Hash Functions

Gaspare FERRARO

CyberSecNatLab

Matteo ROSSI

Politecnico di Torino

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it

License & Disclaimer

License Information

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Goal

- Provide the definition of hash function
- Show requirements of hash functions
- Learn how to build hash functions in practice
- Usage of hash functions

Prerequisites

- Lectures:
  - *CR_1.1 - Introduction to cryptography and classical ciphers*
  - *CR_1.3 - Block Ciphers*
  - *CR_1.4 - Stream Ciphers*

Outline

- Introduction and properties
- Collision resistance and birthday attack
- The Merkle-Damgård paradigm
- Standard hash functions
- Application of hash functions

Outline

- Introduction and properties
- Collision resistance and birthday attack
- The Merkle-Damgård paradigm
- Standard hash functions
- Application of hash functions

Introduction

- Two important goals of cryptography:
  - Message integrity: can the recipient be confident that the message has not been accidentally modified?
  - Message authentication: can the recipient be confident that the message originated from the sender?

Hash function

- A Hash function is a function that maps strings of arbitrary length to strings with fixed (short) length n

$$H: \{0,1\}^* \mapsto \{0,1\}^n$$

- The generated string is usually referred to as *digest* of the input string

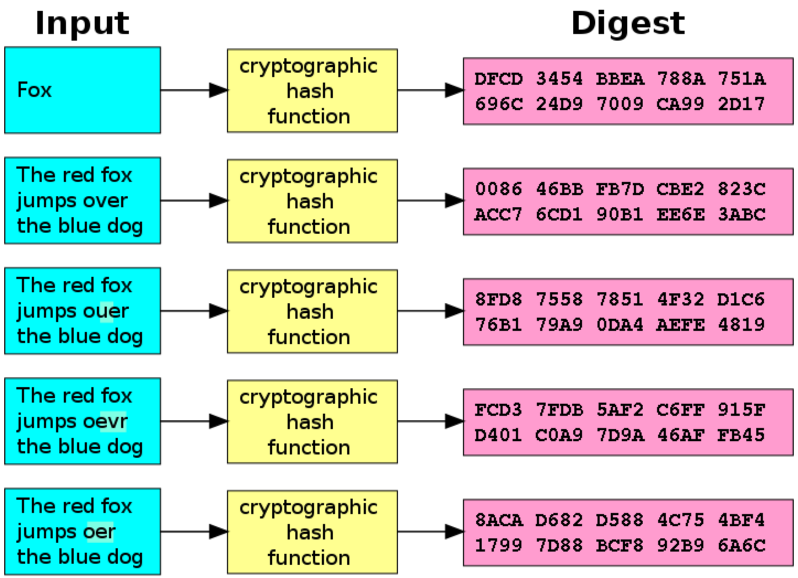
Hash functions - Examples

- The following functions are valid hash functions:
  - $H(s) = 1$
  - $H(s) = s[0]$
  - $H(s) = s^e (mod\, n)$
- The following are not:
  - $H(s) = s$
  - $H(s) = s^e$

Cryptographic hash function

- The interesting hash functions are the cryptographic hash functions, i.e. those members of a family of functions which are:
  - One-way function, i.e., they are practically infeasible to invert
  - Collision resistant, i.e., they:
    - avoid that two different messages generate a same digest
    - guarantee resistance to message forgery

Cryptographic hash function: Example



- A cryptographic hash function (specifically SHA-1) at work
- A small change in the input (in the word "over") drastically changes the digest
- This is the so-called avalanche effect

Outline

- Introduction and properties
- Collision resistance and birthday attack
- The Merkle-Damgård paradigm
- Standard hash functions
- Application of hash functions

Collision

- Let H be a hash function
- A collision for H occurs when a same digest is obtained from a pair of distinct messages m and m':

$$H(m) = H(m')$$

Collision resistance

- We say that H is *collision resistant* if there is no *explicit efficient* algorithm to find collisions for H
  - By "*efficient*" we mean that it should be feasible to execute the algorithm
  - By "*explicit*" we mean that is it possible to find different collisions by execute the algorithm several time

Collision resistance - remarks

- Every hash function has infinite inputs that provide a same digest
- This is because hash functions maps strings of arbitrary length to strings of fixed length
- We are interested in adopting hash functions for which it is "*difficult*" to find algorithms capable of easily (i.e., in a short space of time) identifying collisions for a given input message

Cryptographic hash function Resistances

- Preimage resistance (*one-way function*):
  - Given a digest h, it should be difficult to find a message m such that $h = H(m)$
- 2nd preimage resistance (*Weak collision resistance*):
  - Given a message m_1, it should be difficult to find a message $m_2 \neq m_1$ such that $H(m_1) = H(m_2)$
- Strong collision resistance:
  - It should be difficult find a pair of (m_1, m_2) such that $H(m_1) = H(m_2)$
- Weak collision resistance is related to a specific input, while strong collision resistance to two arbitrary inputs

Birthday attack

- In the sequel, we present a general attack to find collision in hash functions in $O\left(2^{n/2}\right)$ calls to H, where n is the size in bits of the digest
- This attack is called *Birthday attack*

Birthday attack – Algorithm

1. Choose $2^{n/2}$ distinct random messages, hash them and store the digests in a table
2. Look for a collision in that table
3. If there's no collision, repeat from 1

Theorem

- If one takes $1.2 \cdot 2^{n/2}$ samples, then the probability of finding a collision is $\geq 1/2$

Birthday attack – Analysis

- How well does this work?
  - Based on previous theorem we can easily see that the expected number of iterations of the algorithm is 2
- With a birthday attack, it is possible to find a collision of a hash function in $O(2^{n/2})$
- Therefore, the value of n of a hash function must be chosen in order to make this type of attack impracticable

Hash collisions and exploitations

- Ange Albertini and Marc Stevens provide a very useful resource about hash collisions and various exploitation techniques for different hash functions and file formats, available here:
  - https://github.com/corkami/collisions

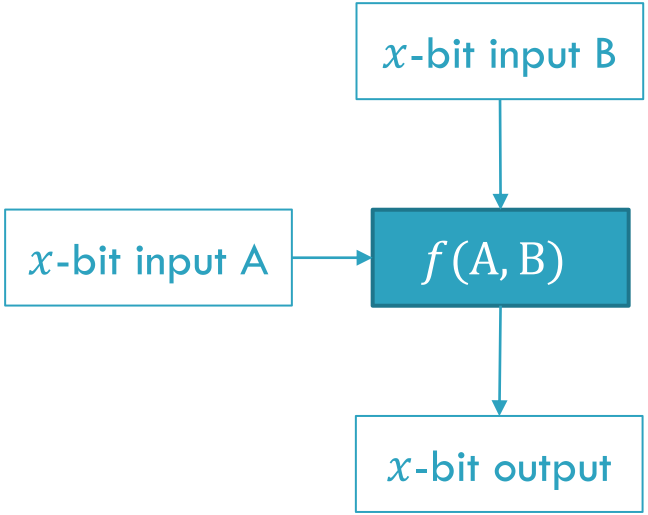
Outline

- Introduction and properties
- Collision resistance and birthday attack
- The Merkle-Damgård paradigm
- Standard hash functions
- Application of hash functions

The Merkle-Damgård Paradigm

- The Merkle-Damgård paradigm:
  - Is a standard construction for hash functions
  - Allows the construction of collision resistant hashes for arbitrary long inputs, exploiting collision resistant functions, called *compression functions*, for short (fixed-length) inputs

Compression function



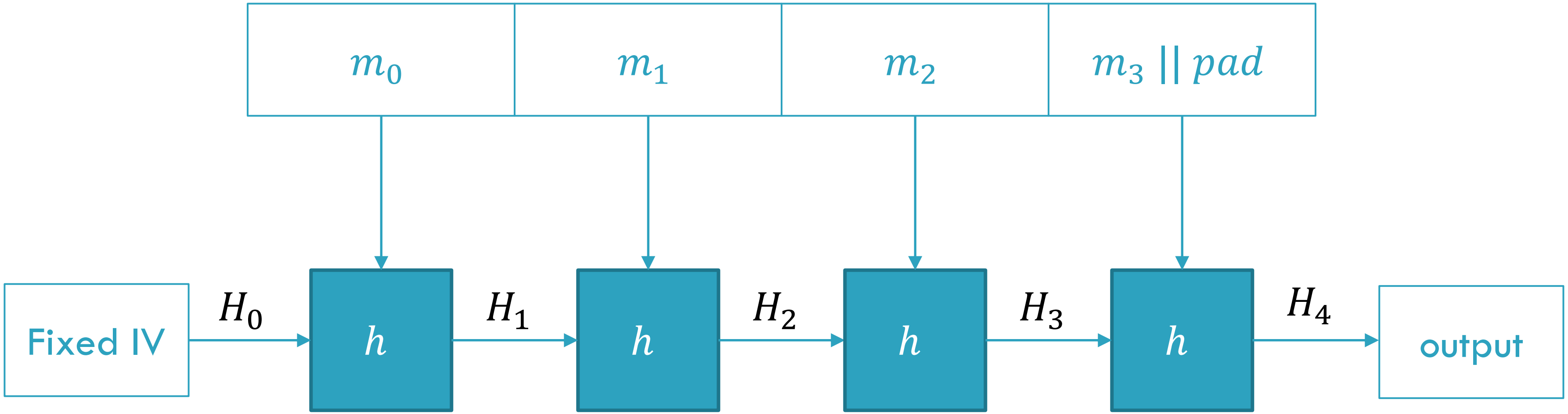
- A compression function is one-way function that transforms two fixed-length inputs into a fixed-length output

The Merkle-Damgård Paradigm

- In practice:
  - Pad, as described in the next slide, the message m to $(m||pad)$, until it reaches a length multiple of the size required by the compression function
  - Split m into k blocks $m_1, m_2, \dots\ m_k$
  - Implement the Merkle-Damgård scheme presented in the next slide

The Merkle-Damgård Paradigm

- Merkle-Damgård scheme with a message of $k = 4$ blocks and a compression function h, initially fed by a fixed initialization vector IV



Padding

- Padding in hashes is done differently than in block ciphers:
  - We append a "1" (in binary) at the end of the message
  - We append "0" in all bits except the last 64
  - We encode the length of the message in the last 64 bits
  - If less than 64 bits are needed to complete the block, an additional block is added

Theorem on Collision resistance

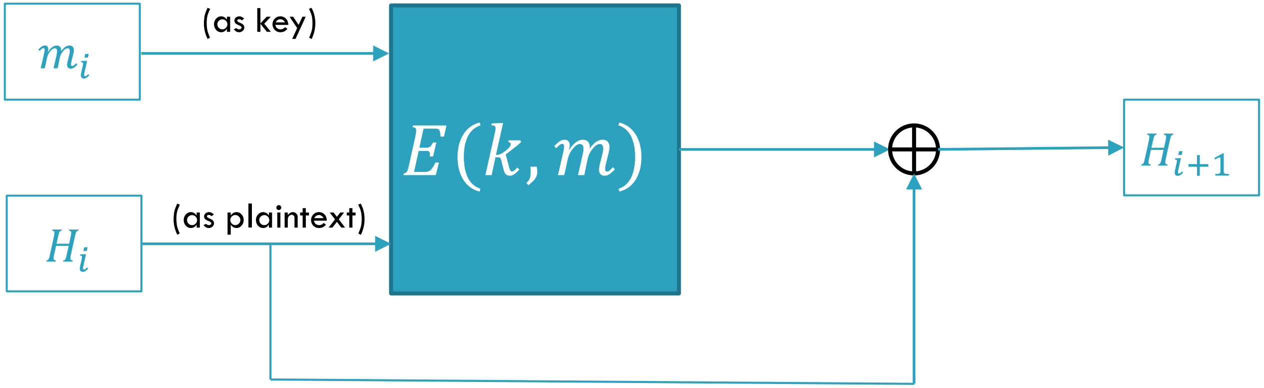
- If the compression function h is collision resistance, then the full hash function is collision resistant too

- We only need a way to build compression functions and we are done

The Davies-Meyer construction

- The most popular way to build compression function is called the "Davies-Meyer construction"
  - Start from a *secure* block cipher E (CR_1.3 - Block Ciphers)
  - The current block m_i is the key
  - The previous result (or the IV) H_i is the message
  - The output is XORed ($\bigoplus$) with the previous result
- The Davies-Meyer construction is presented in the next slide

The Davies-Meyer construction



Remarks

- Other variants of the construction are possible
  - Es. Miyaguchi and Preneel proposed 12 different and secure variants
- The most widely used hash functions (MD5, SHA-1, SHA2) use the Merkle-Damgård construction
- Davies-Meyer is used in most of the hashes with Merkle-Damgård structure
- "Natural" variants of Davies-Meyer are insecure:
  - Omitting the XOR
  - XOR-ing with m_i instead of H_i

Outline

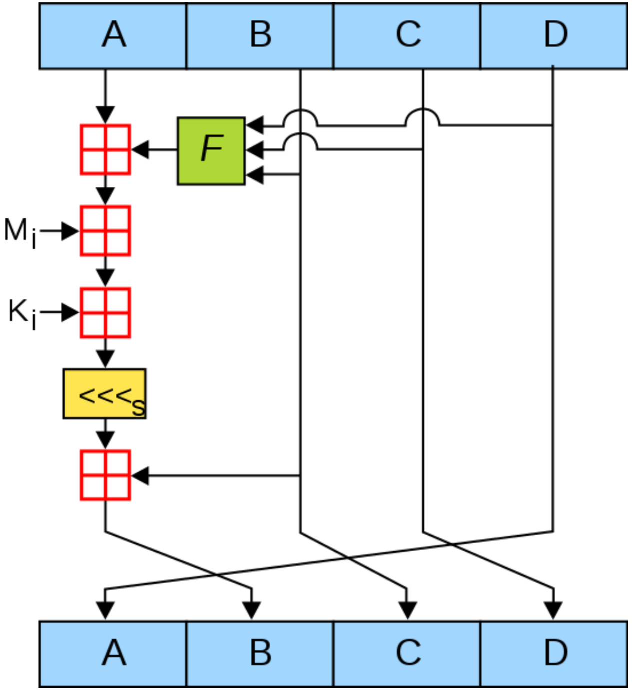
- Introduction and properties
- Collision resistance and birthday attack
- The Merkle-Damgård paradigm
- Standard hash functions
- Application of hash functions

Standard hash functions

- Various cryptographic (and not) hash functions exist in literature
- They differ in construction, length, round and many other parameters
- A comparison can be found here:
  - https://en.wikipedia.org/wiki/Comparison_of_cryptographic_hash_functions
- In the next slides some cryptographic hash functions, based on Merkle-Damgård construction, are presented

MD5

- The MD5 is a message-digest algorithm, based on Merkle–Damgård construction, designed in 1991 by Ronald Rivest
- It is a widely used hash function producing a 128-bit digest
- Initially designed to be used as a cryptographic hash function, MD5 has later been found present extensive vulnerabilities
- It is still used as a checksum for message integrity (e.g., *md5sum* command in Linux)



Secure Hash Algorithm - SHA

- As for DES and AES, for symmetric cryptography, there are also *standards* for cryptographic hash functions
- The standard for cryptographic functions is called Secure Hash Algorithm (SHA), a suite of 3 cryptographic functions:
  - *SHA-1*: a 160-bit hash function, no longer approved after 2010
  - *SHA-2*: a set of 6 hash functions, characterized by different digest lengths (224, 256, 384 or 512 bits)
  - *SHA-3:* a new hash function chosen in 2012 which support same SHA-2 hash lengths but with different internal structures

SHA-1

- Is cryptographic hash function that returns a 160-bit Message Digest
- As MD5, is based on Merkle–Damgård construction
- Widely used in:
  - Standard protocols, likes TLS, SSL, PGP, and SSH
  - Versioning systems, like Git and Mercurial

SHA-1 Weaknesses

- SHA-1 dates back to 1995 and is known to be vulnerable since 2005
- Since 2010, many organizations have recommended its replacement
- On 2017 Google announced the SHAttered attack (https://shattered.io/), in which they generated two different PDF files with the same SHA-1
- All major web browsers ceased accepting SHA-1 based SSL certificates in 2017
- Starting in 2020, it is recommended to always use SHA-x variants

SHA-2

- SHA-2 is a set of cryptographic hash functions designed by National Security Agency in 2001 and published by the NIST
- The SHA-2 family consists of six hash functions with digest which are 224, 256, 384 or 512 bits:
  - SHA-256 and SHA-512 use different shifting and additive constants, but their structures are virtually identical, differing only in the number of rounds
  - SHA-224 and SHA-384 are truncated versions of SHA-256 and SHA-512, calculated with different initial values
  - SHA-512/224 and SHA-512/256 are also truncated versions of SHA-512, but the initial values are generated differently

Outline

- Introduction and properties
- Collision resistance and birthday attack
- The Merkle-Damgård paradigm
- Standard hash functions
- Application of hash functions

Application of hash functions

- Hash functions can be used in many applications:
  - Message integrity
  - Message authentication (see CR_3.2 - MAC)
  - Digital signatures (see CR_3.3 - Digital Signatures)
  - Password verification by hash comparison
  - Pseudo-random number generator (PRNG)

Message integrity

- The main use of a hash function is to ensure the integrity of a message M:
  - The message M is sent along with its digest $\mathrm{H}(M)$
  - If, during the communication, the message M is altered to become M', the recipient, calculating the digest on M', will most likely obtain a different digest from the one sent together with M

Password verification by hash comparison

- In some (hopefully, all) applications, passwords are not stored in clear, but only their digest
- Anyone able to access the application would not be able to recover passwords from the stored digest
- When a user types a password, the application evaluate its digest and compares it to the corresponding digest stored in the application

Pseudo-random number generator

- Hashing is also used to generate random numbers as:

$$n = hash(seed)$$

- The result is a pseudo-random number as a cryptographic hash function takes a variable amount of data as input and outputs a fixed-length string
- If the seed is chosen with care and used with a cryptographic hash, due to the avalanche effect, the digest will contain evenly distributed bits

Hash Functions

Gaspare FERRARO
CyberSecNatLab
Matteo ROSSI
Politecnico di Torino

https://cybersecnatlab.it