Hash Functions and MACs

Cryptography

School of Engineering and Technology
CQUniversity Australia

Prepared by Steven Gordon on 15 Apr 2020,
hash.tex, r1851

Summary of Authentication Primitives

- Two types of general hash functions:
- Unkeyed hash function, $h = \mathrm{H}(M)$
  - Also simply called (cryptographic) hash function
  - Output hash value, h, also called *message digest*, *digital fingerprint*, or *imprint*
  - Primary function: MDC
- Keyed hash function, $h = \mathrm{H}(K, M)$
  - Output h often called *code*, *tag* or *MAC*
  - Primary function: MAC1

Contents

Hash Functions for Cryptography

- Hash function or algorithm $\mathrm{H}()$:
  - Input: variable-length block of data M
  - Output: fixed-length, small, hash value, h, where $h = \mathrm{H}(M)$
  - Another name for hash value is digest
  - Output hash values should be evenly distributed and appear random
- A secure, cryptographic hash function is practically impossible to:
  - Find the original input given the hash value
  - Find two inputs that produce the same hash value

Applications of Hash Functions

- Message authentication
- Digital signatures
- Storing passwords
- Signatures of data for malicious behaviour detection (e.g. virus, intrusion)
- Generating pseudorandom number

Design Approaches for Hash Functions

Based on Block Ciphers Well-known and studied block ciphers are used with a mode of operation to produce a hash function. Generally, less efficient than customised hash functions.

Based on Modular Arithmetic Similar motivation as to basing on block ciphers, but based on public key principles. Output length can be any value. Precautions are needed to prevent attacks that exploit mathematical structure.

Customised Hash Functions Functions designed for the specific purpose of hashing. Disadvantage is they haven't been studied as much as block ciphers, so harder to design secure functions.

Selected Cryptographic Hash Functions

Primitive	Output Length	Classification Legacy	Classification Future
SHA-2	256, 384, 512, 512/256	✓	✓
SHA-3	256, 384, 512	✓	✓
SHA-3	SHAKE128, SHAKE256	✓	✓
Whirlpool	512	✓	✓
BLAKE	256, 384, 512	✓	✓
RIPEMD-160	160	✓	✗
SHA-2	224, 512/224	✓	✗
SHA-3	224	✓	✗
MD5	128	✗	✗
RIPEMD-128	128	✗	✗
SHA-1	160	✗	✗

Credit: ECRYPT CSA Algorithms, Key Size and Protocols Report, 2018

Contents

Pre-image of a Hash Value (definition)

For hash value $h = \mathrm{H}(x)$, x is pre-image of h. As H is a many-to-one mapping, h has multiple pre-images. If H takes a b-bit input, and produces a n-bit hash value where $b > n$, then each hash value has 2^{b-n} pre-images.

Hash Collision (definition)

A collision occurs if $x \neq y$ and $\mathrm{H}(x) = \mathrm{H}(y)$. Collisions are undesirable in cryptographic hash functions.

Number of Collisions (exercise)

If H_1 takes fixed length 200-bit messages as input, and produces a 80-bit hash value as output, are collisions possible?

Requirements of Cryptographic Hash Functions

Variable input size: H can be applied to input block of any size

Fixed output size: H produces fixed length output

Efficiency: $\mathrm{H}(x)$ relatively easy to compute (practical implementations)

Pseudo-randomness: Output of H meets standard tests for pseudo-randomness

Properties: Satisfies one or more of the properties: Pre-image Resistant, Second Pre-image Resistant, Collision Resistant

Pre-image Resistant Property (definition)

For any given h, it is computationally infeasible to find y such that $\mathrm{H}(y) = h$. Also called the *one-way property*.

Second Pre-image Resistant Property (definition)

For any given x, it is computationally infeasible to find $y \neq x$ with $\mathrm{H}(y) = \mathrm{H}(x)$. Also called *weak collision resistant* property.

Collision Resistant Property (definition)

It is computationally infeasible to find any pair (x, y) such that $\mathrm{H}(x) = \mathrm{H}(y)$. Also called *strong collision resistant* property.

Required Hash Function Properties for Different Applications

	Preimage Resistant	**Second Preimage Resistant**	**Collision Resistant**
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

* Resistance required if attacker is able to mount a chosen message attack

Credit: Table 11.2 . In W. Stallings, "Cryptography and Network Security: Principles and Practice", 7th Edition, Pearson Education, 2017.

Brute Force Attacks on Properties

- Pre-image and Second Pre-image Attack
  - Find a y that gives specific h; try all possible values of y
  - With b-bit hash code, effort required proportional to 2^b
- Collision Resistant Attack
  - Find any two messages that have same hash values
  - Effort required is proportional to $2^{b/2}$
  - Due to birthday paradox, easier than pre-image attacks

Brute Force Attack on Hash Function (exercise)

Consider a hash function to be selected for use for digital signatures. Assume an attacker has compute capabilities to calculate 10^{12} hashes per second and is prepared to wait for approximately 10 days for a brute attack. Find the minimum hash value length that the hash function should support, such that a brute force is not possible.

Contents

Unkeyed and Keyed Hash Functions

- Hash functions have no secret key
  - Can be referred to as unkeyed hash function
  - Also called Modification Detection Code
- A variation is to allow a secret key as input, in addition to the message
  - $h = \mathrm{H}(K, M)$
  - Keyed hash function or Message Authentication Code (MAC)
- Hashes and MACs can be used for message authentication, but hashes also used for multiple other purposes
- MACs are more common for authentication messages

Design Approaches for MACs

Based on Block Ciphers CBC-MAC, OMAC, PMAC,

Customised MACs MAA, MD5-MAC, UMAC, Poly1305

Based on Hash Functions HMAC

Computation Resistance of MAC (definition)

Given one or more text-tag pairs, $[x_i, \text{MAC}(K, x_i)]$, computationally infeasible to compute any text-tag pair $[y, \text{MAC}(K, y)]$, for a new input $y \neq x_i$

Security of MACs

- Brute Force Attack on Key

  Attacker knows $[x_1, T_1]$ where $T_1 = MAC(K, x_1)$Key size of k bits: brute force on key, 2^kBut ... many tags match T_1For keys that produce tag T_1, try again with $[x_2, T_2]$Effort to find K is approximately 2^k

- Brute Force Attack on MAC value

  For x_m, find T_m without knowing KSimilar effort required as one-way/weak collision resistant property for hash functionsFor n bit MAC value length, effort is 2^n

- Effort to break MAC: $\min(2^k, 2^n)$