Hash functions

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Emai: gianluca.dini@unipi.it
Version: 07/04/2025

1

An example

The input size is finite but arbitrary

```
Nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura
che' la diritta via era smarrita.

Ahi quanto a dir qual era e` cosa dura
esta selva selvaggia e aspra e forte
che nel pensier rinova la paura!
```

MD5

```
0xd94f329333386d5abef6475313755e94
```

128 bit

The hash size is fixed, generally smaller than the message size

2

Informal properties

- Applicable to messages of any size
- Output of fixed length (digest, hash, fingerprint, tag,…)
- No key (!)
- "Easy" to compute
- "Difficult" to invert
- "Unique": the hash of a message can be used to "uniquely" represent the message

3

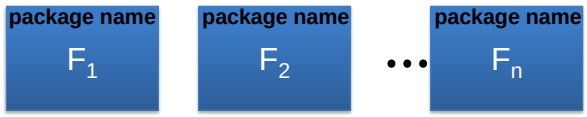
Informal properties

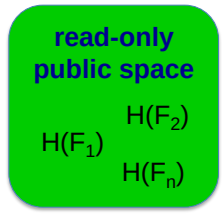
- The fingerprint must be *highly* sensitive to *all* input bits
- if we make minor modifications to the input, the output should look like very different
- Example
  - Input «I am not a crook»
    - Hash (MD5): 6d17fcd4ae0e82fa4409f4ea6f4106a6
  - Input «I am not a cook»
    - Hash (MD5): 9ebe3d42d5c01fc59fe3daacbf42f515
- https://www.fileformat.info/tool/hash.htm

4

Example: protecting files

- **Software packages**





- When user downloads package, can verify that contents are valid
  - An attacker cannot modify a package without detection
- No key needed (public verifiability), but requires read-only space

5

Example: protecting files

6

Properties and collisions

- A hash function H: $\{0,1\}^* \rightarrow \{0,1\}^n$
- **Properties**
  - **Compression**: H maps an input *x* of arbitrary finite length into an output H(x) of fixed length *n*
  - **Ease to compute:** given *x*, H(x) must be "easy" to compute
  - **Many-to-one:** a hash function is many-to-one and thus implies collisions (pigeonhole principle)
- (**Def**) A **collision** for H is a pair x_1, x_2 s.t. $H(x_1) = H(x_2)$ and $x_1 \neq x_2$

Security properties [1/2]

- **Preimage resistance (one-wayness)**
  - For essentially all pre-specified outputs, it is *computationally infeasible* to find any input which hashes to that output
    - i.e., given an output y, to find x such that y = h(x) for which x is not known
- **2nd-preimage resistance (weak collision resistance)**
  - it is computationally infeasible to find any second input which has the same output as any specified input
    - i.e., given *x*, to find x' ≠ x such that h(x) = h(x')

Security properties [2/2]

- **Collision resistance (strong collision resistance)**
  - it is computationally infeasible to find any two distinct inputs which hash to the same output,
    - i.e., find x, x' such that $h(x) = h(x')$

Classification

- **One-way hash function (OWHF)**
  - Provides preimage resistance, 2-nd preimage resistance
  - OWHF is also called weak one-way hash function
- **Collision resistant hash function (CRHF)**
  - Provides 2-nd preimage resistance, collision resistance
  - CRHF is also called strong one-way hash function

Relationship between security properties

- **FACT 1:** Collision resistance implies 2nd preimage resistance
- **FACT 2:** Collision resistance does not imply preimage resistance
  - However, in practice, CRHF almost always has the additional property of preimage resistance

11

Attacking Hash Functions

- An attack is successful if it produces a collision (**forgery**)
- Types of forgery
  - **Selective forgery**: the adversary has complete, or partial, control over x
  - **Existential forgery**: the adversary has no control over x

12

Black box attacks →

- Consider H(·) as a black box
- Only consider the output bit length n
- Assume H(·) approximates a random variable
  - Each output is equally likely for a random input (so weak collisions exist for all output values)

13

Specific Black box Attacks ↓

- **Guessing attack**
  - find a 2nd pre-image
  - Running time: $O(2^n)$ hash ops
- **Birthday attack**
  - find a collision
  - Running time: $O(2^{n/2})$ hash ops
- These attacks constitute a **security upper bound**
  - More efficient analytical attacks may exist (e.g., against MD5 and SHA-1)

14

Guessing attack

- Objective: to find a 2nd pre-image
  - Given x_1, find $x_2 \neq x_1$ s.t. $H(x_1) = H(x_2)$
- The attack

```
int GuessingAttack(x1) {
    do
        x2 ← random(); // guessing
    while ( H(x1) != H(x2) )
    return x2;
}
```


15

Guessing attack

- Running time
  - Every step requires
    - 1 random number generation: efficient!
    - 1 hash function computation: efficient!
  - Constant and negligible data/storage complexity
  - Running time in the order of 2^n operations
    - At each step $Pr[H(x1) = H(x2)] = 1/2^n$

16

Birthday attack →

- Start with
  - x_1 = «Transfer $10 into Oscar's account»
  - x_2 = «Transfer $10.000 into Oscar's account»
- The attack
  - **do**
    - Alter x_1 and x_2 at non-visible locations so that semantics is unchanged (e.g., insert spaces, tabs, return,…)
  - **while** ($H(x_1)$!= $H(x_2)$)

17

Birthday attack →

- The birthday attack algorithm
  1. Choose $N = 2^{n/2}$ random input messages $x_1, x_2, \ldots, x_N$ (distinct w.h.p.)
  2. For i := 1 to N compute $t_i = H(x_i)$
  3. Look for a collision ($t_i = t_j$), $i \neq j$. If not found, go to step 1.
- Attack complexity
  - Running Time: $2^{n/2}$
  - Space: $2^{n/2}$
  - Probability of collision is 50%

18

Birthday paradox: intuition →

- Problem #1.
  - In a room of t = 23 people, what is the probability that at least a person is born on 25 December?
    - Answer: 0.063
- Problem #2.
  - In a room of t = 23 people, what is the probability that at least 2 people have the same birthdate?
    - Answer: 0.507

Birthday attack →

- Apply the birthday paradox to hash function
  - We have: 2^n elements and t inputs (x_1, x_2, …, x_t)
  - π = Pr[no collision] = $\left(1-\frac{1}{2^n}\right)\left(1-\frac{2}{2^n}\right)\cdots\left(1-\frac{t-1}{2^n}\right) =$

    $\prod_{i=1}^{t-1}\left(1-\frac{i}{2^n}\right) \approx \prod_{i=1}^{t-1} e^{-\frac{i}{2^n}} = e^{-\frac{1+2+\cdots+t-1}{2^n}} \approx e^{-\frac{t(t-1)}{2^{n+1}}} \cong$

    $e^{-\frac{t^2}{2^{n+1}}}$

Birthday paradox: intuition →

- Problem #1. Violating 2nd pre-image property is comparable to:
  - In a room of t = 23 people, what is the probability that at least a person is born on 25 December?
    - Answer: 0.063

SPECIFIC HASH

- Problem #2. Finding a collision is comparable to:
  - In a room of t = 23 people, what is the probability that at least 2 people have the same birthdate?
    - Answer: 0.507

19

Birthday attack →

- Apply the birthday paradox to hash function
  - We have: 2^n elements and t inputs ($x_1, x_2, \ldots, x_t$)

possible hash outputs

  - π = Pr[no collision] = $\left(1-\frac{1}{2^n}\right)\left(1-\frac{2}{2^n}\right)\cdots\left(1-\frac{t-1}{2^n}\right) =$

$$\prod_{i=1}^{t-1}\left(1-\frac{i}{2^n}\right) \approx \prod_{i=1}^{t-1} e^{-\frac{i}{2^n}} = e^{-\frac{1+2+\cdots+t-1}{2^n}} \approx e^{-\frac{t(t-1)}{2^{n+1}}} \cong e^{-\frac{t^2}{2^{n+1}}}$$

when $\frac{i}{2^n}$ small, this holds

$1+\ldots+(t-1) = \frac{t(t-1)}{2}$

20

Birthday attack

- Probability of collision $\lambda = 1 - \pi$
- Solve in t, ➔ $t \approx 2^{(n+1)/2} \sqrt{\ln\left(\frac{1}{1-\lambda}\right)}$
- For $\lambda = 0.5$, $t \approx 1.2 \times 2^{n/2}$

$\lambda = 1 - \pi = 1 - e^{-t^2/2^{n+1}}$

$t \cong 2^{\frac{(n+1)}{2}} \sqrt{\ln\left(\frac{1}{1-\lambda}\right)}$

• Solve in t in order to have a certain probability of collision.

So under $t = 2^{n/2}$ assumption, probability of collision is in the order of 0.5.

Birthday attack

- In practice,
  - The number of messages we need to hash to find a collision is in the order of the square root of the number of possible output values, i.e., $\sqrt{2^n} = 2^{n/2}$
- Example
  - n = 80 bit, $\lambda = 0.5$ ➔ $t \approx 2^{40.2}$ (doable with current laptops)
  - The probability of collision λ does not influence the attack complexity very much
- Rule of thumb: sizeof(digest) = 2 × sizeof(key)
  - Key is a block cipher key

It's just a matter of computation.

Birthday attack tell me that the security level is half of the one of the key.

For Symm.-Cr. sec level is # bits key. Here, double.

We implement hash functions with block cyphers that usually are cyphers.

Hash functions

HOW TO BUILD HASH FUNCTIONS

23

Hash functions | How to build hash functions

THE MERKLE-DAMGÅRD SCHEME

24

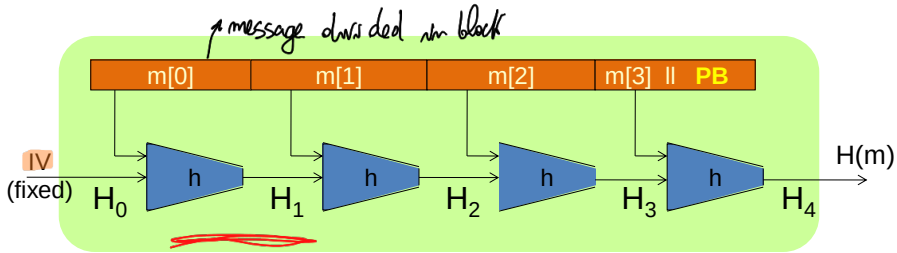
The Merkle-Damgård iterated construction

- Intuition. Given a CRHF for short messages, construct a CRHF for long messages

26

The Merkle-Damgård iterated construction

1. This is sequential, not parallelizable



- **Compression function** h: T × X ⟶ T
  - H_i - chaining variables
- Padding block PB: 1000… || msg len (#blocks)
  - msg len (on 64 bits) complicates adversary's task
  - If no space for PB add another block

27

- If a block has not the required size, it is padded.

All the up to now hash functions have been designed this way.

H takes message and produces the hash. If h is collision resistant, so is H. h is the compression function that takes smaller messages of fixed size. Designing a compression function for small and fixed length messages is easier.

The Merkle-Damgård iterated construction

- THEOREM. if compression function h is collision resistant (and message length is part of the input) then so is H.
  - Proof (by contradiction)
    - Collision on H ➔ collision on h. Q.E.D.
- Comment
  - To construct a CRHF, it *suffices* to construct a collision resistant compression function
  - The condition h is collision resistant is sufficient (not necessary)

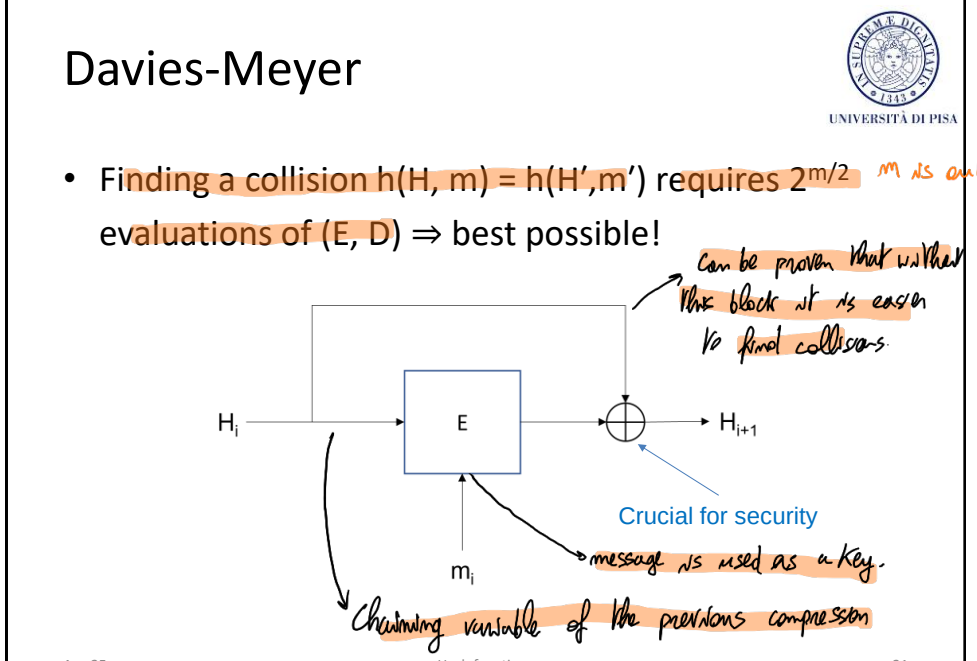
↳ I can build an hash f. according to the scheme although h is not.

How to build h? 3 most important ways: each one uses a block cypher:

Compression function

- Use block cipher (we use) chaining techniques
  - Matyas-Meyer-Oseas (most used one)
  - Davies-Meyer
  - Miyaguchi-Preneel

Davies-Meyer



- Finding a collision h(H, m) = h(H',m') requires $2^{m/2}$ evaluations of (E, D) ⇒ best possible!

31

Exercise

- Problem
  - If we remove the xor, the compression function is not collision resistant anymore.
  - Proof (by contradiction)
    - Remove the xor ➔ h(H, m) = E(m, H)
    - To construct a collision (H, m) and (H', m') is easy
      - Choose a random triple (H, m, m')
      - Determine H' such that E(m, H) = E(m', H') ➔ H' = D(m', E(m, H)

Q.E.D.

32

Comments of Davies-Meyer

- The SHA family uses Davies-Meyer and Damgard strup
- Davies-Meyer uses a custom cipher (No AES. Ad hoc designed)
  - Cipher key (m_i) changes at each round* while off-the-shelf ciphers are optimized to encrypt large messages with a fixed key. AES was optimized not for this
  - Block size of off-the-shelf ciphers would be too small (e.g., 128-bit). (1)
  - Off-the-shelf block ciphers have a too small key (e.g., 128-bit). This implies a small $| m_i |$ and thus too many rounds.
    - Key size of a DM Cipher is typically 512-, 1024-bit

*At every step.

(1) Output of hash f. would be in size output of block cypher. For birthday attack to break it you would have 2^{64} complexity. Not much. Hash f. up to now have been built according to Merkle-Damgard (advantage of collision resistancy). Davies-Meyer allow us to work with block cyphers for hash.

Merkle-Damgård collision resistant hash functions

Name	year	digest size	message block size	Speed[2] MB/sec	best known attack time
SHA0	1993	160	512		2^{39}
SHA1	1995	160	512	153	2^{63}
SHA224	2004	224	512		
SHA256	2002	256	512	111	
SHA384	2002	384	1024		
SHA512	2002	512	1024	99	
MD4	1990	128	512		2^{1}
MD5	1992	128	512	255	2^{16}
Whirlpool	2000	512	512	57	

MD5

- Developed in 1991
- 128-bit outuput lenght
- Collisions found in 2004, should be non longer used
  - Collision attack: $O(2^{24.1})$
  - Chosen-prefix collision attack: $O(2^{39})$

37

SHA-1

- Designed by NSA and standardised by NIST in 1995
- 160 bits output length
- Collision on SHA-1 in 2017, now deprecated Not only, but also able to forge PDF docs.
  - CWI – Google team
  - Forged PDF documents
  - Running time
    - Over 9+ quintillion SHA1 computations that took 6,500 years of CPU computation and 100 years of GPU computations however 10^5 times faster than black box attack
      - https://www.cwi.nl/news/2017/cwi-and-google-announce-first-collision-for-industry-security-standard-sha-1

38

Other hash functions

- SHA-2 (NIST 2002) ← standard in the 3 versions
  - 256-bit, 384-bit or 512-bit output lenght
  - No known significant weaknesses but its structure is similar to SHA-1 and MD5
- SHA-3/Keccak
  - Result from public competition from 2008-2012
  - Very different design than SHA family Doesn't use Merkle Dangland
    - Requirement from NIST to defend from possible weakness in SHA family
  - Support 224, 256, 384, and 512-bit output lenght

39

There exist a number of forensic tools that use MD5 and SHA1. They take hash of the HDD (as a sequence of bytes) and store hash, typically computed by MD5 and SHA1. Hash is used to prove integrity of the proof. Why still MD5 and SHA1?
Hash f. are deprecated because you can violate collision resistance, but you are interested in not breaking 2nd preimage resistance. You have x and you should define x' such that $H(x) = H(x')$. MD5 and SHA1 are not secure with respect to the 3rd property.

Hash functions | How to build hash functions

THE SPONGE CONSTRUCTION

40

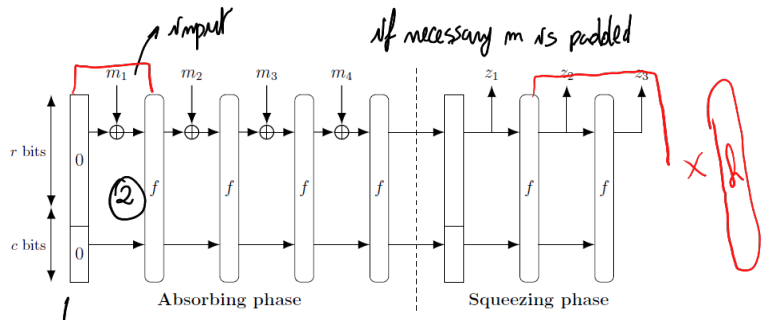
General issues

- The sponge construction is based on a permutation rather than a compression function
  - f: $\{0, 1\}^n \rightarrow \{0, 1\}^n$ [no key]

New construction in addition to Merkle-Damgard

41

The sponge scheme



initial value: all bits zero

r = rate
c = capacity.

r+c = n size of chain variable.

42

Scheme works in absorbing phase: message is mixed with chaining variable, and squeezing phase in which you get out the output.
If r is larger, computation is faster because blocks are larger (less blocks to use). C is relevant for security: larger c = larger security.

C is hidden from adversary.

How the sponge works

- n = r + c with r: rate and c: capacity
  - Larger r → faster evaluation
  - Larger c → better security
- The sponge works in two phases
  - **Absorbing phase**, where the message blocks get «mixed in» to a chaining variable
  - **Squeezing phase**, where the output is «pulled out» of the chaining variable.

Comments

- **Pros**. The sponge is designed to be used flexibly and securely in a variety of applications where collision resistance is not the main property we need.
  - E.G. Transform H into a PRF

↳ can also be used to transform hash in PRG.

- **Cons**. It is not known how to reduce the collision resistance of the sponge to a concrete security property of *f*. The only known analysis of the sponge is in the ideal permutation model, where we (heuristically) model f as a truly random permutation.

Properties can only be proven in case of ideal random permutation.

The NIST SHA3 family

- Permutation Keccak: $\{0,1\}^{1600} \rightarrow \{0,1\}^{1600}$
- **Def.** Denote by Keccak[c], he sponge derived from Keccak with capacity c
- Keccak[c](m, v) where *m* is an arbitrary size message and *v* is the output lenght

45

The NIST SHA3 family

- Fixed-lenght output hash function
  - SHA3-224(m) = Keccak[448](m ||01, 224)
  - SHA3-256(m) = Keccak[512](m ||01, 256)
  - SHA3-384(m) = Keccak[768](m ||01, 384)
  - SHA3-512(m) = Keccak[1024](m ||01, 512)
- Variable-lenght output hash function
  - SKAKE128(m, v) = Keccak[256](m ||1111, v)
  - SKAKE256(m, v) = Keccak[512](m ||1111, v)
- Note: m is padded as necessary

46

Hash functions

USES OF HASH FUNCTIONS

47

Uses of hash functions

- Digital signatures
  - Requires strong collision resistance
- Password storage
  - Requires weak collision resistance
- Authentication of origin
  - Requires weak collision resistance
- Identification (one-time password)
  - Requires weak collision resistance and one-wayness

48

Hash Functions

AUTHENTICATION OF ORIGIN

49

Integrity vs authentication

- Message integrity
  - The property whereby data has not been altered in an unauthorized manner since the time it was created, transmitted, or stored by an authorized source
- Message origin authentication
  - A type of authentication whereby a party is corroborated as the (original) source of specified data created at some time in the past
- Data origin authentication => data integrity

50

Use of hash functions for authentication

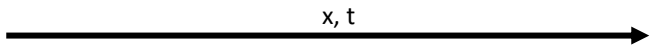
- The purpose of a hash functions, *in conjunction with other mechanisms* (authentic channel, encryption, digital signature), is to provide message integrity and authentication

Cannot be used alone for authenticity.

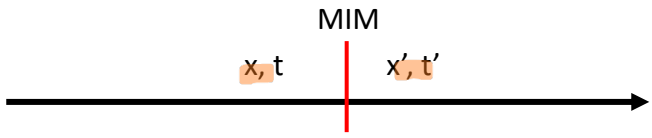
51

Authentic channel →

- Alice — Bob
  - Let t = H(x)



- MIM attack



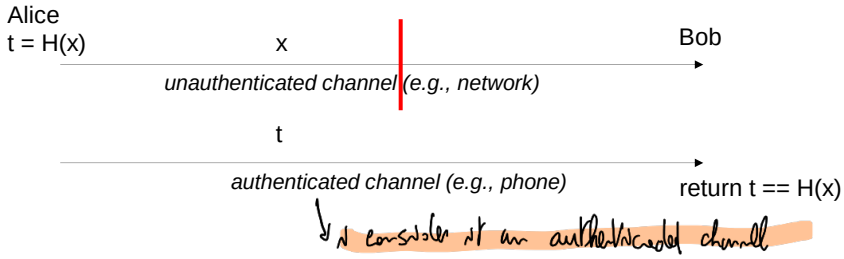
Man in the middle intercepts message and changes hash

52

Authentic channel

UNIVERSITÀ DI PISA

- Alice
  - Computes t = H(x)
  - Sends x to Bob through the network
  - Reads t to Bob over the phone
    - An additional channel considered authenticated by assumption



Apr-25 Hash functions 53

53

Hash functions with block ciphers

UNIVERSITÀ DI PISA

- $E_k(x||H(x))$ recommended
  - Confidentiality and integrity
  - As secure as E
  - H has weaker properties than digital signatures
- $x, E_k(H(x))$ not recommended
  - Prove that sender has seen H(x)
  - H must be collision resistant
  - Key k must be used only for this integrity function
- $E_k(x), H(x)$ not recommended
  - H(x) can be used to check guesses on x
  - H must be collision resistant

Apr-25 Hash functions 54

54