Collision - Resistant Hash Functions

Plan

* Intuition & motivation
* Definition of CRHF
* Constructions
* More applications

Logistics

* Lab 0 code & Lab 0 theory due tomorrow 10pm ET
  → Gradescope
  → LaTeX for written parts

* Lab 1 out on Friday

* Interrupt any time!
  Hard to see faces w/ masks... use eyebrows?
Last time...

**Authenticating PEOPLE**

![Diagram](image)

Password, pass storage, MACs, biometrics...

Today...

**Authenticating FILES / CODE / DATA**

Main new tool: Collision-resistant hash functions. (CRHF)
Intuition behind CRHF....

A hash function $H: \{0,1\}^* \rightarrow \{0,1\}^{256}$

In practice $SHA2/SHA256, SHA3/Keccak, ...$ (broken: $MD5, SHA1$)

Compressing!

Security goal: It is "computationally infeasible" to find two distinct files that have the same hash value (a "collision").
Application I: Secure mirroring

1. Get hash from trustworthy source
   - Source (e.g., software vendor)
   - $h = H(f)$

2. Fetch large file from untrustworthy source
   - Sketchy mirror
   - $h = H(\hat{f})$
   - Client: accept if so

If hash is CRHF, then sketchy mirror will not be able to find a file $\hat{f} \neq f$. That client will accept.
Application II: Outsourced File Storage

1. \( h = H(\text{photos}) \)

2. \( h = H(\hat{\text{photos}}) \)

If hash is cDHF, then Google can't trace you into accepting incorrect photos/files.
More generally, CRHFs let you authenticate a LONG message by authenticating only a SHORT string.

We will see more applications—"Hash and sign"...
Adversary's goal in breaking ORHF.

\[
m_0, m_1 
\rightarrow \quad \text{Distinct msgs s.t.} \quad H(m_0) = H(m_1)
\]

Observe: There are lots of collisions!

If ORHF is good/secure, those collisions will be hard to find.

\[
\text{Mega Pigeonhole Principle!} \quad \text{Jamming infinitely many pigeons in finite holes}
\]

How do we formalize this?
Definition: Collision-Resistant Hash Function

A function \( H : \{0,1\}^k \rightarrow \{0,1\}^A \) is collision resistant if for all "efficient" adversaries \( A \)

\[
\Pr[\exists \mathbf{m}_0 \neq \mathbf{m}_1 : (\mathbf{m}_0, \mathbf{m}_1) \leftarrow A() \mid H(\mathbf{m}_0) = H(\mathbf{m}_1)] \leq "\text{negligible}".
\]

(To be useful, \( H \) must also be efficiently computable.)

In theory:

\( \lambda = "\text{security parameter} " \quad (\approx \text{key length}) \)

"efficient" = randomized alg runs in time \( \text{poly}(\lambda) \)

"negl" = \( O\left(\frac{1}{\lambda^c}\right) \quad \forall \lambda \in \mathbb{N} \)

(e.g., \( \frac{1}{2^{\lambda}}, \frac{1}{2^{\lambda^2}}, \frac{1}{2^{\lambda \log \lambda}}, \ldots \))

In practice:

\( \lambda = 128, 256, 384 \)

"efficient" adversary = runs in time \( \leq 2^{128} \)

"negl" = \( \text{prob} \leq 2^{-128} \)
In practice, aim to defend against attacks running in time \(2^{128}\).

**Time**

- \(2^{30}\) ops/sec on your laptop
- \(2^{68}\) ops/sec on Fugaku supercomputer \((\approx \$1\ billion)\)
- \(2^{66}\) hashes/sec computed by Bitcoin miners
- \(2^{90}\) hashes/year
- \(2^{114}\) hashes requires enough energy to boil all water on the planet
- \(2^{140}\) hashes requires one year of Sun's energy

**Probability**

- \(2^{-1}\) fair coin lands heads
- \(2^{-8}\) tax returns audited by IRS
- \(2^{-13}\) probability that randomly sampled MIT grad is Nobel prize winner
- \(2^{-19}\) struck by lightning in next year
- \(2^{-28}\) probability of winning Mega Millions jackpot (now \$20m)
- \(2^{-69}\) probability of all happening \(\ldots\) (assuming independence)
- \(2^{-128}\) ... a billion billion times less likely than that.
How to construct CRHF:

Two steps:

1. Construct CRHF \( H_{small} : \{0, 1\}^2 \to \{0, 1\}^3 \)

   - More art than science.
   - Come up with candidate, try to break it using known techniques, assume it's CRHF

2. Current standard is SHA256, designed by NSA, published 2001

   - Can also build from number theory (factoring, etc)
   - ...but too slow

[Aside: If P=NP, CRHFs don't exist.
So security of CRHFs relies on unproven assumptions... P≠NP & more]
2. Use $H_{\text{small}}$ to construct $H: \{0,1\}^* \to \{0,1\}^\lambda$

"Merkle–Damgård"

$H$ is CRHF if $H_{\text{small}}$ is

(No need for extra assumption)

Need to be careful about padding - don't implement yourself.
Why Merkle-Damgard works

Consider hash fn $h_{bi_3} : \{0,1\}^n \rightarrow \{0,1\}^\lambda$

Claim: If $\exists$ eSS eff adv $A$ that finds collisions in $H_{big}$,  
$\exists$ eSS eff adv $B$ that finds collisions in $H_{small}$

Run $A() \rightarrow [ (m_0, m_1, m_2), (m_0', m_1', m_2') ]$

$(x, m_2) \neq (x, m_2')$  

$(x, m_2), (x, m_2')$ is a collision for $H_{small}$!

$(m_0, m_1) = (m_0', m_1')$  

Contradiction! $(m_0, m_1, m_2) = (m_0', m_1', m_2')$
Given hash \( f_n \) with \( n \)-bit output, can find collision in time \( O(2^{n/2}) \).

\[ \text{Versus } 2^n \text{ for brute-force search} \]

\( \Rightarrow \) If you want adv to do \( 2^{128} \) work \( \leq \) to find collision, need to have 256-bit output.

\( \Rightarrow \) In practice, we use SHA256 (or SHA3)

(on my laptop, get \( \approx 1 \) GB/s)

openssl speed sha256

---

**Historical Note:**

\* For many years, MD5 (designed by Ron Rivest) was the standard CRHF - 128-bit output

\* 2004: Wang et al. find collision - time is now \( \approx 2^{63} \)

\* We used to use SHA1 (160-bit output)

\* In 2017 researchers at CWI AMS & Google found a collision in SHA1 using \( 2^{63} \) hashes

\* Attack cost \( \leq \$100k - \$500k \)

\( \Rightarrow \) SHA1 deprecated
Given: \( H: \Sigma^* \rightarrow \{0,1\}^n \) [model \( H \) as a random function]

Find: \( m_0, m_1 \in \Sigma^{2^n} \) s.t. \( m_0 \neq m_1, H(m_0) = H(m_1) \).

Let \( T = 2^{n/2} \)

Choose distinct \( r_1, r_2, r_3, \ldots, r_T \in \Sigma^{2^n} \)

Compute \( H(r_1), H(r_2), \ldots, H(r_T) \).

\( \implies \) Likely to find a collision!

Let \( B_i = \) event that \( \exists \) collision after computing \( i \)th hash

\[
Pr[B_i | B_{i-1}] = 1 - \frac{i}{2^n}
\]

\[
Pr[\text{no collision}] = Pr[B_T] = Pr[B_T | B_{T-1}] \cdot Pr[B_{T-1}]
\]

\[
= \frac{1}{1 - \frac{i}{2^n}} \cdot \frac{1}{1 - \frac{i-1}{2^n}}
\]

\[
\leq \exp \left( \sum_{i=1}^{T} \frac{i}{2^n} \right) \leq \exp \left( \sum_{i=1}^{T} \frac{i^2}{2^n} \right)
\]

\[
Pr[\text{collision}] \geq 1 - \text{constant}
\]

\( \sim \) repeat a few times
**Application:** Merkle trees

(Authenticating many files with a single digest)

A variant on our secure mirroring application...

\[ f_i \rightarrow f_N \]

\[ h \]

\[ X \rightarrow h \]

Source sends \( N \) hashes

- a lot of communication over wide area net

Client downloads all \( N \) files

- even more communication

Better idea: Use the Merkle construction
\[ h \]

\[ h_0, h', h'' \]

\[ s_3 \]

\[ s_1, s_2, s_3, s_4 \]

\[ \text{Source} \]

\[ \text{Mirror} \]

\[ \Rightarrow \text{Mirror sends one file plus } O(\log N) \text{ hashes} \]

\[ \ll \text{ than } N \text{ hashes} \]

\[ \ll \text{ than } N \text{ files} \]

\[ \Rightarrow \text{CP property ensures that mirror can't cheat} \]
Application: Commitments

* "Sealed envelope" with cryptography.
* Just a small tweak to the earlier applications.
* Requires a bit more than plain CHTF, but any CHTF can be made suitable.

"Coin Flipping"

\[ b_A \leftarrow \{0,1\} \]
\[ r \leftarrow \{0,1\} \]
\[ h = \text{Hash}(b_A, r) \]

Alice

\[ \text{Bob} \]

\[ b_0 \leftarrow \{0,1\} \]
\[ b' = b_A \oplus b_0 \]

\[ h \leftarrow \text{Hash}(b_A, r) \]
\[ b = b_A \oplus b_b \]

Modulo Alice refusing to open, neither party can control bit b.

\[ \text{Distributed randomness used for protocols that require good randomness, i.e. trusted dealer (e.g. lott...)} \]
The End