

Introduction to Modern Cryptography

Michele Ciampi

(Slides courtesy of Prof. Jonathan Katz)

Lecture 11, Part 1

Hash Functions

Hash Functions and Message Authentication

Recall

- ▶ We showed how to construct a secure MAC for short, **fixed-length messages** based on any PRF/block cipher
- ▶ We extended this to a secure MAC for **arbitrary-length messages** using CBC-MAC

Question

Can we use **hash functions** to construct a secure MAC for **arbitrary-length messages**?

Hash Functions

(Cryptographic) hash function

Deterministic function mapping arbitrary length inputs to a short, fixed-length output (a digest)

Keyed or unkeyed

- ▶ In practice, hash functions are unkeyed
- ▶ Theoretically, need to be keyed: key is public
- ▶ Assume unkeyed hash functions for simplicity

Collision-resistance

Collision

Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$ be a hash function. A **collision** is a pair of distinct inputs x, x' such that $H(x) = H(x')$

Collision-resistance

H is **collision-resistant** if it is infeasible to find collision in H

Generic Hash Function Attacks

Observation

Collisions are guaranteed to exist

Generic Attack Complexity

- ▶ What is the best **generic** collision attack on a hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$?
- ▶ If we compute $H(x_1) \dots H(x_{2^l+1})$, we are guaranteed to find a collision (why?)
- ▶ Can we do better?

Birthday Paradox

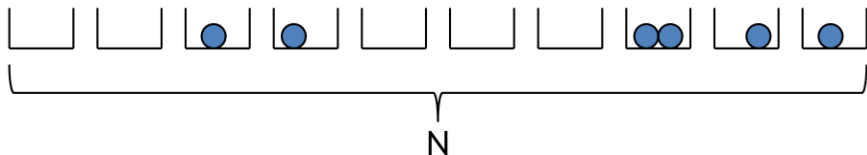
- ▶ Compute $H(x_1) \dots H(x_k)$
- ▶ What is the probability of a collision (as a function of k)?
- ▶ i.e. how many hashes do we need ($k=?$) in order to find a colliding pair $H(x_i) = H(x_j)$?

The Birthday Paradox

How many people are needed to have a **50%** chance that some two people share a birthday?

Birthday Paradox: Balls and Bins Experiment

How many balls do we need to throw to have a **50%** chance that two balls fall in the same bin (a collision)?



- **Bins** = days of year $N = 365$ (#hashes $N = 2^l$)
- **Balls** = k people (k hash function inputs)

Birthday Attack

Theorem

The collision probability is $k(k-1)/2N = \mathcal{O}(k^2/N)$

- ▶ Consider **pairs of elements** $\implies k(k-1)/2$ pairs
- ▶ When $k \approx \sqrt{N}$, probability of a collision is $\approx 50\%$
- ▶ $k = 23$ people suffice: $23 \cdot 22/2 \cdot 365 = 253/365 \approx 0.7$
- ▶ $k \approx \sqrt{2^l}$ hash-function evaluations: $\mathcal{O}(\sqrt{2^l}) \equiv \mathcal{O}(2^{l/2})$

Note

In the analysis, $\mathbf{H}$ is modelled as a random function $\implies$
worst case in terms of $\mathbf{Pr}$

Security Implications of the Birthday Attack

Implication

To protect against attackers running in time 2^n we need the output of our hash function to be $l = 2n$

- i.e. twice as long as symmetric keys for the same security

Comparison to Encryption Algorithms

To protect against attackers running in time 2^n (e.g. brute-force attack) we need the key to our symmetric-key algorithm (e.g. block-cipher keys, PRG seeds) to be n

Example

To ensure **128** bit security we need a block cipher with **128** bit key and a hash function with **256** bit output

Birthday Bound

The birthday bound $2^{n/2}$ comes up in many other cryptographic contexts

Example

IV reuse in CTR-mode encryption:

- ▶ If k messages are encrypted, what are the chances that some **IV** is used twice?
- ▶ Note: this is much higher than the probability that a specific **IV** is used again

Building a Hash Function

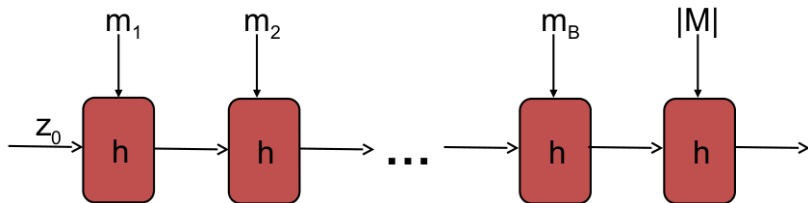
Two-stage approach

1. Build a **compression function h** i.e. hash function for fixed-length inputs
2. Build a **hash function H** for arbitrary length inputs from a compression function **h**

Building a Hash Function

- ▶ Assume we have a “good” compression function h
 - ▶ i.e. collision-resistant for fixed-length inputs
- ▶ (Will discuss how to construct such an h later)
- ▶ Construct a hash function H (for arbitrary length inputs) based on h
- ▶ Prove that collision resistance of h implies collision resistance of H

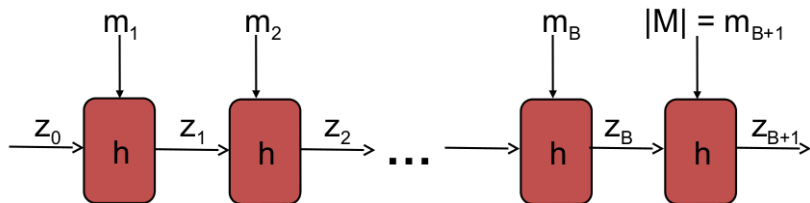
Merkle-Damgård Transform



Claim

If h is collision-resistant, then so is H

Merkle-Damgård Transform



Proof.

Collision in $H \implies$ collision in h

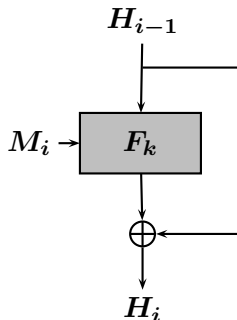
- ▶ Say $H(m_1 \dots m_B) = H(m'_1 \dots m'_{B'})$
- ▶ $|M| \neq |M'|$, look at the last block
- ▶ $|M| = |M'|$, look at largest i with $(z_{i-1}, m_i) \neq (z'_{i-1}, m'_i)$

□

Compression Function from a PRF/Block Cipher

Davies-Meyer

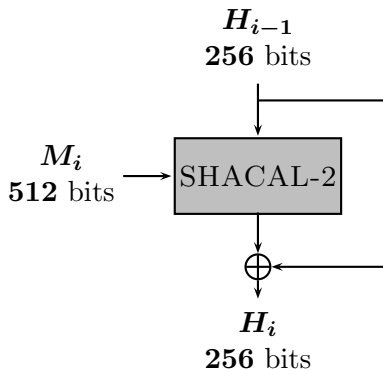
The Davies-Meyer construction is a method to transform a **block cipher** into a **compression function** using a feedforward and the message block as the key



Example: SHA-256

SHA-256

Merkle-Damgård + Davis-Meyer + Block cipher (SHACAL-2)



Hash Functions in Practice

MD5 (**broken!**)

- ▶ Developed in 1991
- ▶ **128-bit** output length
- ▶ Collisions found in 2004, **should no longer be used**

SHA-1 (**broken!**)

- ▶ Introduced in 1995
- ▶ **160-bit** output length
- ▶ Collision found in 2017 (fixed prefix) and in 2020 (chosen prefix); **should no longer be used**

Hash Functions in Practice

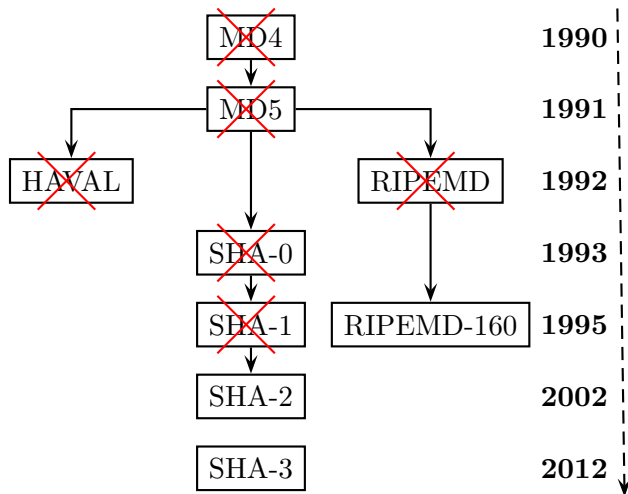
SHA-2

- ▶ Introduced in 2001
- ▶ Versions with **224**, **256**, **384**, and **512**-bit outputs
- ▶ No significant known weaknesses

SHA-3/Keccak

- ▶ Result of a public competition from 2008-2012
- ▶ Very different design than SHA-1/SHA-2
 - ▶ Does not use Merkle-Damgård transform
- ▶ Supports **224**, **256**, **384**, and **512**-bit outputs

Hash Functions History



Hash Functions and Message Authentication

Recall

We showed how to construct a secure MAC for short, **fixed-length messages** based on any PRF/block cipher

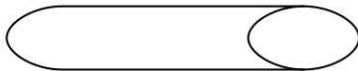
Question

Can we use **hash functions** to construct a secure MAC for **arbitrary-length messages**?

Main Idea



M



- ▶ A and B share a **reliable channel** that can handle short messages
- ▶ A wants to send reliably a **long message** M

Main Idea



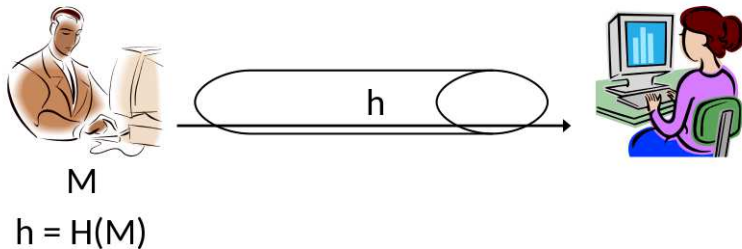
M

$$h = H(M)$$



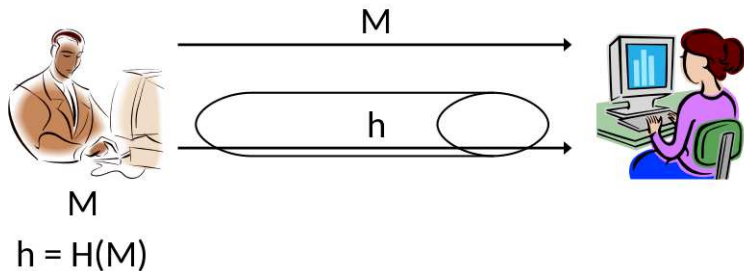
- A hashes the long message M to a shorter fixed-length digest $h = H(M)$

Main Idea



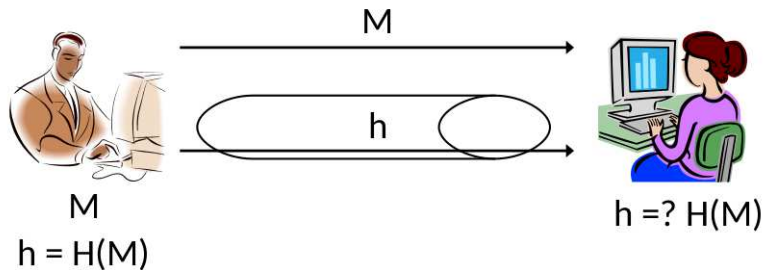
- A sends h over the reliable channel

Main Idea



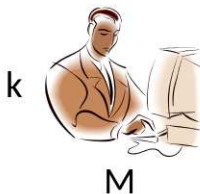
- A sends M over the general (unreliable) channel

Main Idea



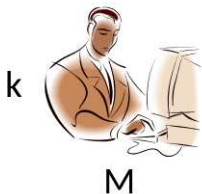
- ▶ B receives M and recomputes its hash $h = H(M)$
- ▶ B checks whether h matches the hash received by A
- ▶ If no match $\implies$ the long message M has been modified

Hash-and-MAC



- ▶ A and B share a key k ; A transmits long message M
- ▶ The reliable channel for short messages is replaced by a MAC for short messages

Hash-and-MAC

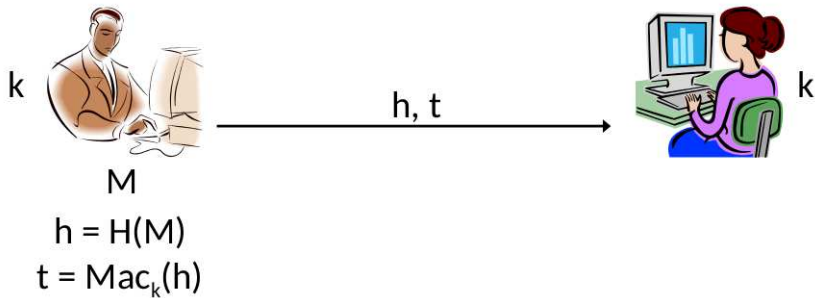


$$h = H(M)$$
$$t = \text{Mac}_k(h)$$



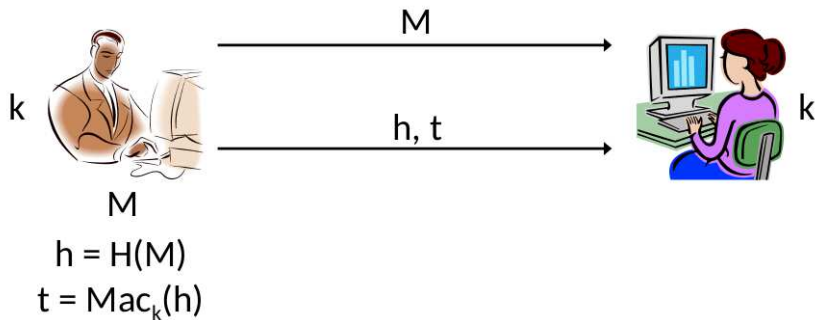
- ▶ A computes the hash $h = H(M)$
- ▶ A authenticates the hash with the tag $t = \text{Mac}_k(h)$

Hash-and-MAC



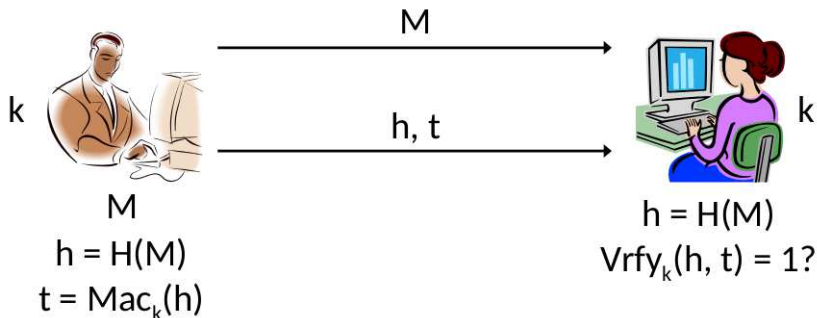
- A transmits the hash and the tag h, t

Hash-and-MAC



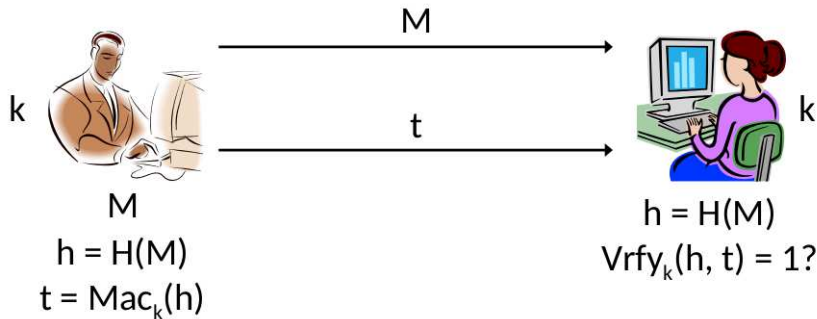
- A transmits the long message M

Hash-and-MAC



- ▶ B receives M and recomputes its hash $h = H(M)$
- ▶ B verifies the received tag t by $\text{Vrfy}_k(h, t)$
- ▶ If $\text{Vrfy}_k(h, t) = 1 \implies M$ has not been modified

Hash-and-MAC



Not necessary to transmit h as B can recompute it from M

Proof of Security

Claim

If the MAC is secure for fixed-length messages and H is collision-resistant, then the [previous] construction is a secure MAC for arbitrary-length messages

Proof sketch

- ▶ The sender authenticates messages $M_1, M_2, \dots$
- ▶ As usual the attacker can choose (adaptively) $M_1, M_2, \dots$
- ▶ Attacker outputs forgery $(M, t) : M \neq M_i, \forall i$
- ▶ Two cases:
 1. $H(M) = H(M_i)$ for some $i \implies$ collision in H
 2. $H(M) \neq H(M_i) : \forall i \implies$ forgery in the underlying, fixed-length MAC

Instantiation

Question

Can we instantiate the described scheme using a hash function (e.g. SHA2) and a block cipher-based MAC (e.g. AES as a PRF)?

Problems

- ▶ Block-length mismatch (e.g. **128** bits for AES vs. **256** bits for SHA256)
- ▶ Need to implement two crypto primitives (block cipher and hash function)

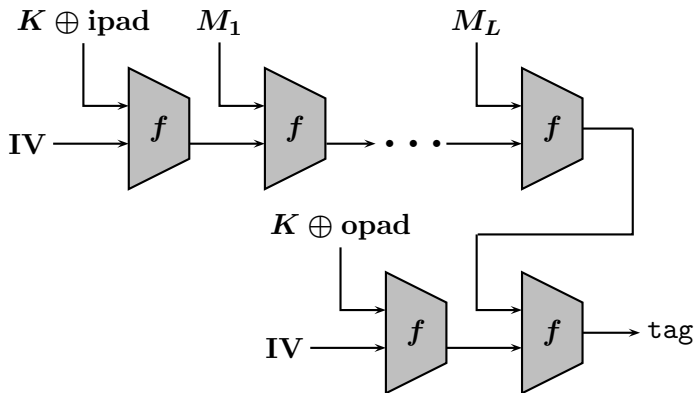
Solution: HMAC

HMAC

HMAC is a practical instantiation of the hash-and-MAC paradigm

- ▶ Constructed entirely from Merkle-Damgård hash functions
 - ▶ MD5, SHA-1, SHA-2
 - ▶ Not SHA-3
- ▶ Follows the hash-and-MAC approach with (part of) the hash function being used as a PRF

HMAC [Bellare,Canetti,Krawczyk,1996]



- **ipad**: inner padding (the byte 0x36 repeated $|K|$ times).
- **opad**: outer padding (the byte 0x5C repeated $|K|$ times).

End

References: Sec 5.1, 5.2, 5.3.1, 5.4.1 (not Theorems 5.4 and 5.6)