Introduction to Cryptology

7.1 - Hash functions: Definitions

Federico Pintore
Mathematical Institute, University of Oxford (UK)

UNIVERSITY OF OXFORD

Michaelmas term 2020

Introduction

Informally speaking, hash functions take long bit strings and output shorter bit strings, called digests.

- They are used almost everywhere in Cryptography.

Introduction

Informally speaking, hash functions take long bit strings and output shorter bit strings, called digests.

- They are used almost everywhere in Cryptography.
- Treating hash functions as truly random functions makes proving the security of some schemes achievable.
  - To evaluate a hash function, a random oracle must be queried;
  - a debate/controversy over the soundness of the Random Oracle Model (ROM).

Keyed Hash Functions

Definition

A keyed hash function with output length $\ell(n)$ is a pair

$$(\mathrm{KeyGen}, H)$$

of two PPT algorithms, defined as follows:

- $s \leftarrow \mathrm{KeyGen}(n)$: *it takes a security parameter n and outputs a key s.*
- $H^s(x) \leftarrow H(s, x)$: *on input a key s and a bit string $x \in \{0,1\}^*$, it outputs a bit string $H^s(x) \in \{0,1\}^{\ell(n)}$.*

Keyed Hash Functions

Definition

A keyed hash function with output length $\ell(n)$ is a pair

$$(\mathrm{KeyGen}, H)$$

of two PPT algorithms, defined as follows:

- $s \leftarrow \mathrm{KeyGen}(n)$: *it takes a security parameter n and outputs a key s.*
- $H^s(x) \leftarrow H(s, x)$: *on input a key s and a bit string $x \in \{0,1\}^*$, it outputs a bit string $H^s(x) \in \{0,1\}^{\ell(n)}$.*

If, for any value of n, H is defined only for inputs $x \in \{0,1\}^{\ell'(n)}$, then the hash function is said to be fixed-length.

We consider only compression hash functions, i.e. $\ell'(n) > \ell(n)$.

Keyed Hash Functions

For any value of n, (KeyGen, H) determines a keyed function

$$H : \mathrm{KeySet}_n \times \mathrm{InSet}_n \to \mathrm{OutSet}_n$$

where

- KeySet_n contains all outputs of KeyGen on input n;
- InSet_n is $\{0,1\}^*$ or $\{0,1\}^{\ell'(n)}$;
- $\mathrm{OutSet}_n = \{0,1\}^{\ell(n)}$;
- $H(s,x) = H^s(x)$.

Security Guarantees - Collision Resistance

Let (KeyGen, H) be a keyed hash function.

Given a key $s \in \text{KeySet}_n$, it should be infeasible for any PPT adversary to find a collision, i.e. $x \neq x'$ s.t. $H^s(x) = H^s(x')$.

- Since the domain is larger than the range, collisions always exist, but it is required that they are hard to find.
- The key is not a secret.

Security Guarantees - Collision Resistance

Let (KeyGen, H) be a keyed hash function.

Collision-Finding Experiment $\mathrm{Hash}^{\mathrm{coll}}_{\mathcal{A},H}(n)$

Security Guarantees - Collision Resistance

Let (KeyGen, H) be a keyed hash function.

Collision-Finding Experiment $\text{Hash}^{\text{coll}}_{\mathcal{A},H}(n)$

Challenger Ch	Adversary $\mathcal{A}$
$s \leftarrow \text{KeyGen}(n)$	
	Receives s
	Outputs x, x'

$\mathcal{A}$ wins the game, i.e. $\text{Hash}^{\text{coll}}_{\mathcal{A},H}(n) = 1$, if $x, x' \in \text{InSet}_n$, $x \neq x'$ and $H^s(x) = H^s(x')$.

Security Guarantees - Collision Resistance

Let (KeyGen, H) be a keyed hash function.

Collision-Finding Experiment $\mathrm{Hash}^{\mathrm{coll}}_{\mathcal{A},H}(n)$

Challenger Ch	Adversary $\mathcal{A}$
$s \leftarrow \mathrm{KeyGen}(n)$	
	Receives s
	Outputs x, x'

$\mathcal{A}$ wins the game, i.e. $\mathrm{Hash}^{\mathrm{coll}}_{\mathcal{A},H}(n) = 1$, if $x, x' \in \mathrm{InSet}_n$, $x \neq x'$ and $H^s(x) = H^s(x')$.

Definition

A keyed hash function (KeyGen, H) *is collision resistant if, for every PPT adversary* $\mathcal{A}$*,* $\Pr(\mathrm{Hash}^{\mathrm{coll}}_{\mathcal{A},H}(n) = 1) \leq \mathrm{negl}(n)$.

Hash Functions in Practice

Hash functions used in practices are unkeyed:

$$H : \{0,1\}^* \to \{0,1\}^\ell$$

Hash Functions in Practice

Hash functions used in practices are unkeyed:

$$H : \{0,1\}^* \to \{0,1\}^\ell$$

What is the reason for using keyed functions?

Hash Functions in Practice

Hash functions used in practices are unkeyed:

$$H : \{0,1\}^* \to \{0,1\}^\ell$$

What is the reason for using keyed functions?

- Theoretically speaking, a colliding pair can be hardcoded and output by a polynomial-time algorithm.
- Keyed functions: impossible to hardcode a colliding pair for every value of n.

Hash Functions in Practice

Hash functions used in practices are unkeyed:

$$H : \{0,1\}^* \to \{0,1\}^\ell$$

What is the reason for using keyed functions?

- Theoretically speaking, a colliding pair can be hardcoded and output by a polynomial-time algorithm.
- Keyed functions: impossible to hardcode a colliding pair for every value of n.

Colliding pairs are unknown and computationally hard to find for hash functions used in practice.

Weaker Security Guarantees

Second-preimage (or target-collision) resistance: for any $s \in \mathrm{KeySet}_n$ and a uniform $x \in \mathrm{InSet}_n$, it is infeasible for any PPT adversary to find $x' \in \mathrm{InSet}_n$ s.t. $x \neq x'$ and $H^s(x) = H^s(x')$.

Preimage resistance (or one-wayness): given $s \in \mathrm{KeySet}_n$ and a uniform $y \in \mathrm{OutSet}_n$, it is infeasible for any PPT adversary to find $x \in \mathrm{InSet}_n$ s.t. $H^s(x) = y$.

Weaker Security Guarantees

Second-preimage (or target-collision) resistance: for any $s \in \text{KeySet}_n$ and a uniform $x \in \text{InSet}_n$, it is infeasible for any PPT adversary to find $x' \in \text{InSet}_n$ s.t. $x \neq x'$ and $H^s(x) = H^s(x')$.

Preimage resistance (or one-wayness): given $s \in \text{KeySet}_n$ and a uniform $y \in \text{OutSet}_n$, it is infeasible for any PPT adversary to find $x \in \text{InSet}_n$ s.t. $H^s(x) = y$.

collision resist. $\Rightarrow$ second-preimage resist. $\Rightarrow$ preimage resist.

Further Reading I

- Mihir Bellare and Phillip Rogaway.
  Random oracles are practical: A paradigm for designing efficient protocols.
  In Proceedings of the 1st ACM conference on Computer and communications security, pages 62–73. ACM, 1993.

- Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
  Keccak sponge function family main document.
  Submission to NIST (Round 2), 3:30, 2009.

- Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya.
  Merkle-Damgård revisited: How to construct a hash function.
  In Advances in Cryptology–CRYPTO 2005, pages 430–448. Springer, 2005.

Further Reading II

- Morris J Dworkin.
  SHA-3 standard: Permutation-based hash and extendable-output function.
  No. Federal Inf. Process. Stds.(NIST FIPS)-202, 2015.

- Pierre Karpman, Thomas Peyrin, and Marc Stevens.
  Practical free-start collision attacks on 76-step SHA-1.
  In Advances in Cryptology–CRYPTO 2015, pages 623–642. Springer, 2015.

- Neal Koblitz and Alfred J Menezes.
  The random oracle model: a twenty-year retrospective.
  Designs, Codes and Cryptography, pages 1–24, 2015.

Further Reading III

Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone.
Handbook of applied cryptography.
CRC press, 1996.

Marc Stevens.
New collision attacks on SHA-1 based on optimal joint local-collision analysis.
In Advances in Cryptology–EUROCRYPT 2013, pages 245–261. Springer, 2013.

Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov.
The first collision for full SHA-1.
In Annual International Cryptology Conference–CRYPTO 2017, pages 570–596. Springer, CHam, 2005.