

# CSE 207B: Applied Cryptography

**Nadia Heninger**

UCSD

Fall 2025 Lecture 6

# Announcements

1. HW 2 is due today!
2. HW 3 is out, due before class in 1 week

**Last time:** Message integrity and MACs

**This time:** Hash functions

# Cryptographic hash functions

## Definition (Practical definition)

A hash function is an efficiently computable function that maps arbitrary-length inputs to fixed-length outputs

$$H(x) : \{0, 1\}^* \rightarrow \{0, 1\}^n$$

We would like to define  $H$  as secure if it is collision resistant.

**Collision resistant:** No efficient adversary can find messages  $m_0$ ,  $m_1$  such that

$$H(m_0) = H(m_1)$$

# Cryptographic hash functions

## Definition (Practical definition)

A hash function is an efficiently computable function that maps arbitrary-length inputs to fixed-length outputs

$$H(x) : \{0, 1\}^* \rightarrow \{0, 1\}^n$$

We would like to define  $H$  as secure if it is collision resistant.

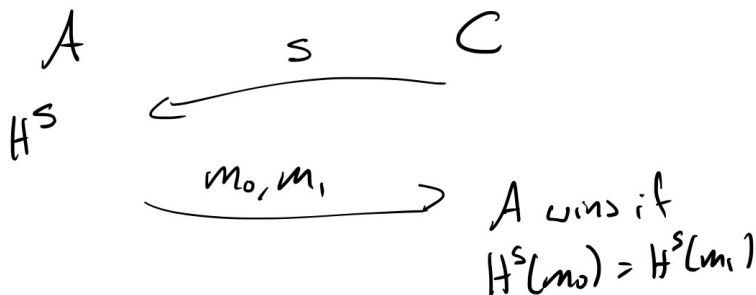
**Collision resistant:** No efficient adversary can find messages  $m_0$ ,  $m_1$  such that

$$H(m_0) = H(m_1)$$

This definition doesn't work in a theoretical sense: By the pigeonhole principle there exist such collisions, so there exists an adversary  $A$  that just prints them out.

# Security game for a collision-resistant hash function

Fix for theoretical proofs: Define a family of hash functions that depends on a parameter that the adversary doesn't know in advance, and define runtime in terms of a security parameter.



## Definition

$H$  is collision-resistant if  $\Pr[A \text{ succeeds}] \leq \text{negligible}$ .

# Practical hash functions

In actual practice, hash functions are fixed functions where all parameters were generated when the algorithms were designed/standardized. (Think MD5, SHA1, SHA256, SHA3.)

So “collision-resistance” means “the best cryptanalytic attacks to find a collision for this fixed function are expected to have infeasible runtimes”.

# Security notions for hash functions

1. Collision resistance:  
Find  $m_0, m_1$  such that  $H(m_0) = H(m_1)$ .
2. 2nd preimage resistance:  
Input  $m_0$ , find  $m_1$  s.t.  $H(m_0) = H(m_1)$ .
3. Preimage resistance:  
Input  $t = H(m_0)$ , find  $m_1$  s.t.  $H(m_1) = t$ .

Strong  
↓  
Weak

# Generic collision-finding attacks on hash functions

## Inefficient pigeonhole algorithm

If  $H$  has  $\ell$ -bit outputs, then after hashing  $2^\ell + 1$  distinct inputs there must be a collision.

We can do much better than this with the birthday “paradox”.

# Birthday attacks

Let  $H$  be a hash function with  $N = 2^n$  possible outputs.

## Birthday Attack Algorithm

1. Choose  $k$  uniformly random messages  $m_1, \dots, m_k$ .
2. Compute  $t_i = H(m_i)$  for every  $i$ .
3. Search for  $i, j$  s.t.  $m_i \neq m_j$  and  $H(m_i) = H(m_j)$ .

## Theorem

$$\Pr[\text{collision after } \Theta(\sqrt{N}) \text{ inputs}] \geq 1/2$$

# Birthday collision theorem

## Theorem

$$\Pr[\text{collision after } \Theta(\sqrt{N}) \text{ inputs}] \geq 1/2$$

## Proof.

We throw  $k$  balls uniformly at random into  $N$  distinct bins.

$$\Pr[k \text{ balls into } N \text{ distinct bins}] = 1 \cdot \left(1 - \frac{1}{N}\right) \cdot \left(1 - \frac{2}{N}\right) \dots \left(1 - \frac{k-1}{N}\right)$$

Use  $1 - \frac{1}{N} \lesssim e^{-1/N}$ .

$$\Pr[\text{distinct}] \leq \prod_{i=1}^{k-1} e^{-i/N} = e^{-1/N \sum_{i=1}^{k-1} i} = e^{-k(k-1)/2N}$$

For  $k = \sqrt{2N} = O(\sqrt{N})$ :

$$\Pr[\text{collision}] = 1 - \Pr[\text{distinct}] \geq 1 - e^{-k^2/2N} \geq 1 - e^{-1} \geq 1/2$$

# Birthday Attack Algorithm Complexity

The previous algorithm requires:

- $\sqrt{N} = 2^{n/2}$  hash evaluations
- $n\sqrt{N} = n2^{n/2}$  memory

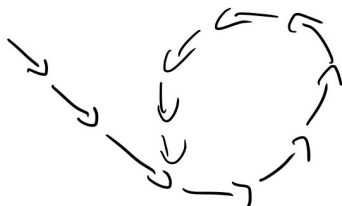
It is possible to find collisions with less memory using the Floyd cycle-finding algorithm.

# Floyd cycle-finding algorithm

## Floyd cycle-finding algorithm

1. Start at an arbitrary  $x_0 = y_0$ .
2. Compute  $x_i = H(x_{i-1})$ ,  $y_i = H(H(y_{i-1}))$ .
3. If  $x_i = y_i$ , then  $m_0 = x_{i-1}$  and  $m_1 = H(y_{i-1})$  collide.
4. To find collision at beginning of cycle, restart  $x$  pointer from start and single-step both  $x$  and  $y$ .

By previous analysis, there is a cycle of length  $\sqrt{N}$  assuming  $H$  behaves like a random walk.



Algorithm time:  $O(\sqrt{N})$   
Memory:  $O(1)$

Side note: Can get parallelization and time/memory tradeoff using method of distinguished points.

Any collision-resistant hash function needs output at least 256 bits in practice.

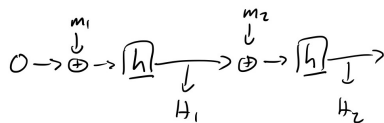
How do you construct a hash function that takes arbitrary-length inputs?

Assume we have a fixed-length hash function  $h$

Does CBC-hash work?

### Strawman CBC-hash construction

1. Set  $h(m) = \text{AES}_0(m)$ .
2. Set  $H_0 = 0000 \dots 0$ .
3. Let  $H_i = \text{AES}_0(H_{i-1} \oplus m_i)$
4. Output  $H_k$



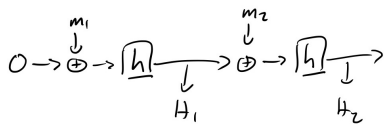
How do you construct a hash function that takes arbitrary-length inputs?

Assume we have a fixed-length hash function  $h$

Does CBC-hash work?

### Strawman CBC-hash construction

1. Set  $h(m) = \text{AES}_0(m)$ .
2. Set  $H_0 = 0000 \dots 0$ .
3. Let  $H_i = \text{AES}_0(H_{i-1} \oplus m_i)$
4. Output  $H_k$



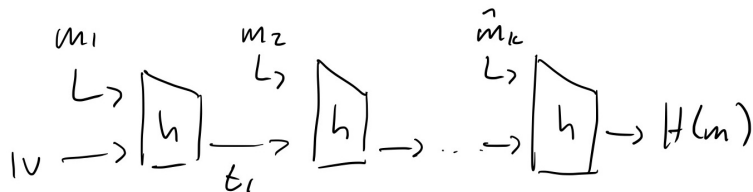
No: Let  $m = m_1 || m_2$  and  $m' = m'_1 || m'_2$  with  $m'_1 = m_2 \oplus H_1$  and  $m'_2 = H_2 \oplus m_2 \oplus H_1$ .

**Conclusion:** CBC mode on its own does not lead to a secure hash function.

# Merkle-Damgård Construction

Assume we have a fixed-length compression function  $h$ .

1. Input message  $m = m_1 || m_2 || \dots$
2. Pad final block to  $\ell$  bits:  $\hat{m}_k = m_k || \text{pad} || \text{len}(m)$ .
3. Set  $t_0 = IV$ .
4. For  $i = 1$  to  $k$  do:  $t_i = h(t_{i-1}, m_i)$
5. Output  $t_k$

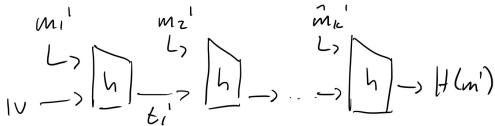
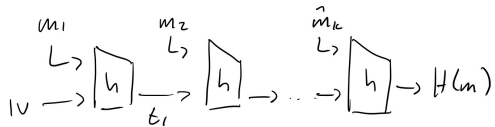


## Theorem

If  $h$  is a fixed-length collision-resistant hash function then  $H$  is a collision-resistant hash function.

## Proof.

Assume have a collision  $(m, m')$  for  $H$ . Construct a collision for  $h$ .

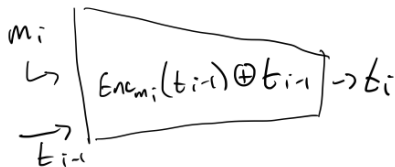


Case 1 Different message lengths  $K \neq K'$ . At last step,  
 $h(t_{k-1}, m_k || \text{pad} || K) = h(t'_{k'-1}, m'_{k'} || \text{pad} || K')$  is a collision.

Case 2 Last block identical;  $K = K'$ . Then collision occurs at some intermediate block. Find last point with  $t_{i-1}, m_i \neq t'_{i-1}, m'_i$ .  
Then  $h(t_{i-1}, m_i) = h(t'_{i-1}, m'_i)$ .

# Davies-Meyer compression functions

Can construct collision-resistant compression functions from block ciphers.



This is collision-resistant.

Requires specialized block ciphers: 128-bit AES is too short to ensure collision resistance.

Need block ciphers that take 512 or 1024-bit keys.

## Hash functions in practice

- MD5: Totally insecure.
- SHA1: Totally insecure.
- SHA2 (SHA256, SHA512): Secure
- SHA3: New hotness standardized by NIST in 2015.

# The long saga of MD5

MD5 was a popular hash function using the Merkle-Damgård construction.

1991 Rivest publishes MD5

1993 “some weaknesses”

1995 SHA1 proposed by NIST

2004 Wang et al. collision for MD5 in Crypto rump session

2009 Rogue CA certificate

2010 NIST recommends replacing SHA1

2012 Flame malware exploits MD5 collision in wild against Microsoft CA

MD5 collisions can be computed in seconds on a laptop.

# The continuing saga of SHA1

SHA1 is a popular hash function using the Merkle-Damgård construction.

1993 SHA0 published by NIST

1995 NIST adds one instruction to compression function; turns out to be necessary for collision resistance

2005 Wang, Yao, Yao  $2^{63}$  collision attack

2017 Stevens et al. first collision for SHA1 in  $2^{63.1}$  evaluations; 6500 CPU-years + 110 GPU-years

2019 Chosen-prefix collisions in  $2^{68}$  evaluations

# SHA2

The SHA2 hash functions use Merkle-Damgård construction with Davies-Meyer compression function.

2002 NIST adds SHA256 and SHA512 to SHA family

- SHA224 and SHA384 are truncated versions of these.
- SHA256 uses 512-bit message blocks and 256-bit chaining variables.
- Recommended for use now.

# SHA3

SHA3 uses a new construction, a sponge.

2006 NIST organizes a new hash function competition

- Goal to have a more “diverse” algorithm portfolio
- Intended to be drop-in replacement for SHA2

2012 Keccak algorithm (Bertoni, Daemen, Peeters, Van Aasche) announced as competition winner

2013 NIST announces decrease to the security parameter for standard, uproar ensues

- Among other things, algorithm would be less quantum resistant

2013 NIST walks back change to original security parameter

2015 NIST publishes standard

- Recommended for use now.

HW 3 due in 1 week!