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 10

Cryptographic Hash

Introduction

- Map a long input string to a shorter output string called a digest.

- Map a long input string to a shorter output string called a digest.
- Primary requirement: Avoid collisions.

Introduction

- Map a long input string to a shorter output string called a digest.
- Primary requirement: Avoid collisions.
- Note: Collisions do exists, but finding it should be hard.

Introduction

- Map a long input string to a shorter output string called a digest.
- Primary requirement: Avoid collisions.
- Note: Collisions do exists, but finding it should be hard.
- Collision-resistant hash functions have numerous uses, e.g., digital signature schemes, H-MAC etc.

Introduction

- Can be viewed as lying between the worlds of private and public-key cryptography (PKC).

Introduction

- Can be viewed as lying between the worlds of private and public-key cryptography (PKC).
- Usually constructed using symmetric-key techniques.

Introduction

- Can be viewed as lying between the worlds of private and public-key cryptography (PKC).
- Usually constructed using symmetric-key techniques.
- From a theoretical point of view, the existence of collision-resistant hash functions appears to represent a qualitatively stronger assumption than the existence of PRF.

Introduction

- Can be viewed as lying between the worlds of private and public-key cryptography (PKC).
- Usually constructed using symmetric-key techniques.
- From a theoretical point of view, the existence of collision-resistant hash functions appears to represent a qualitatively stronger assumption than the existence of PRF.
- But weaker assumption than the existence of PK encryptions.

Introduction

- Can be viewed as lying between the worlds of private and public-key cryptography (PKC).
- Usually constructed using symmetric-key techniques.
- From a theoretical point of view, the existence of collision-resistant hash functions appears to represent a qualitatively stronger assumption than the existence of PRF.
- But weaker assumption than the existence of PK encryptions.
- They have become ubiquitous in cryptography.

Introduction

- Can be viewed as lying between the worlds of private and public-key cryptography (PKC).
- Usually constructed using symmetric-key techniques.
- From a theoretical point of view, the existence of collision-resistant hash functions appears to represent a qualitatively stronger assumption than the existence of PRF.
- But weaker assumption than the existence of PK encryptions.
- They have become ubiquitous in cryptography.
- Are often used in scenarios that require properties much stronger than collision resistance.

Data Integrity

- Can provide data integrity.
- It is used to construct a short fingerprint of some data.

Data Integrity

- Can provide data integrity.
- It is used to construct a short fingerprint of some data.
- If the data is altered then the fingerprint is no longer valid.
- For data stored in an insecure place, its integrity can be checked from time to time.

Data Integrity

- Can provide data integrity.
- It is used to construct a short fingerprint of some data.
- If the data is altered then the fingerprint is no longer valid.
- For data stored in an insecure place, its integrity can be checked from time to time.
- Assume that the data x is of arbitrary length.
- Compute $y = H(x)$ (message digest), typically short binary strings (e.g. 160 bits).
- Store y in a secure place.

Data Integrity

- Can provide data integrity.
- It is used to construct a short fingerprint of some data.
- If the data is altered then the fingerprint is no longer valid.
- For data stored in an insecure place, its integrity can be checked from time to time.
- Assume that the data x is of arbitrary length.
- Compute $y = H(x)$ (message digest), typically short binary strings (e.g. 160 bits).
- Store y in a secure place.
- Suppose x is changed to x'.
- Then, as H is collision resistant, $y \neq y' = H(x')$.
- Therefore ensuring data integrity.

- The example assumed the existence of a single hash function.
- It is often useful to study families of keyed hash functions.

Authentication

- The example assumed the existence of a single hash function.
- It is often useful to study families of keyed hash functions.
- Alice:
  - Computes the authentication tag, $y = H(k, x)$.
  - Sends (x, y) to Bob.

Authentication

- The example assumed the existence of a single hash function.
- It is often useful to study families of keyed hash functions.
- Alice:
  - Computes the authentication tag, $y = H(k, x)$.
  - Sends (x, y) to Bob.
- Bob: Verify $y \stackrel{?}{=} H(k, x)$.
  - If the condition holds, then Bob is confident that neither x nor y was altered by an adversary, provided that the hash family is secure.

Authentication

- The example assumed the existence of a single hash function.
- It is often useful to study families of keyed hash functions.
- Alice:
  - Computes the authentication tag, $y = H(k, x)$.
  - Sends (x, y) to Bob.
- Bob: Verify $y \stackrel{?}{=} H(k, x)$.
  - If the condition holds, then Bob is confident that neither x nor y was altered by an adversary, provided that the hash family is secure.
- Authentication: Bob is assured that the message x originates from Alice.

Definition: Keyed hash functions

A keyed hash function H is a deterministic algorithm that takes two inputs, a key k and a message m; its output $t := H(k, x)$ is called a digest.

Definition: Keyed hash functions

A keyed hash function H is a deterministic algorithm that takes two inputs, a key k and a message m; its output $t := H(k, x)$ is called a digest.
There are associated spaces: the keyspace $\mathcal{K}$, in which k lies, a message space $\mathcal{M}$, in which m lies, and the digest space $\mathcal{T}$, in which t lies. We say that the hash function H is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$.

Hash Function

Definition: Keyed hash functions

A keyed hash function H is a deterministic algorithm that takes two inputs, a key k and a message m; its output $t := H(k, x)$ is called a digest.
There are associated spaces: the keyspace $\mathcal{K}$, in which k lies, a message space $\mathcal{M}$, in which m lies, and the digest space $\mathcal{T}$, in which t lies. We say that the hash function H is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$.

Security Notions

- Collision Reistance,
- Second-Preimage Resistance, and
- Preimage Resistance.

Collision Resistance Attack Game

For a keyed hash function H defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows

- The challenger picks a random $k \xleftarrow{R} \mathcal{K}$ and sends k to $\mathcal{A}$.
- $\mathcal{A}$ outputs two distinct messages $m, m' \in \mathcal{M}$.

Collision Reistance

Collision Resistance Attack Game

For a keyed hash function H defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows

- The challenger picks a random $k \xleftarrow{R} \mathcal{K}$ and sends k to $\mathcal{A}$.
- $\mathcal{A}$ outputs two distinct messages $m, m' \in \mathcal{M}$.

$\mathcal{A}$ Wins

We say that $\mathcal{A}$ wins the above game if $H(k, m) = H(k, m')$. We define $\mathcal{A}$'s advantage with respect to H, denoted $\mathsf{CRadv}[\mathcal{A}, H]$, as the probability that A wins the game.

Collision Reistance

Collision Resistance Attack Game

For a keyed hash function H defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows

- The challenger picks a random $k \xleftarrow{R} \mathcal{K}$ and sends k to $\mathcal{A}$.
- $\mathcal{A}$ outputs two distinct messages $m, m' \in \mathcal{M}$.

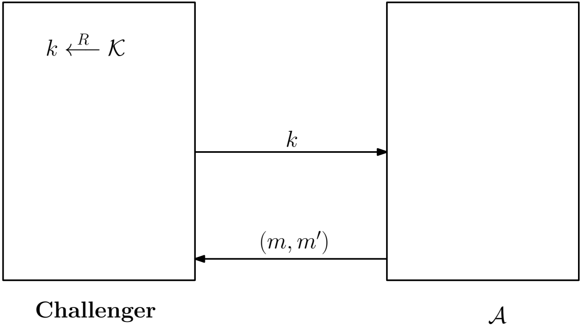
$\mathcal{A}$ Wins

We say that $\mathcal{A}$ wins the above game if $H(k,m) = H(k,m')$. We define $\mathcal{A}$'s advantage with respect to H, denoted $\mathsf{CRadv}[\mathcal{A}, H]$, as the probability that A wins the game.

H is Collision Resistant

We say that H is Collision Resistant if for all PPT adversary $\mathcal{A}$, $\mathsf{CRadv}[\mathcal{A}, H]$ is negligible.

Collision Resistance Attack Game



Second-Preimage Resistance

Second-Preimage Resistance Attack Game

For a keyed hash function H defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows

- The challenger picks a random $k \xleftarrow{R} \mathcal{K}$, $m \xleftarrow{R} \mathcal{M}$ and sends (k, m) to $\mathcal{A}$.
- $\mathcal{A}$ outputs a message $m' \in \mathcal{M}$.

Second-Preimage Resistance

Second-Preimage Resistance Attack Game

For a keyed hash function H defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows

- The challenger picks a random $k \xleftarrow{R} \mathcal{K}$, $m \xleftarrow{R} \mathcal{M}$ and sends (k, m) to $\mathcal{A}$.
- $\mathcal{A}$ outputs a message $m' \in \mathcal{M}$.

$\mathcal{A}$ Wins

We say that $\mathcal{A}$ wins the above game if $H(k, m) = H(k, m')$ and $m \neq m'$. We define $\mathcal{A}$'s advantage with respect to H, denoted $\mathsf{SPRadv}[\mathcal{A}, H]$, as the probability that A wins the game.

Second-Preimage Resistance

Second-Preimage Resistance Attack Game

For a keyed hash function H defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows

- The challenger picks a random $k \xleftarrow{R} \mathcal{K}$, $m \xleftarrow{R} \mathcal{M}$ and sends (k, m) to $\mathcal{A}$.
- $\mathcal{A}$ outputs a message $m' \in \mathcal{M}$.

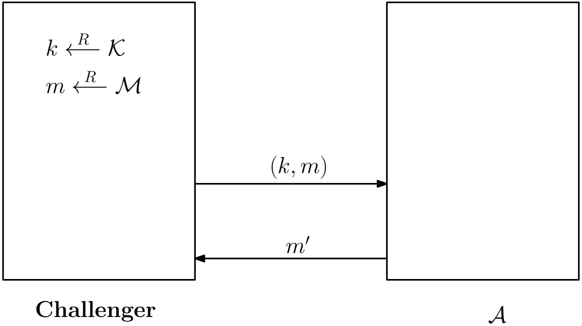
$\mathcal{A}$ Wins

We say that $\mathcal{A}$ wins the above game if $H(k,m) = H(k,m')$ and $m \neq m'$. We define $\mathcal{A}$'s advantage with respect to H, denoted $\mathsf{SPRadv}[\mathcal{A}, H]$, as the probability that A wins the game.

H is Second-Preimage Resistant

We say that H is Second-Preimage Resistant if for all PPT adversary $\mathcal{A}$, $\mathsf{SPRadv}[\mathcal{A}, H]$ is negligible.

Second-Preimage Resistance



Preimage Resistance

Preimage Resistance Attack Game

For a keyed hash function H defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows

- The challenger picks a random $k \xleftarrow{R} \mathcal{K}$, $m \xleftarrow{R} \mathcal{M}$ and computes $t \longleftarrow H(k, m)$.
- The challenger sends (k, t) to $\mathcal{A}$.
- $\mathcal{A}$ outputs a message $m' \in \mathcal{M}$.

Preimage Resistance

Preimage Resistance Attack Game

For a keyed hash function H defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows

- The challenger picks a random $k \xleftarrow{R} \mathcal{K}$, $m \xleftarrow{R} \mathcal{M}$ and computes $t \longleftarrow H(k, m)$.
- The challenger sends (k, t) to $\mathcal{A}$.
- $\mathcal{A}$ outputs a message $m' \in \mathcal{M}$.

$\mathcal{A}$ Wins

We say that $\mathcal{A}$ wins the above game if $H(k, m') = t$. We define $\mathcal{A}$'s advantage with respect to H, denoted $\mathsf{OWadv}[\mathcal{A}, H]$, as the probability that A wins the game.

Preimage Resistance Attack Game

For a keyed hash function H defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $\mathcal{A}$, the attack game runs as follows

- The challenger picks a random $k \xleftarrow{R} \mathcal{K}$, $m \xleftarrow{R} \mathcal{M}$ and computes $t \longleftarrow H(k,m)$.
- The challenger sends (k,t) to $\mathcal{A}$.
- $\mathcal{A}$ outputs a message $m' \in \mathcal{M}$.

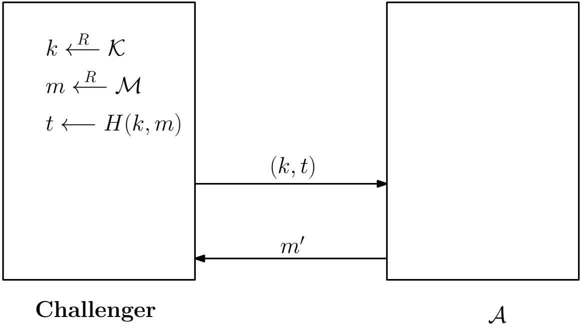
$\mathcal{A}$ Wins

We say that $\mathcal{A}$ wins the above game if $H(k,m') = t$. We define $\mathcal{A}$'s advantage with respect to H, denoted $\mathsf{OWadv}[\mathcal{A}, H]$, as the probability that A wins the game.

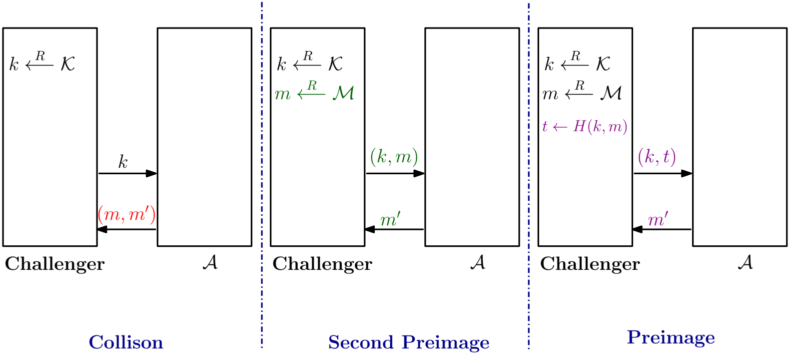
H is Preimage Resistant

We say that H is Preimage Resistant if for all PPT adversary $\mathcal{A}$, $\mathsf{OWadv}[\mathcal{A}, H]$ is negligible.

Preimage Resistance



Collision Vs. Second Preimage Vs. Preimage



Collision Vs. Second Preimage Vs. Preimage

- Let H is compressing, meaning that the domain of H is bigger than its range.
- Let $|\mathcal{M}| \geqslant s \cdot |\mathcal{T}|$ for some compression factor $s > 1$.
- If s is super-poly

$$\text{collision resistant} \Rightarrow \text{2nd-preimage resistant} \Rightarrow \text{one-way}.$$

- The converse is not true.

Collision Vs. Second Preimage

Lemma

Any hash function that is collision resistant is also second preimage resistant.

Collision Vs. Second Preimage

Lemma

Any hash function that is collision resistant is also second preimage resistant.

Proof

- We prove the contrapositive statement.
- Let the hash function is not second preimage resistant.

Collision Vs. Second Preimage

Lemma

Any hash function that is collision resistant is also second preimage resistant.

Proof

- We prove the contrapositive statement.
- Let the hash function is not second preimage resistant.
- Not second preimage resistant $\Rightarrow$ given a uniform pair (k, m), an adversary $\mathcal{A}$ can find $m' \neq m$, for which $H(k, m') = H(k, m)$.

Collision Vs. Second Preimage

Lemma

Any hash function that is collision resistant is also second preimage resistant.

Proof

- We prove the contrapositive statement.
- Let the hash function is not second preimage resistant.
- Not second preimage resistant $\Rightarrow$ given a uniform pair (k, m), an adversary $\mathcal{A}$ can find $m' \neq m$, for which $H(k, m') = H(k, m)$.
- Finding collision:
  - Given a random k.
  - Select a uniform m.

Collision Vs. Second Preimage

Lemma

Any hash function that is collision resistant is also second preimage resistant.

Proof

- We prove the contrapositive statement.
- Let the hash function is not second preimage resistant.
- Not second preimage resistant $\Rightarrow$ given a uniform pair (k, m), an adversary $\mathcal{A}$ can find $m' \neq m$, for which $H(k, m') = H(k, m)$.
- Finding collision:
  - Given a random k.
  - Select a uniform m.
  - Use $\mathcal{A}$ for second preimage to find m', s.t., $m \neq m'$ and $H(k, m') = H(k, m)$.

Collision Vs. Second Preimage

Lemma

Any hash function that is collision resistant is also second preimage resistant.

Proof

- We prove the contrapositive statement.
- Let the hash function is not second preimage resistant.
- Not second preimage resistant $\Rightarrow$ given a uniform pair (k,m), an adversary $\mathcal{A}$ can find $m' \neq m$, for which $H(k,m') = H(k,m)$.
- Finding collision:
  - Given a random k.
  - Select a uniform m.
  - Use $\mathcal{A}$ for second preimage to find m', s.t., $m \neq m'$ and $H(k,m') = H(k,m)$.
  - Return (m,m').

Collision Vs. Second Preimage

Converse

- A hash function can be 2nd-preimage resistant, but not collision resistant.

Collision Vs. Second Preimage

Converse

- A hash function can be 2nd-preimage resistant, but not collision resistant.
- For example, SHA1 is believed to be 2nd-preimage resistant even though SHA1 is not collision resistant.

Second Preimage Vs. Preimage

Lemma

Any hash function that is second preimage resistant is also preimage resistant.

Second Preimage Vs. Preimage

Lemma

Any hash function that is second preimage resistant is also preimage resistant.

Proof

- We prove the contrapositive statement.
- Let the hash function is not preimage resistant.

Second Preimage Vs. Preimage

Lemma

Any hash function that is second preimage resistant is also preimage resistant.

Proof

- We prove the contrapositive statement.
- Let the hash function is not preimage resistant.
- Not preimage resistant $\implies$ given a uniform pair (k,t), an adversary $\mathcal{A}$ can find m', for which $H(k,m') = t$.

Second Preimage Vs. Preimage

Lemma

Any hash function that is second preimage resistant is also preimage resistant.

Proof

- We prove the contrapositive statement.
- Let the hash function is not preimage resistant.
- Not preimage resistant $\implies$ given a uniform pair (k,t), an adversary $\mathcal{A}$ can find m', for which $H(k,m') = t$.
- Finding Second Preimage:
  - Given a uniform pair (k,m), compute $t \longleftarrow H(k,m)$.

Second Preimage Vs. Preimage

Lemma

Any hash function that is second preimage resistant is also preimage resistant.

Proof

- We prove the contrapositive statement.
- Let the hash function is not preimage resistant.
- Not preimage resistant $\implies$ given a uniform pair (k,t), an adversary $\mathcal{A}$ can find m', for which $H(k,m') = t$.
- Finding Second Preimage:
  - Given a uniform pair (k,m), compute $t \longleftarrow H(k,m)$.
  - Use $\mathcal{A}$ for preimage to find m', s.t., $H(k,m') = t$.

Second Preimage Vs. Preimage

Lemma

Any hash function that is second preimage resistant is also preimage resistant.

Proof

- We prove the contrapositive statement.
- Let the hash function is not preimage resistant.
- Not preimage resistant $\Rightarrow$ given a uniform pair (k,t), an adversary $\mathcal{A}$ can find m', for which $H(k,m') = t$.
- Finding Second Preimage:
  - Given a uniform pair (k,m), compute $t \longleftarrow H(k,m)$.
  - Use $\mathcal{A}$ for preimage to find m', s.t., $H(k,m') = t$.
  - With high probability, $m \neq m'$.

Second Preimage Vs. Preimage

Lemma

Any hash function that is second preimage resistant is also preimage resistant.

Proof

- We prove the contrapositive statement.
- Let the hash function is not preimage resistant.
- Not preimage resistant $\Rightarrow$ given a uniform pair (k, t), an adversary $\mathcal{A}$ can find m', for which $H(k, m') = t$.
- Finding Second Preimage:
  - Given a uniform pair (k, m), compute $t \longleftarrow H(k, m)$.
  - Use $\mathcal{A}$ for preimage to find m', s.t., $H(k, m') = t$.
  - With high probability, $m \neq m'$.
  - Return m'.

Second Preimage Vs. Preimage

Converse

- Let H be a preimage resistant hash function as $H : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{T}$.

Second Preimage Vs. Preimage

Converse

- Let H be a preimage resistant hash function as $H : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{T}$.
- Define a hash function $H' : \mathcal{K} \times (\mathcal{X} \times \{0,1\}) \longrightarrow \mathcal{T}$ as

$$H'(k,(m,b)) := H(k,m).$$

Second Preimage Vs. Preimage

Converse

- Let H be a preimage resistant hash function as $H : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{T}$.
- Define a hash function $H' : \mathcal{K} \times (\mathcal{X} \times \{0, 1\}) \longrightarrow \mathcal{T}$ as

$$H'(k, (m, b)) := H(k, m).$$

- H' is not second preimage resistant.

The Merkle-Damgård Transform

Motivation

- Hash functions are often constructed by first designing a collision-resistant compression function handling fixed-length inputs.

Motivation

- Hash functions are often constructed by first designing a collision-resistant compression function handling fixed-length inputs.
- Domain extension is used to handle arbitrary-length inputs.

Motivation

- Hash functions are often constructed by first designing a collision-resistant compression function handling fixed-length inputs.
- Domain extension is used to handle arbitrary-length inputs.
- A common approach for extending a compression function to a full-fledged hash function, while maintaining the collision-resistance property of the former.

Motivation

- Hash functions are often constructed by first designing a collision-resistant compression function handling fixed-length inputs.
- Domain extension is used to handle arbitrary-length inputs.
- A common approach for extending a compression function to a full-fledged hash function, while maintaining the collision-resistance property of the former.
- It is used extensively in practice.

Motivation

- Hash functions are often constructed by first designing a collision-resistant compression function handling fixed-length inputs.
- Domain extension is used to handle arbitrary-length inputs.
- A common approach for extending a compression function to a full-fledged hash function, while maintaining the collision-resistance property of the former.
- It is used extensively in practice.
- Examples: MD5 and the SHA family.

The Merkle-Damgård Transform

Motivation

- Hash functions are often constructed by first designing a collision-resistant compression function handling fixed-length inputs.
- Domain extension is used to handle arbitrary-length inputs.
- A common approach for extending a compression function to a full-fledged hash function, while maintaining the collision-resistance property of the former.
- It is used extensively in practice.
- Examples: MD5 and the SHA family.
- Theoretical point of view: Compressing by a single bit is as easy (or as hard) as compressing by an arbitrary amount.

Assumptions

- Let $\hat{h}$ be a keyed hash function defined over $(\mathcal{K}, \mathcal{X}, \mathcal{T})$ as $\hat{h} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{T}$.

The Merkle-Damgård Transform

Assumptions

- Let $\hat{h}$ be a keyed hash function defined over $(\mathcal{K}, \mathcal{X}, \mathcal{T})$ as $\hat{h} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{T}$.
- Let $\hat{h}(k, \cdot)$ be a keyed hash function whose key k is publicly known and fixed.

Assumptions

- Let $\hat{h}$ be a keyed hash function defined over $(\mathcal{K},\mathcal{X},\mathcal{T})$ as $\hat{h}:\mathcal{K}\times\mathcal{X}\to\mathcal{T}$.
- Let $\hat{h}(k,\cdot)$ be a keyed hash function whose key k is publicly known and fixed.
  - Now $\hat{h}(k,\cdot)$ is same as a hash function without a key as $\hat{h}_k(\cdot)$.
  - $h_k(\cdot)$, $h(\cdot)$ in short, can be considered as keyless-hash function defined over $(\mathcal{X},\mathcal{T})$.

The Merkle-Damgård Transform

Assumptions

- Let $\hat{h}$ be a keyed hash function defined over $(\mathcal{K}, \mathcal{X}, \mathcal{T})$ as $\hat{h} : \mathcal{K} \times \mathcal{X} \to \mathcal{T}$.
- Let $\hat{h}(k, \cdot)$ be a keyed hash function whose key k is publicly known and fixed.
  - Now $\hat{h}(k, \cdot)$ is same as a hash function without a key as $\hat{h}_k(\cdot)$.
  - $h_k(\cdot)$, $h(\cdot)$ in short, can be considered as keyless-hash function defined over $(\mathcal{X}, \mathcal{T})$.

Assumptions

- Let $h : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathcal{X}$ be a keyless hash function defined over $(\mathcal{X} \times \mathcal{Y}, \mathcal{X})$.

Assumptions

- Let $\hat{h}$ be a keyed hash function defined over $(\mathcal{K}, \mathcal{X}, \mathcal{T})$ as $\hat{h} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{T}$.
- Let $\hat{h}(k, \cdot)$ be a keyed hash function whose key k is publicly known and fixed.
  - Now $\hat{h}(k, \cdot)$ is same as a hash function without a key as $\hat{h}_k(\cdot)$.
  - $h_k(\cdot)$, $h(\cdot)$ in short, can be considered as keyless-hash function defined over $(\mathcal{X}, \mathcal{T})$.

Assumptions

- Let $h : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathcal{X}$ be a keyless hash function defined over $(\mathcal{X} \times \mathcal{Y}, \mathcal{X})$.
- Assume that $\mathcal{Y}$ (data blocks) is of the form $\{0, 1\}^{\ell}$ for some ℓ.

The Merkle-Damgård Transform

Assumptions

- Let $\hat{h}$ be a keyed hash function defined over $(\mathcal{K}, \mathcal{X}, \mathcal{T})$ as $\hat{h} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{T}$.
- Let $\hat{h}(k, \cdot)$ be a keyed hash function whose key k is publicly known and fixed.
  - Now $\hat{h}(k, \cdot)$ is same as a hash function without a key as $\hat{h}_k(\cdot)$.
  - $h_k(\cdot)$, $h(\cdot)$ in short, can be considered as keyless-hash function defined over $(\mathcal{X}, \mathcal{T})$.

Assumptions

- Let $h : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathcal{X}$ be a keyless hash function defined over $(\mathcal{X} \times \mathcal{Y}, \mathcal{X})$.
- Assume that $\mathcal{Y}$ (data blocks) is of the form $\{0,1\}^{\ell}$ for some ℓ.
- Typically $\mathcal{X}$ is of the form $\{0,1\}^{n}$ for some n.

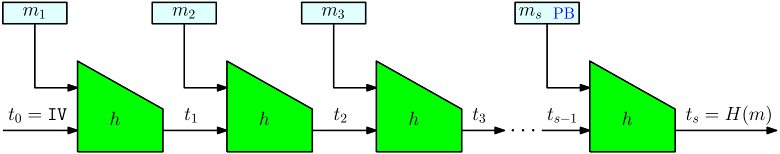
Assumptions

- Let $\hat{h}$ be a keyed hash function defined over $(\mathcal{K}, \mathcal{X}, \mathcal{T})$ as $\hat{h} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{T}$.
- Let $\hat{h}(k, \cdot)$ be a keyed hash function whose key k is publicly known and fixed.
  - Now $\hat{h}(k, \cdot)$ is same as a hash function without a key as $\hat{h}_k(\cdot)$.
  - $h_k(\cdot)$, $h(\cdot)$ in short, can be considered as keyless-hash function defined over $(\mathcal{X}, \mathcal{T})$.

Assumptions

- Let $h : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathcal{X}$ be a keyless hash function defined over $(\mathcal{X} \times \mathcal{Y}, \mathcal{X})$.
- Assume that $\mathcal{Y}$ (data blocks) is of the form $\{0, 1\}^{\ell}$ for some ℓ.
- Typically $\mathcal{X}$ is of the form $\{0, 1\}^{n}$ for some n.
- The Merkle-Damgård function derived from h, denoted H_{MD} defined over $\left(\{0, 1\}^{\leqslant L}, \mathcal{X}\right)$.

The Merkle-Damgård Transform



The Merkle-Damgård Transform

The Merkle-Damgård Paradigm

input: $M \in \{0,1\}^{\leqslant L}$
output: a tag in $\mathcal{X}$

1. $\hat{M} \longleftarrow M \| \mathtt{PB}$ //pad with $\mathtt{PB}$ to ensure that the length of M is a multiple of ℓ bits
2. partition $\hat{M}$ into consecutive ℓ-bit blocks so that
3. $\hat{M} = m_1 \| m_2 \| \cdots \| m_s$ where $m_1, m_2, \ldots, m_s \in \{0,1\}^{\ell}$.
4. $t_0 \longleftarrow IV \in \mathcal{X}$
5. for $i = 1$ to s do:
6. $t_i \longleftarrow h(t_{i-1}, m_i)$
7. output t_s

The Merkle-Damgård Paradigm

- The function SHA256 is a Merkle-Damgård function where $\ell = 512$ and $n = 256$.

The Merkle-Damgård Paradigm

- The function SHA256 is a Merkle-Damgård function where $\ell = 512$ and $n = 256$.
- The hash function h is called the compression function of H.

The Merkle-Damgård Transform

The Merkle-Damgård Paradigm

- The function SHA256 is a Merkle-Damgård function where $\ell = 512$ and $n = 256$.
- The hash function h is called the compression function of H.
- The constant IV is called the initial value and is fixed to some pre-specified value.

The Merkle-Damgård Paradigm

- The function SHA256 is a Merkle-Damgård function where $\ell = 512$ and $n = 256$.
- The hash function h is called the compression function of H.
- The constant IV is called the initial value and is fixed to some pre-specified value.
  - Could take $IV = 0^n$.
  - For SHA256,

$$\begin{aligned} \texttt{IV} \quad := \quad & 6A09E667\ BB67AE85\ 3C6EF372\ A54FF53A\backslash \\ & 510E527F\ 9B05688C\ 1F83D9AB\ 5BE0CD19. \end{aligned}$$

The Merkle-Damgård Transform

The Merkle-Damgård Paradigm

- The function SHA256 is a Merkle-Damgård function where $\ell = 512$ and $n = 256$.
- The hash function h is called the compression function of H.
- The constant IV is called the initial value and is fixed to some pre-specified value.
  - Could take $IV = 0^n$.
  - For SHA256,

$$\begin{aligned} \texttt{IV} \quad := \quad & 6A09E667\ BB67AE85\ 3C6EF372\ A54FF53A\backslash \\ & 510E527F\ 9B05688C\ 1F83D9AB\ 5BE0CD19. \end{aligned}$$

- The variables $m_1, m_2, \ldots, m_s$ are called message blocks.

The Merkle-Damgård Paradigm

- The function SHA256 is a Merkle-Damgård function where $\ell = 512$ and $n = 256$.
- The hash function h is called the compression function of H.
- The constant IV is called the initial value and is fixed to some pre-specified value.
  - Could take $IV = 0^n$.
  - For SHA256,

$$\begin{aligned} \texttt{IV} \quad := \quad & 6A09E667\ BB67AE85\ 3C6EF372\ A54FF53A\backslash \\ & 510E527F\ 9B05688C\ 1F83D9AB\ 5BE0CD19. \end{aligned}$$

- The variables $m_1, m_2, \ldots, m_s$ are called message blocks.
- The variables $t_0, t_1, \ldots, t_s \in \mathcal{X}$ are called chaining variables.

The Merkle-Damgård Paradigm

- The string PB is called the padding block. It is appended to the message to ensure that the message length is a multiple of ℓ bits.

The Merkle-Damgård Paradigm

- The string $\texttt{PB}$ is called the padding block. It is appended to the message to ensure that the message length is a multiple of ℓ bits.
  - $\texttt{PB} := 100\ldots00 \| \langle s \rangle$.
  - $\langle s \rangle$ encodes the number of ℓ-bit blocks in M.

The Merkle-Damgård Paradigm

- The string PB is called the padding block. It is appended to the message to ensure that the message length is a multiple of ℓ bits.
    - $\mathtt{PB} := 100\ldots00\|\langle s\rangle$.
    - $\langle s\rangle$ encodes the number of ℓ-bit blocks in M.
    - If the message length is such that there is no space for PB in the last block (for example, if the message length happens to be a multiple of ℓ), then an additional block is added just for the padding block.

Theorem

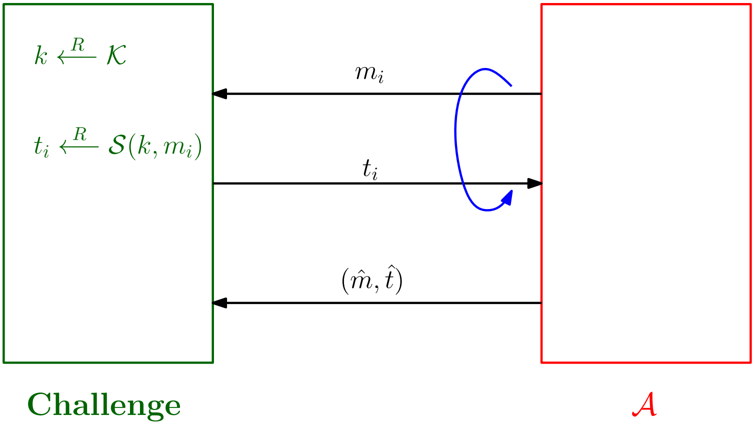
Let L be a poly-bounded length parameter and let h be a collision resistant hash function defined over $(\mathcal{X} \times \mathcal{Y}, \mathcal{X})$. Then the Merkle-Damgård hash function H_{MD} derived from h, defined over $(\{0,1\}^{\leqslant L}, \mathcal{X})$, is collision resistant.

The Merkle-Damgård Transform

Theorem

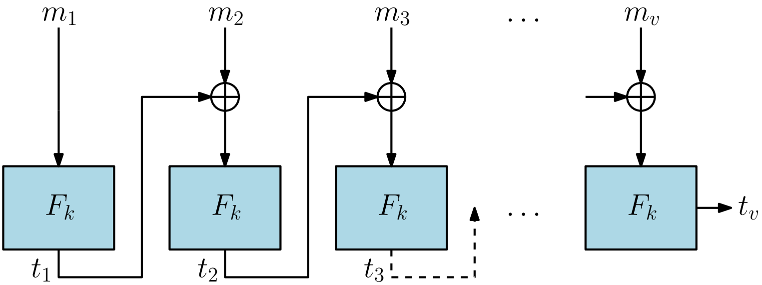
Let L be a poly-bounded length parameter and let h be a collision resistant hash function defined over $(\mathcal{X} \times \mathcal{Y}, \mathcal{X})$. Then the Merkle-Damgård hash function H_{MD} derived from h, defined over $(\{0,1\}^{\leqslant L}, \mathcal{X})$, is collision resistant.
In particular, for every collision finder $\mathcal{A}$ attacking H_{MD} there exists a collision finder $\mathcal{A}$ attacking h, where $\mathcal{B}$ is an elementary wrapper around $\mathcal{A}$, such that

$$\mathsf{CRadv}[\mathcal{A}, H_{\text{MD}}] = \mathsf{CRadv}[\mathcal{B}, h].$$



MAC Attcak Game

Recap: CBC-MAC



CBC-MAC: Problems

- Secure if queries are prefix-free.

CBC-MAC: Problems

- Secure if queries are prefix-free.
- But in the actual MAC attack game, it is not secure.

CBC-MAC: Problems

- Secure if queries are prefix-free.
- But in the actual MAC attack game, it is not secure.
    - The condition of prefix-free is not required there.

CBC-MAC: Problems

- Secure if queries are prefix-free.
- But in the actual MAC attack game, it is not secure.
  - The condition of prefix-free is not required there.
  - We can make a forged tag.

Attack on CBC-MAC

- Adversary makes a query on arbitrary $m_1 \in \mathcal{X}$.

CBC-MAC: Problems

- Secure if queries are prefix-free.
- But in the actual MAC attack game, it is not secure.
  - The condition of prefix-free is not required there.
  - We can make a forged tag.

Attack on CBC-MAC

- Adversary makes a query on arbitrary $m_1 \in \mathcal{X}$.
- Challenger returns (m_1, t_v).

CBC-MAC: Problems

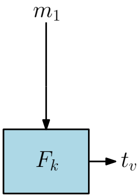
- Secure if queries are prefix-free.
- But in the actual MAC attack game, it is not secure.
  - The condition of prefix-free is not required there.
  - We can make a forged tag.

Attack on CBC-MAC

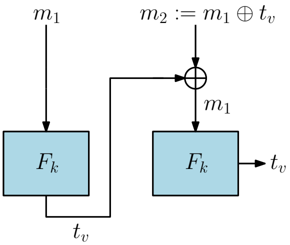
- Adversary makes a query on arbitrary $m_1 \in \mathcal{X}$.
- Challenger returns (m_1, t_v).
- Adversary returns $(\hat{m}, \hat{t})$, where

$$\begin{aligned} \hat{m} &:= m_1 \| (m_1 \oplus t_v) \\ \hat{t} &:= t_v. \end{aligned}$$

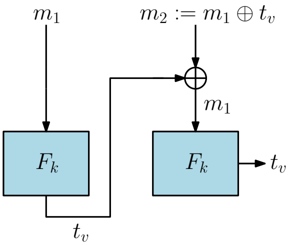
CBC-MAC: Problems



CBC-MAC: Problems



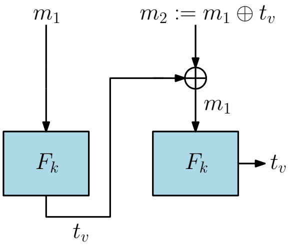
CBC-MAC: Problems



Correctness of the Attack

$$S(k,(m_1 \| m_2)) = F(k, m_2 \oplus F(k,m_1)) = F(k,(m_1 \oplus t_v) \oplus t_v) = F(k,m_1) = t_v.$$

CBC-MAC: Problems



Correctness of the Attack

$$S(k,(m_1 \| m_2)) = F(k, m_2 \oplus F(k,m_1)) = F(k,(m_1 \oplus t_v) \oplus t_v) = F(k,m_1) = t_v.$$

Refusal

- Many practitioners refused to use CBC-MAC.
    - They claim it is also too slow.

- Question 1: Can we use the Merkle-Damgård Construction to Create a MAC?

HMAC

- Question 1: Can we use the Merkle-Damgård Construction to Create a MAC?
- Problem 1: H construction from h is a keyless hash function.

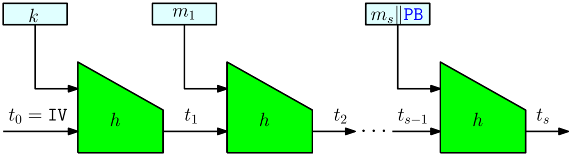
- Question 1: Can we use the Merkle-Damgård Construction to Create a MAC?
- Problem 1: H construction from h is a keyless hash function.
- Question 2: How to incorporate the Key?

- Question 1: Can we use the Merkle-Damgård Construction to Create a MAC?
- Problem 1: H construction from h is a keyless hash function.
- Question 2: How to incorporate the Key?
- A Common Solution: $S(k,m) := H(k\|m)$.

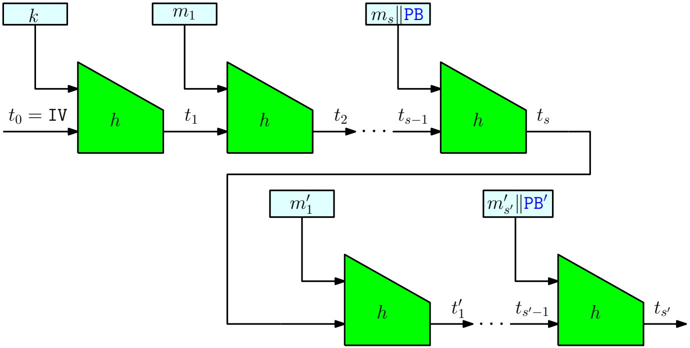
HMAC

- Question 1: Can we use the Merkle-Damgård Construction to Create a MAC?
- Problem 1: H construction from h is a keyless hash function.
- Question 2: How to incorporate the Key?
- A Common Solution: $S(k,m) := H(k\|m)$.
  - We can create a new tag on the message $(m\|PB\|m')$ where m' can be chosen randomly.

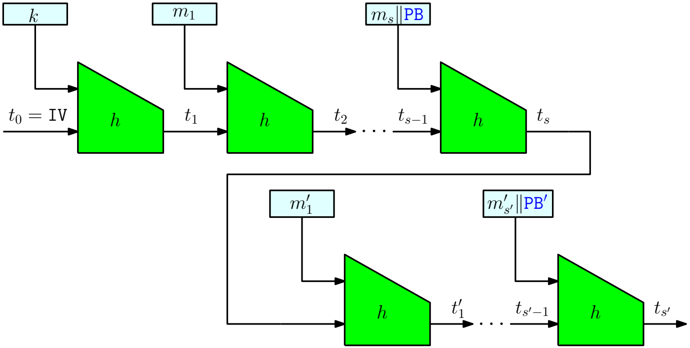
HMAC



HMAC


k
m_1
$m_s \| \mathsf{PB}$
$t_0 = \mathtt{IV}$
h
t_1
h
t_2
$\cdots$
t_{s-1}
h
t_s
m'_1
$m'_{s'} \| \mathsf{PB}'$
h
t'_1
$\cdots$
$t_{s'-1}$
h
$t_{s'}$

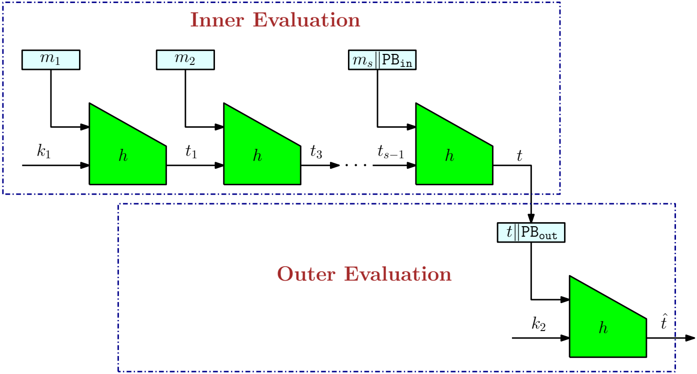
HMAC



HMAC

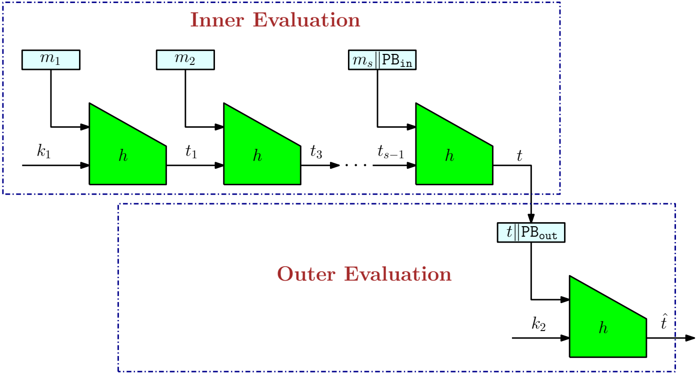
- One standard solution to the problem: A two keyed approach.
- One such standard construction is HMAC.
- Before HMAC, we will discuss something called Nested MAC or NAMC.

NMAC



Inner Evaluation
m_1
m_2
$m_s \| \mathtt{PB_{in}}$
k_1
h
t_1
h
t_3
$\cdots$
t_{s-1}
h
t
$t \| \mathtt{PB_{out}}$
Outer Evaluation
k_2
h
$\hat{t}$

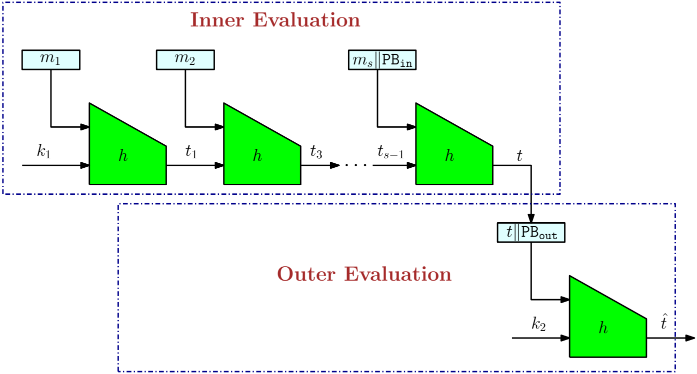
NMAC



NMAC

- A deterministic MAC defined over $\left(\mathcal{K}^2, \{0,1\}^{\leqslant L}, \mathcal{X}\right)$.

NMAC



NMAC

- A deterministic MAC defined over $\left(\mathcal{K}^2, \{0,1\}^{\leqslant L}, \mathcal{X}\right)$.

$$\mathcal{S}((k_1, k_2), m) := H_{k_2}(H_{k_1}(m)),$$

where H_{k_i} is $H(\cdot)$ with $\mathtt{IV} = k_i$.

Fixed length MAC $\mathcal{I}_\circ$

- $\mathcal{I}_\circ$ defined over $(\mathcal{X}, \mathcal{Y}, \mathcal{X})$.

NMAC

Fixed length MAC $\mathcal{I}_\circ$

- $\mathcal{I}_\circ$ defined over $(\mathcal{X}, \mathcal{Y}, \mathcal{X})$.
- The MAC key k is chosen uniformly at random from $\mathcal{K}$.

NMAC

Fixed length MAC $\mathcal{I}_\circ$

- $\mathcal{I}_\circ$ defined over $(\mathcal{X}, \mathcal{Y}, \mathcal{X})$.
- The MAC key k is chosen uniformly at random from $\mathcal{K}$.
- $\mathcal{S}(k, m) := h(k \| m)$.

NMAC

Fixed length MAC $\mathcal{I}_\circ$

- $\mathcal{I}_\circ$ defined over $(\mathcal{X}, \mathcal{Y}, \mathcal{X})$.
- The MAC key k is chosen uniformly at random from $\mathcal{K}$.
- $\mathcal{S}(k, m) := h(k \| m)$.
- Verification is done in natural way.

Theorem

Let H denote the Merkle-Damgård transform applied to h and let $\mathcal{I}_\circ$ denote the fixed length MAC constructed from h. If h is collision resistant and $\mathcal{I}_\circ$ is a secure MAC, then NMAC is existentially unforgeable under an adaptive chosen-message attack (for arbitrary-length messages).

NMAC

Theorem

Let H denote the Merkle-Damgård transform applied to h and let $\mathcal{I}_\circ$ denote the fixed length MAC constructed from h. If h is collision resistant and $\mathcal{I}_\circ$ is a secure MAC, then NMAC is existentially unforgeable under an adaptive chosen-message attack (for arbitrary-length messages).

Proof

- An efficient $\mathcal{A}$ adversary makes queries and based on the received queries, constructs the following lists:

$$Q = \{(m_1, t_1), (m_2, t_2), \ldots\} \text{ and } Q_m = \{m_1, m_2, \ldots\}.$$

NMAC

Theorem

Let H denote the Merkle-Damgård transform applied to h and let $\mathcal{I}_\circ$ denote the fixed length MAC constructed from h. If h is collision resistant and $\mathcal{I}_\circ$ is a secure MAC, then NMAC is existentially unforgeable under an adaptive chosen-message attack (for arbitrary-length messages).

Proof

- An efficient $\mathcal{A}$ adversary makes queries and based on the received queries, constructs the following lists:

$$Q = \{(m_1, t_1), (m_2, t_2), \ldots\} \text{ and } Q_m = \{m_1, m_2, \ldots\}.$$

- After the query phase, adversary outputs a valid (m^*, t^*).

NMAC

Theorem

Let H denote the Merkle-Damgård transform applied to h and let $\mathcal{I}_\circ$ denote the fixed length MAC constructed from h. If h is collision resistant and $\mathcal{I}_\circ$ is a secure MAC, then NMAC is existentially unforgeable under an adaptive chosen-message attack (for arbitrary-length messages).

Proof

- An efficient $\mathcal{A}$ adversary makes queries and based on the received queries, constructs the following lists:

$$Q = \{(m_1, t_1), (m_2, t_2), \ldots\} \text{ and } Q_m = \{m_1, m_2, \ldots\}.$$

- After the query phase, adversary outputs a valid (m^*, t^*).
- Assume $m^* \notin Q_m$.

Proof

- Case 1: Let $m' \in Q_m$ and $H_{k_1}(m^*) = H_{k_1}(m')$.

  1. We have collision for H_{k_1}.

Proof

- Case 1: Let $m' \in Q_m$ and $H_{k_1}(m^*) = H_{k_1}(m')$.
  1. We have collision for H_{k_1}.
  2. h is collision resistant and H_{k_1} is Merkle-Damgård transform applied to h implies H_{k_1} is collision resistant.

Proof

- Case 1: Let $m' \in Q_m$ and $H_{k_1}(m^*) = H_{k_1}(m')$.
  1. We have collision for H_{k_1}.
  2. h is collision resistant and H_{k_1} is Merkle-Damgård transform applied to h implies H_{k_1} is collision resistant.
  3. Both the above facts are contradicting.

Proof

- Case 1: Let $m' \in Q_m$ and $H_{k_1}(m^*) = H_{k_1}(m')$.
    1. We have collision for H_{k_1}.
    2. h is collision resistant and H_{k_1} is Merkle-Damgård transform applied to h implies H_{k_1} is collision resistant.
    3. Both the above facts are contradicting.
- Case 2: Let $H_{k_1}(m^*) \neq H_{k_1}(m)$ for all $m \in Q_m$.

Proof

- Case 1: Let $m' \in Q_m$ and $H_{k_1}(m^*) = H_{k_1}(m')$.
  1. We have collision for H_{k_1}.
  2. h is collision resistant and H_{k_1} is Merkle-Damgård transform applied to h implies H_{k_1} is collision resistant.
  3. Both the above facts are contradicting.
- Case 2: Let $H_{k_1}(m^*) \neq H_{k_1}(m)$ for all $m \in Q_m$.
  1. $Q' = \{H_{k_1}(m) \mid m \in Q_m\}$.

Proof

- Case 1: Let $m' \in Q_m$ and $H_{k_1}(m^*) = H_{k_1}(m')$.
  1. We have collision for H_{k_1}.
  2. h is collision resistant and H_{k_1} is Merkle-Damgård transform applied to h implies H_{k_1} is collision resistant.
  3. Both the above facts are contradicting.
- Case 2: Let $H_{k_1}(m^*) \neq H_{k_1}(m)$ for all $m \in Q_m$.
  1. $Q' = \{H_{k_1}(m) \mid m \in Q_m\}$.
  2. $H_{k_1}(m^*) \notin Q'$.

Proof

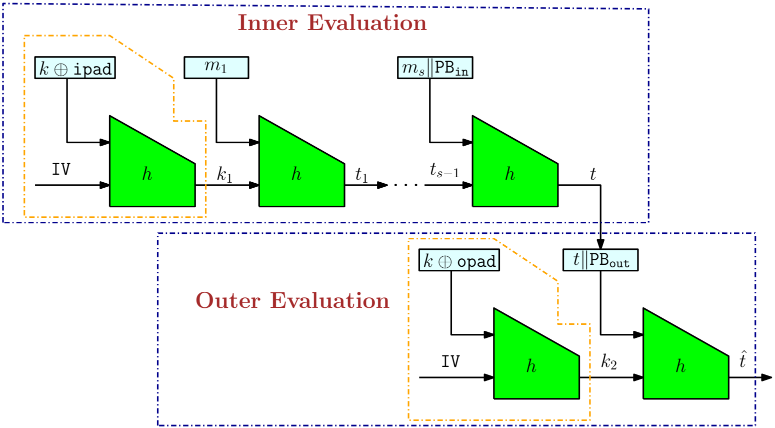
- Case 1: Let $m' \in Q_m$ and $H_{k_1}(m^*) = H_{k_1}(m')$.
  1. We have collision for H_{k_1}.
  2. h is collision resistant and H_{k_1} is Merkle-Damgård transform applied to h implies H_{k_1} is collision resistant.
  3. Both the above facts are contradicting.
- Case 2: Let $H_{k_1}(m^*) \neq H_{k_1}(m)$ for all $m \in Q_m$.
  1. $Q' = \{H_{k_1}(m) \mid m \in Q_m\}$.
  2. $H_{k_1}(m^*) \notin Q'$.
  3. Implies, $\mathcal{A}$ forges a valid MAC for a new message $\left(H_{k_1}(m^*) \| PB\right)$ under the key k_2.

NMAC

Proof

- Case 1: Let $m' \in Q_m$ and $H_{k_1}(m^*) = H_{k_1}(m')$.
  1. We have collision for H_{k_1}.
  2. h is collision resistant and H_{k_1} is Merkle-Damgård transform applied to h implies H_{k_1} is collision resistant.
  3. Both the above facts are contradicting.
- Case 2: Let $H_{k_1}(m^*) \neq H_{k_1}(m)$ for all $m \in Q_m$.
  1. $Q' = \{H_{k_1}(m) \mid m \in Q_m\}$.
  2. $H_{k_1}(m^*) \notin Q'$.
  3. Implies, $\mathcal{A}$ forges a valid MAC for a new message $\left(H_{k_1}(m^*) \| PB\right)$ under the key k_2.
  4. Contradicts the fact $\mathcal{I}_\circ$ is a secure MAC.

HMAC



Inner Evaluation
$k \oplus$ ipad
m_1
$m_s \| \mathrm{PB_{in}}$
IV
h
k_1
h
t_1
$\cdots$
t_{s-1}
h
t
$t \| \mathrm{PB_{out}}$
$k \oplus$ opad
Outer Evaluation
IV
h
k_2
h
$\hat{t}$

HMAC

- A deterministic MAC defined over $(\mathcal{Y}, \{0,1\}^{\leqslant L}, \mathcal{X})$.

HMAC

- A deterministic MAC defined over $(\mathcal{Y}, \{0,1\}^{\leqslant L}, \mathcal{X})$.
- $k_1 = k \oplus \texttt{ipad}$ and $k_2 = k \oplus \texttt{opad}$.

HMAC

HMAC

- A deterministic MAC defined over $(\mathcal{Y}, \{0,1\}^{\leqslant L}, \mathcal{X})$.
- $k_1 = k \oplus \texttt{ipad}$ and $k_2 = k \oplus \texttt{opad}$.
- $\texttt{ipad}$ = the byte 0x36 repeated $\frac{\ell}{8}$ times.
- $\texttt{opad}$ = the byte 0x5C repeated $\frac{\ell}{8}$ times.

Theorem

Let us define $G(k)$ as

$$G(k) := h(\mathtt{IV} \| k \oplus \mathtt{ipad}) \| h(\mathtt{IV} \| k \oplus \mathtt{opad}) = k_1 \| k_2.$$

If h is collision resistant, and $\mathcal{I}_\circ$ is a secure MAC and if G is a pseudorandom generator, then HMAC is existentially unforgeable under an adaptive chosen-message attack (for arbitrary-length messages).

End