r) \oplus g(r \oplus m)$

Are these constructions secure PRF ?

Secure PRF ?/ Secure MAC ?

Let f, g be two random functions and h be a hash function. We ask the following :

- $F_f(r, m) = f(r) \oplus m$
- $F_{f,g}(r, m) = f(r) \oplus g(m)$
- $F_{f,h}(r, m) = f(r) \oplus h(m)$
- $F_{f,g,h}(r, m) = f(r) \oplus g(h(m))$
- $F_{f,g}(r, m) = f(r) \oplus g(r \oplus m)$
- $F_{f,g,h}(r, m) = f(r) \oplus g(r \oplus h(m))$.

Approaches of Different Constructions

Mainly, three types of approaches of MAC constructions exists

Approaches of Different Constructions

Mainly, three types of approaches of MAC constructions exists

- Universal Hash Based MAC Construction (e.g UMAC)

Approaches of Different Constructions

Mainly, three types of approaches of MAC constructions exists

- Universal Hash Based MAC Construction (e.g UMAC)
- Compression Function Based MAC Construction (e.g HMAC)

Approaches of Different Constructions

Mainly, three types of approaches of MAC constructions exists

- Universal Hash Based MAC Construction (e.g UMAC)
- Compression Function Based MAC Construction (e.g HMAC)
- Iterated Block Cipher Based MAC Construction (e.g CBC-MAC, PMAC, CMAC, GCBC etc.)

Approaches of Different Constructions

Mainly, three types of approaches of MAC constructions exists

- Universal Hash Based MAC Construction (e.g UMAC)
- Compression Function Based MAC Construction (e.g HMAC)
- Iterated Block Cipher Based MAC Construction (e.g CBC-MAC, PMAC, CMAC, GCBC etc.)

Note: We discuss here only two candidates : (a) HMAC and (b) CBC-MAC

Composition Theorem : PRF(U) $\equiv$ PRF

Theorem (Shoup [4])

Let $G_{K_1,K_2} := F_{K_2} \circ h_{K_1} : \mathcal{D} \to \{0,1\}^n$ *where* h *is an* ϵ*-universal hash over* $\mathcal{D}$*. Then,*

$$\mathbf{Adv}^{\mathrm{prf}}_{G}(t,q,\ell) \leq \mathbf{Adv}^{\mathrm{prf}}_{F}(t',q,\ell) + \binom{q}{2} \times \epsilon,$$

where $t' = t + \mathcal{O}(qT)$ *and* T *denotes the maximum time for computing* h*.*

Proof.

Output of the function is indistinguishable from random until the collision occurs in the input of F_{K_2}. $\square$

Merkle-Damgard Iterated Hash Construction

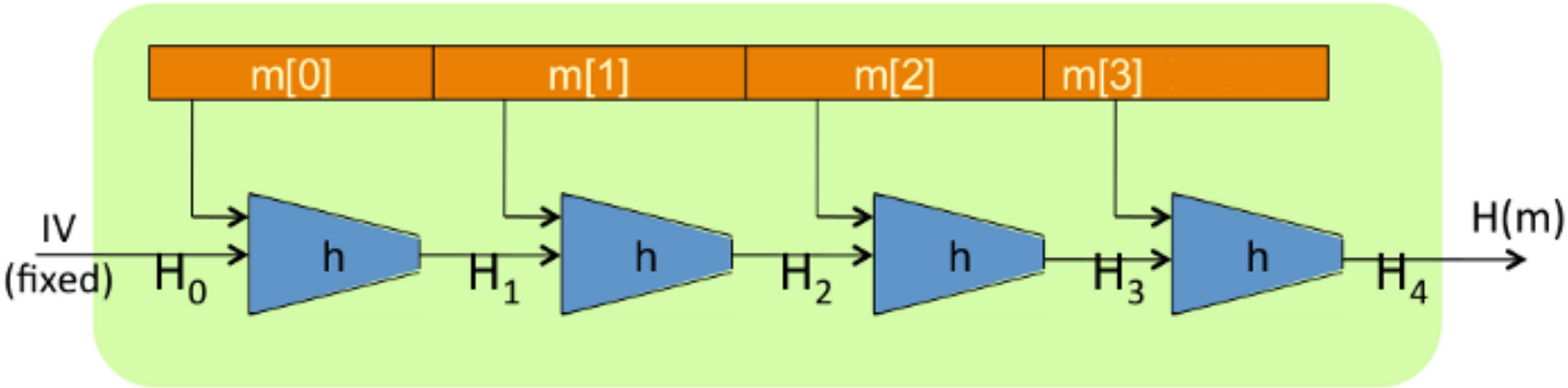
Objective

Develop a Hash function $H(\cdot)$ with larger domain from a Hash function $h(\cdot)$ with smaller domain.

Merkle-Damgard Iterated Hash Construction

Objective

Develop a Hash function $H(\cdot)$ with larger domain from a Hash function $h(\cdot)$ with smaller domain.



Construction of Hash-Based-MAC (HMAC)

- HMAC is a Hash-Function Based Mac designed by H. Krawczyk, M. Bellare and R. Canetti (1997).

Construction of Hash-Based-MAC (HMAC)

- HMAC is a Hash-Function Based Mac designed by H. Krawczyk, M. Bellare and R. Canetti (1997).
- Widely used in internet.

Construction of Hash-Based-MAC (HMAC)

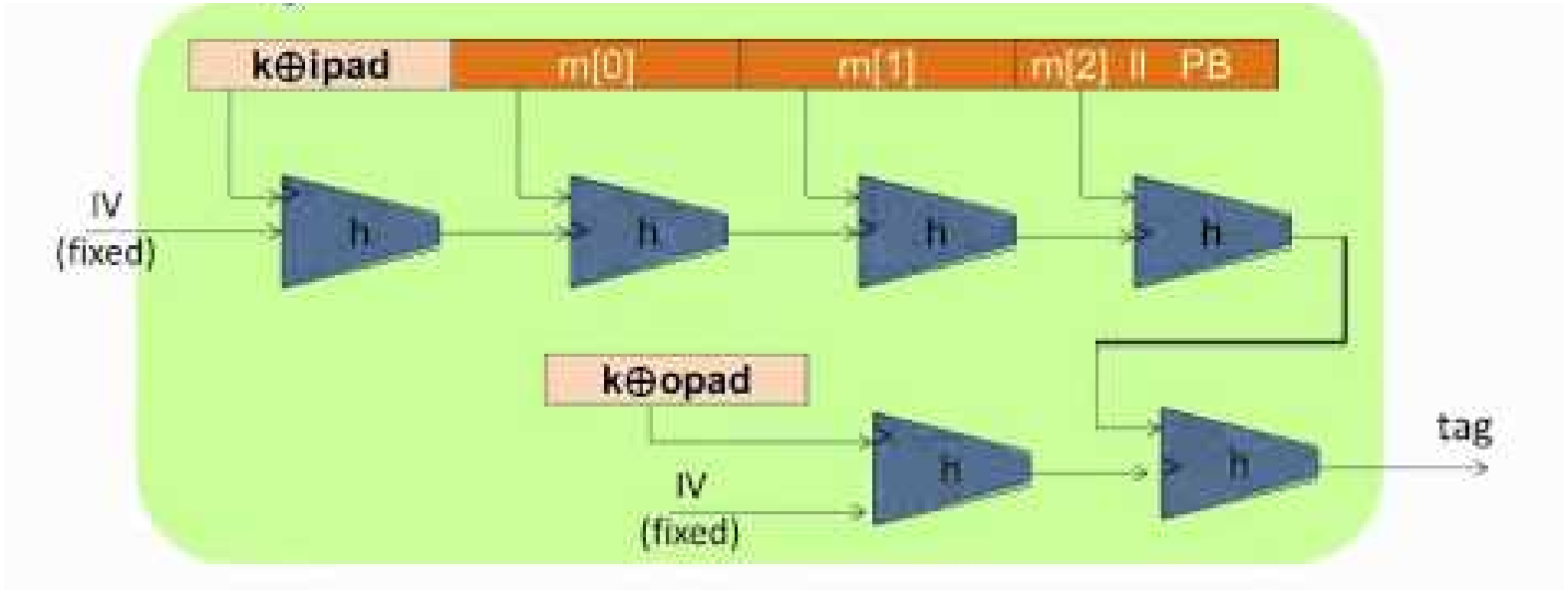
- HMAC is a Hash-Function Based Mac designed by H. Krawczyk, M. Bellare and R. Canetti (1997).
- Widely used in internet.
- HMAC is built based on the technique of Merkle-Damgard Construction

Construction of Hash-Based-MAC (HMAC)

- HMAC is a Hash-Function Based Mac designed by H. Krawczyk, M. Bellare and R. Canetti (1997).
- Widely used in internet.
- HMAC is built based on the technique of Merkle-Damgard Construction
- Instantiated with SHA-256 hash Function, output is 256 bits.

Construction of Hash-Based-MAC (HMAC)

- HMAC is a Hash-Function Based Mac designed by H. Krawczyk, M. Bellare and R. Canetti (1997).
- Widely used in internet.
- HMAC is built based on the technique of Merkle-Damgard Construction
- Instantiated with SHA-256 hash Function, output is 256 bits.



Construction of Cipher Block Chaining MAC (CBC-MAC)

- Iterated block cipher based MAC, specified by US Govt. Standard *FIPS PUB 113 Computer Data Authentication*(2000) using DES as a block-cipher

Construction of Cipher Block Chaining MAC (CBC-MAC)

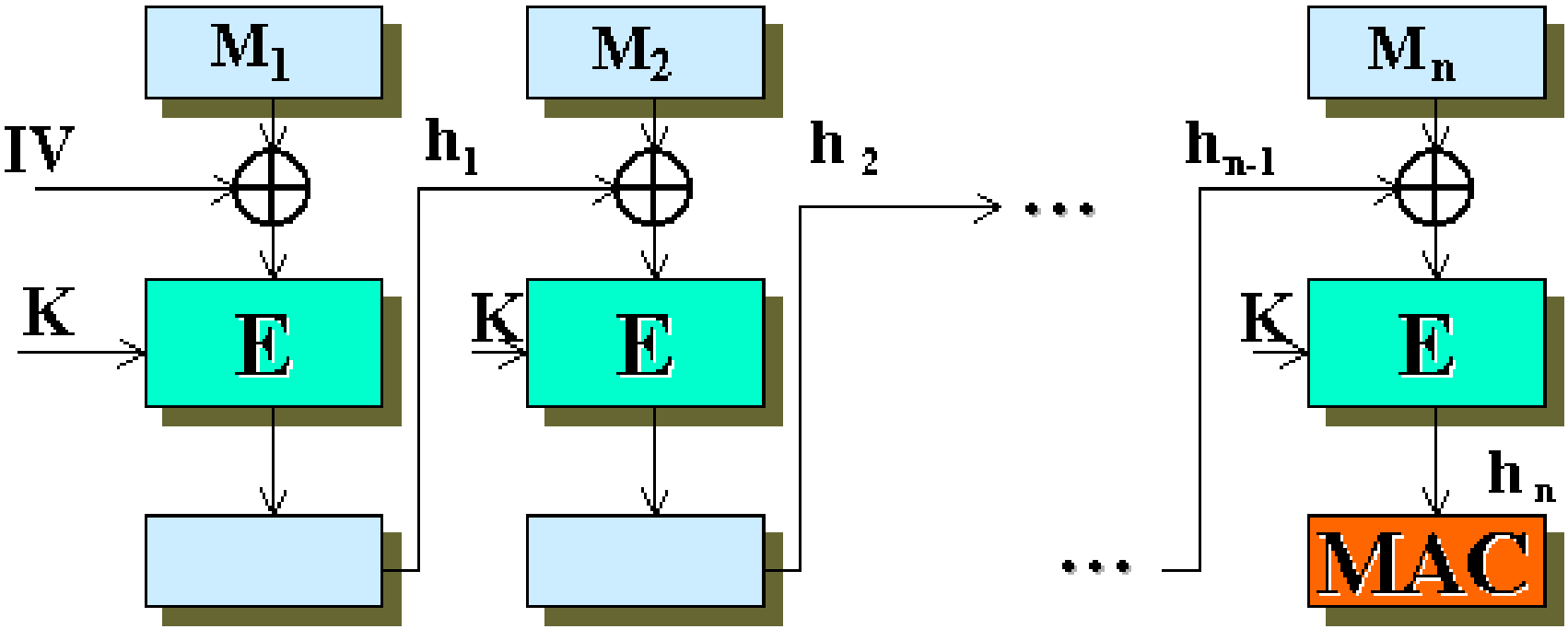
- Iterated block cipher based MAC, specified by US Govt. Standard *FIPS PUB 113 Computer Data Authentication*(2000) using DES as a block-cipher
- Standard in Crypto Community and widely used in practice.

Construction of Cipher Block Chaining MAC (CBC-MAC)

- Iterated block cipher based MAC, specified by US Govt. Standard *FIPS PUB 113 Computer Data Authentication*(2000) using DES as a block-cipher
- Standard in Crypto Community and widely used in practice.
- Sequential in nature. Thus efficieny wise it is not attractive.

Construction of Cipher Block Chaining MAC (CBC-MAC)

- Iterated block cipher based MAC, specified by US Govt. Standard *FIPS PUB 113 Computer Data Authentication*(2000) using DES as a block-cipher
- Standard in Crypto Community and widely used in practice.
- Sequential in nature. Thus efficieny wise it is not attractive.



Encrypted Cipher Block Chaining MAC (ECBC-MAC)

CBC-MAC is not secure when message length is not fixed

Encrypted Cipher Block Chaining MAC (ECBC-MAC)

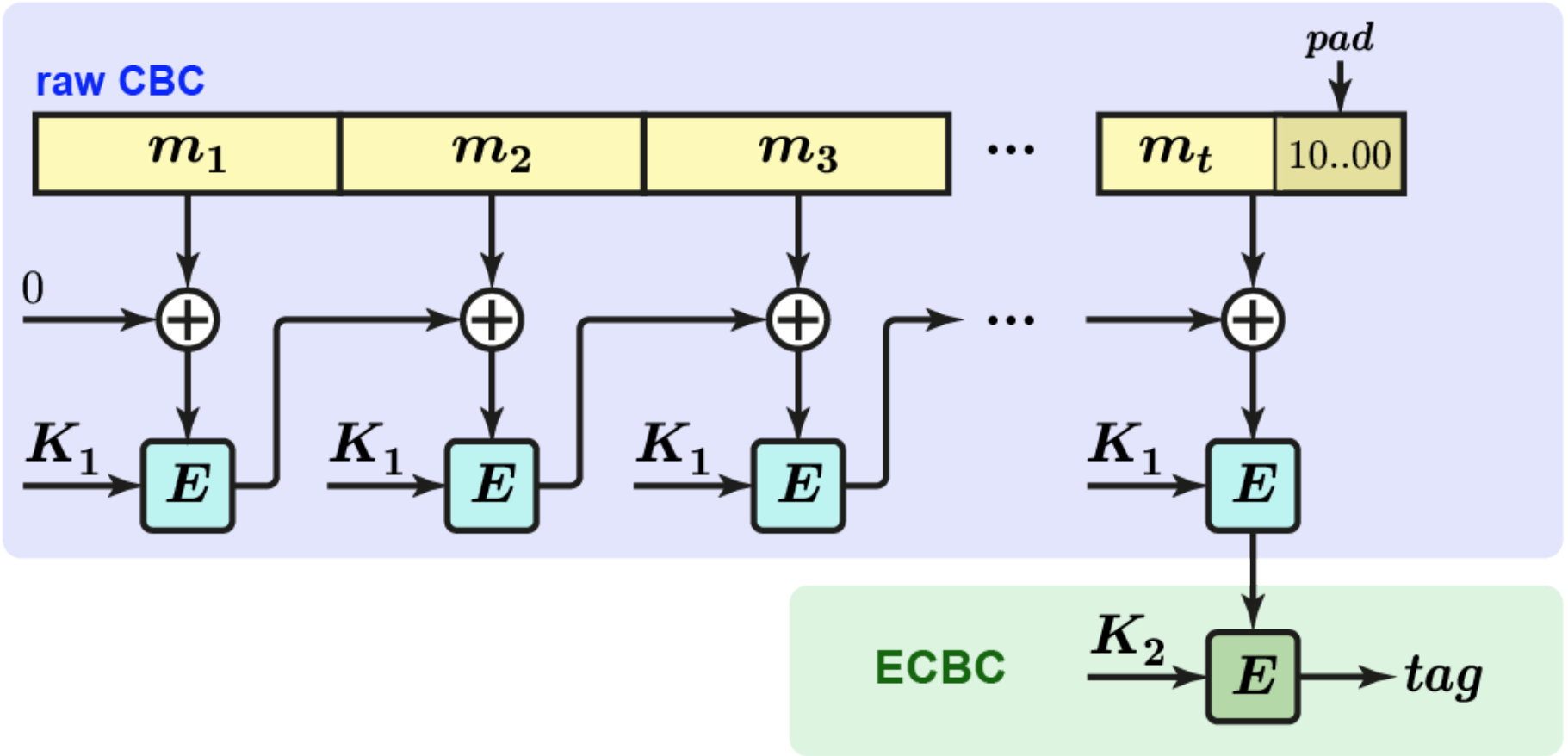
CBC-MAC is not secure when message length is not fixed

To overcome this, we encrypt the output of the last block with different key. Thus the construction is called Encrypted-CBC MAC (ECBC-MAC)

Encrypted Cipher Block Chaining MAC (ECBC-MAC)

CBC-MAC is not secure when message length is not fixed

To overcome this, we encrypt the output of the last block with different key. Thus the construction is called Encrypted-CBC MAC (ECBC-MAC)



Thanks

References

- Antoine Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In Matthew K. Franklin, editor, CRYPTO 2004, Volume 3152 of LNCS, pages 306-316. Springer, 2004.
- John Kelsey and Bruce Schneier. Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In Ronald Cramer, editor, EUROCRYPT 2005, Volume 3494 of LNCS, pages 474-490. Springer, 2005.
- Mihir Bellare and Oded Goldreich and Anton Mityagin. The Power of Verification Queries in Message Authentication and Authenticated Encryption. http://eprint.iacr.org/2004/309, Volume 2004, pages 309, 2004.
- Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. IACR Cryptology ePrint Archive, 2004:332, 2004

References


E. Biham, O. Dunkelman, A Framework for Iterative Hash Functions HAIFA, Cryptology ePrint Archive, Report 2007/278, http://eprint.iacr.org/2007/278 (August 2425 2006)


Bouillaguet, C., Fouque, P.: Practical hash functions constructions resistant to generic second preimage attacks beyond the birthday bound (2010), submitted to Information Processing Letters