Lecture 4. Hash functions

Swiss Army knife of cryptography

Bruno Grenet

https://membres-ljk.imag.fr/Bruno.Grenet/IntroCrypto.html

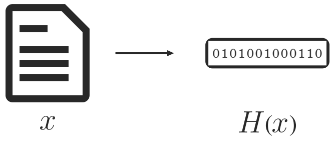
Introduction to cryptology
Université Grenoble Alpes – IM²AG
M1 INFO, MOSIG & AM

What are hash functions?

Definition

A(n unkeyed) hash function is a mapping $H : \mathcal{M} \to \mathcal{H}$, with

- $\mathcal{M} = \{0,1\}^{<N}$: the *message space* typically $N \geq 2^{64}$
- $\mathcal{H} = \{0,1\}^{n}$ with $N \gg n$: the *digests* $n \in \{\xcancel{128}, \xcancel{160}, \cancel{224}, 256, 384, 512\}$



Variants

- *extendable-output function* (XOF) $\to \mathcal{H} = \{0,1\}^{<n}$
- *keyed* hash function $H : \mathcal{K} \times \mathcal{M} \to \mathcal{H}$ family of hash functions

A hash function is simply a function: when is it *good*?

Usefulness of hash functions

Hash functions are an essential tool underlying most of (modern) cryptography!

- *Hash-and-sign* — RSA signatures, (EC)DSA, …
- Message authentication codes — HMAC, … $\to$ next lecture!
- Password hashing (with a grain of salt)
- Hash-based signatures
- Commitment
- Key derivation
- As one-way functions or *random oracle*
- …

Efficiency

- A few dozen cycles per byte
- Small memory
- …

Security of hash functions

Goals

- We would like $H(x)$ to
  - uniquely identify x
  - be much smaller than x
  - reveal no information on x
- Impossible: there exist *collisions* $H(x) = H(y)$ while $x \neq y$

Security notions

- First preimage resistance: given t, *hard* to find m such that $H(m) = t$
- Second preimage resistance: given m, *hard* to find $m' \neq m$ such that $H(m') = H(m)$
- Collision resistance: *hard* to find $m \neq m'$ such that $H(m) = H(m')$

Remarks

- No definition of *hard* H is *fixed*!
- Collision resistance $\Rightarrow$ 2nd preimage resistance
- 2nd preimage is *in some sense* stronger than 1st preimage resistance

The ideal world: random oracles

Definition

A random oracle is (equivalently)

- a function H sampled uniformly at random amongst the functions from $\mathcal{M}$ to $\mathcal{H}$
- a function $H : \mathcal{M} \to \mathcal{H}$ such that $\forall x \in \mathcal{M}, H(x) \twoheadleftarrow \mathcal{H}$

- As random as possible: each value $H(x)$ is sampled uniformly and independently
- Used in proof as the *random oracle model* — eq. to ideal cipher model
- Irrealistic but good hash functions are *approximations* — whatever this means

Generic attacks

- 1st preimage: $O(2^n)$ — exhaustive search
- 2nd preimage: $O(2^n)$ — *idem*
- Collision: $O(2^{n/2})$ — *"birthday attack"*

$\rightarrow$ A hash function is *good* if the generic attack is (almost) the best one

On the birthday attack

Reminder

- If $h_1, \dots, h_q \twoheadleftarrow \mathcal{H}$, $\Pr[\exists i \neq j, h_i = h_j] \geq \frac{q(q-1)}{4 \cdot 2^n}$ $\qquad q \simeq 2^{n/2} \Rightarrow$ collision prob. $\simeq \frac{1}{4}$
- Sample $\Omega(2^{n/2})$ values x_i: with good probability, $\exists\, x_i \neq x_j$ s.t. $H(x_i) = H(x_j)$

Space complexity

- To find a collision, need to store $\Omega(2^{n/2})$ values
- Floyd's *tortoise and hare* algorithm:
  1. $x_0 \twoheadleftarrow \mathcal{M}$
  2. do $(x_i, x_{2i}) \leftarrow (H(x_{i-1}), H(H(x_{2(i-1)})))$ until $x_i = x_{2i}$

  $\rightarrow$ Only two values to store, same time complexity

Useful collisions

Goal: Find two messages m_0 and m_1 of opposite meanings s.t $H(m_0) = H(m_1)$

- "I owe 1000€ to Bruno" and "Bruno owes me 1000€"

Method: Produce many variants of m_0 and m_1 until a collision is found

- "I have a 1000€ debt to Bruno", "Bruno is 1000€ in debt to me", …
- Variant of birthday bound: find a collision between two lists

Contents

Compression functions

Definition

A compression function is a mapping $f : \{0,1\}^n \times \{0,1\}^w \to \{0,1\}^n$

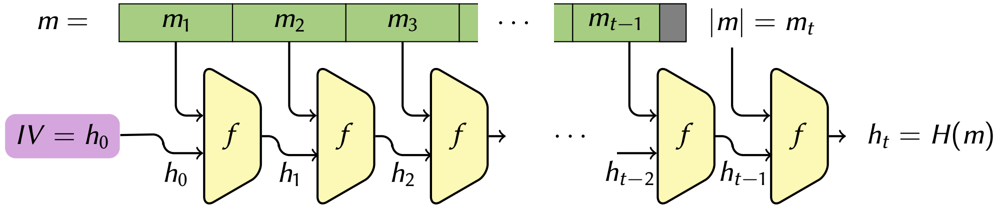
- Family of functions from $\{0,1\}^n$ to itself
- Compare to hash functions: fixed-length input
- Compare to block ciphers: not invertible

Goal

Assuming a *good* f is given, how to construct a *good* hash function?

- Fixed-size $\to$ Variable-size *domain extension*
- Compare to bock cipher modes of operation

Merkle-Damgård construction (1989)



- IV: **fixed** initial value in $\{0,1\}^n$ — part of H's specification
- $f : \{0,1\}^n \times \{0,1\}^w \to \{0,1\}^n$
- $\mathrm{pad}(m) = m \| 0 \cdots 0 \| \langle \text{length of } m \rangle \rightsquigarrow |\mathrm{pad}(m)| = t \times w$
- $H(m) = f(\cdots f(f(IV, m_1), m_2) \ldots, m_t)$

Efficiency

- t sequential calls to $f \to$ OK

Merkle-Damgård construction: security

Warm-up: first preimage resistance

If f is 1$^{\text{st}}$ preimage resistant, then H is 1$^{\text{st}}$ preimage resistant too

Proof by contrapositive.

Merkle-Damgård construction: security

Warm-up: first preimage resistance

If f is 1st preimage resistant, then H is 1st preimage resistant too

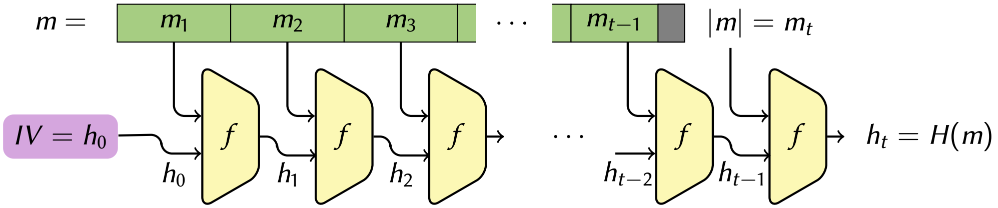
Collision resistance

If f is collision resistant, then H is collision resistant too

Proof by contrapositive.

Merkle-Damgård construction: 2nd preimage vulnerability

Idea of an attack by Kelsey & Schneier (2005)

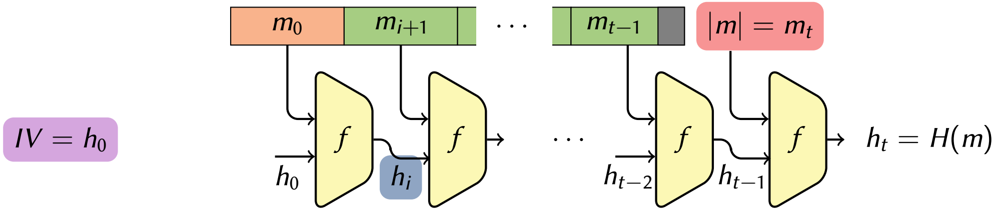


Goal: Given m, find $m' \neq m$ s.t. $H(m') = H(m)$

- Find m_0 such that $f(h_0, m_0) = h_i$ for *any* h_i $\simeq 2^w/t$

Merkle-Damgård construction: 2nd preimage vulnerability

Idea of an attack by Kelsey & Schneier (2005)

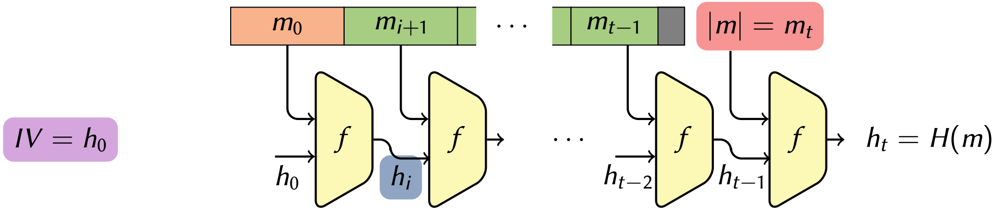


Goal: Given m, find $m' \neq m$ s.t. $H(m') = H(m)$

- Find m_0 such that $f(h_0, m_0) = h_i$ for *any* h_i $\simeq 2^w/t$
  - $m_0 \| m_{i+1} \| \cdots \| m_t$ almost works but m_t contains the wrong length

Merkle-Damgård construction: 2nd preimage vulnerability

Idea of an attack by Kelsey & Schneier (2005)



Goal: Given m, find $m' \neq m$ s.t. $H(m') = H(m)$

- Find m_0 such that $f(h_0, m_0) = h_i$ for *any* h_i $\simeq 2^w/t$
  - $m_0 \| m_{i+1} \| \cdots \| m_t$ almost works but m_t contains the wrong length
- Works if we can find a family of m_0s of variable lengths
  - from *fixed points* $h_f = f(h_f, m_f)$ $\simeq 2^{n/2}$ (in some cases)
  - from *multicollisions*: $m^1, \ldots, m^{2^t}$ s.t. $f(h_0, m^1) = \cdots = f(h_0, m^{2^t})$ $\simeq t \cdot 2^{n/2}$

$\Rightarrow$ 2nd preimage in $\simeq 2^w/t + (t\times)\, 2^{n/2}$ instead of $O(2^n)$

Merkle-Damgård construction: security summary

How vulnerable for 2nd preimage?

- Kelsey-Schneier attack requires to find collisions in f
- Actually: a 2nd preimage *is* a collision!
  - Reduction to collision resistance of $H \to$ collision resistance of f
  - *birthday security* $\simeq 2^{n/2}$

Patch: Chod-MD / Wide-pipe MD (2005)

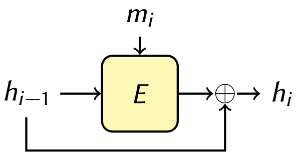
- Use $f : \{0,1\}^{n+k} \times \{0,1\}^{w} \to \{0,1\}^{n+k}$
- Only keep the first n bits of $f(h_{i-1}, m_i)$ as input to next f
- Very strong provable guarantees

Summary

- Same collision resistance for H as for f
- Same 1st preimage resistance for H as for f
- 2nd preimage resistance of H related to collision resistance of f

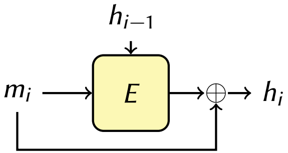
How to design compression functions?

Davies-Meyer construction



$$f(h_{i-1}, m_i) = E_{m_i}(h_{i-1}) \oplus h_{i-1}$$

Matyas-Meyer-Oseas construction



$$f(h_{i-1}, m_i) = E_{h_{i-1}}(m_i) \oplus m_i$$

Security

- Systematic analysis of possible constructions (“PGV constructions”)
- Rigorous proofs in the **ideal cipher model**
  - Not sufficient since actual block ciphers are not ideal!
  - Example: XBOX used a Davies-Meyer based construction with non-ideal cipher

Final words on Merkle-Damgård construction

- Many examples: MD4, MD5, SHA-0, SHA-1, SHA-2, …
- MD5 failure:
  - 1992: Designed by Rivest
  - 1993: Collision attack on the compression function
  - 2005: Collision attack on the hash function
  - 2007-9: Practical useful collisions

  Used up to 2008 (at least), while alternatives were available since (at least) 1996!
- Another bad example: Git chose SHA-1 in 2005 while weaknesses were known

Lessons

- Care about attacks! Even *theoretical*!
- Most (every?) weaknesses can evolve to damaging attacks

Don't design your own crypto!

Contents

Hash function from a permutation

Definition

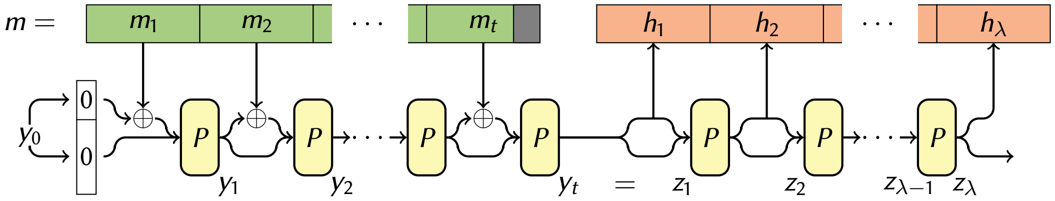
A permutation of $\{0,1\}^n$ is an one-to-one correspondence $P : \{0,1\}^n \to \{0,1\}^n$

- No key – no security notion such as PRP
- Ex.: for any block cipher, $E(0,\cdot)$ is a permutation
- Possible view: block cipher where key and plaintext are given together
- A permutation is invertible, but its inverse is often non necessary

Construction of a hash function

- *Sponge* construction : permutation $\to$ hash function
- Same general idea (but completely different construction) than Merkle-Damgård

The sponge construction



Parameters: r, v and λ

```
1. $m_1 \| \cdots \| m_t \leftarrow \mathrm{pad}(m) = m \| 10 \cdots 0$        $|\mathrm{pad}(m)| = t \cdot r$
2. $y_0 \leftarrow 0^n$
3. for $i = 1$ to $t$: $y_i \leftarrow P(y_{i-1} \oplus (m_i \| 0^c))$        *absorbing phase*
4. $z_1 \leftarrow y_t$
5. for $i = 2$ to $\lambda$: $z_i \leftarrow P(z_{i-1})$; $h_i \leftarrow$ first $v$ bits of $z_i$        *squeezing phase*
6. return $H(m) = h_1 \| h_2 \| \cdots \| h_\lambda$
```

Sponge features

Sponge are convenient!

- If P is a random permutation, H is indifferentiable from a RO
- Flexible:
  - For a fixed permutation size, values of r, v and $\lambda \to$ speed/security trade-off
  - Natively a XOF (variable λ)
- Simplicity: easier to design a (good) permutation

SHA-3 – Keccak

- Hash function using the sponge construction, from a permutation of $\{0,1\}^{1600}$
- Standardized by NIST, after an academic competition (2008-2012)
- Best current choice for a hash function
- Four main variants: SHA3-224, SHA3-256, SHA3-384 and SHA3-512

If you need a hash function, use SHA-3!

Sponge security proof sketch

Theorem

If P is a uniform random permutation and $\lambda = 1$, an adversary making q queries to $P^{\pm}$ has probability $\leq \frac{q^2}{2^v} + \frac{q^2}{2^c}$ to produce a collision.

Admitted claim. If $\mathcal{A}$ produces a collision, at least one of the following events occurs:

- E_1 The adv. makes a query to $P^{\pm}$ whose result ends with 0^c
- E_2 The adv. makes 2 queries to P whose results agree on their first v bits
- E_3 The adv. makes 2 queries to $P^{\pm}$ whose results agree on their last c bits

Proof of the theorem.

Conclusion

Two main families

- Merkle-Damgård construction from a compression function
- Sponge construction from a random permutation
- Many broken constructions, few good ones…

… therefore:

Don’t design crypto yourself!

- No generic way to build a hash function
- Every small detail counts!

Use SHA-3 (or maybe SHA-2)

- Don’t use MD5!
- Don’t use SHA-1!