Hash Functions

Ali El Kaafarani

[1]Mathematical Institute
[2] PQShield Ltd.

Outline

Introduction

- Informally speaking, hash functions take a long input string and output a shorter string called a *digest*.
- They are used almost everywhere in Cryptography.
- If you *imagine* that hash functions are truly random (modelled as *random oracle model*), then proving the security of some cryptographic schemes becomes achievable (e.g. RSA-OAEP).
- A debate/controversy over the soundness of the random oracle model.
- Cryptographic hash functions are much harder to design than those used to build *hash tables* in data structures.

Notions of Security-Collision Resistance

- Given a hash function H, it should be infeasible for any PPT algorithm to find $x \neq x'$ s.t. $H(x) = H(x')$.
- Remember that the domain of H is larger than its range, therefore collisions must exist.
- We want these collisions to be hard to find.
- Keyed hash functions take as input a key s and a string x.
- This time the key is not a secret, i.e. collision resistance should hold even when this key is in the adversary's hands.
- We denote a keyed hash function by H^s for a key s.

Keyed Hash Functions: a Definition

Definition

A keyed hash function consists of two PPT algorithms (KeyGen, H) *which can be defined as follows:*

- $\text{KeyGen}(1^n)$: *it takes a security parameter* n *and outputs a key* s*.*
- $H(s, x \in \{0,1\}^*)$: *it takes a key* s *and a string* $x \in \{0,1\}^*$ *and outputs a string* $H^s(x) \in \{0,1\}^{\ell(n)}$

Collision Resistance

Given a keyed hash function H, an adversary A, and a security parameter n, we define the collision-finding experiment $\mathsf{Hash}^{coll}_{\mathsf{A},H}(n)$ as follows:

- A key is generated by $KeyGen$ and is given the adversary.
- Adversary's output: two strings x and x'
- Experiment's output: 1 iff $x \neq x'$ and $H^s(x) = H^s(x')$

Definition

A hash function H is collision resistant if for all PPT adversaries A *we have*

$$\Pr[\mathsf{Hash}^{coll}_{\mathsf{A},H}(n) = 1] \leq \mathsf{negl}(n)$$

Hash Functions in Practice

- They are *unkeyed* with fixed output i.e. $H : \{0,1\}^* \to \{0,1\}^\ell$.
- Theoretically speaking, you can always find a collision using a constant-time algorithm.
- However, they are computationally hard to find.
- This shouldn't affect the security proofs as long as it shows that the adversary who can break a cryptographic primitive that uses a certain hash function can *in practice* find a collision!

Weaker Security Notions

- *Second-preimage* or *target-collision resistant*: Given s and a uniform x, it is hard for any PPT adversary to find x' s.t. $x \neq x'$ and yet $H^s(x) = H^s(x')$
- *Preimage resistance* or *one-wayness*: Given s and a uniform y, it is hard for any PPT adversary to find x s.t. $H^s(x) = y$

Note that: collision resistance $\Rightarrow$ second preimage resistance $\Rightarrow$ preimage resistance. (Check them!)

Outline

How to Design a Hash Function?

- First, consider a collision-resistant compression function (handling only fixed-length inputs).
- Second, apply a domain extension method to deal with arbitrary-length inputs.
- This should maintain the collision-resistance property.
- Merkle-damgård transform is a very famous approach for domain extension.
- It has been used for MD5 and the SHA family.
- Theoretical implication of Merkle-damgård: if you can compress by a single bit, then you can compress by an arbitrary amount of bits!

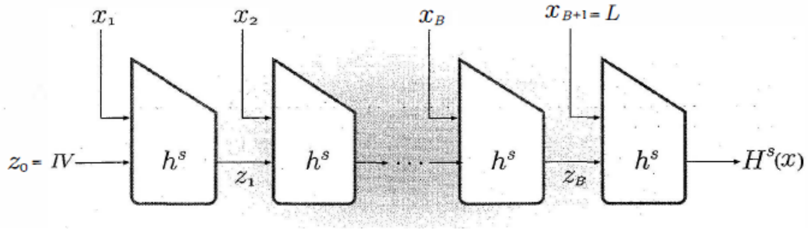
The Merkle-damgård Transform

Given a fixed-length hash function h that takes inputs $\in \{0,1\}^{2n}$ and outputs digests $\in \{0,1\}^n$. We construct an arbitrary-length hash function as follows:

- KeyGen : No Change to it.
- H : it takes a key s and a string $x \in \{0,1\}^*$ of length $L < 2^n$ and does the following:
  - Set the number of blocks in x as $B \leftarrow \left\lceil \frac{L}{n} \right\rceil$ and pad with zeros to get the following sequence of n-bit blocks, i.e. $x_1, \cdots, x_B$. Set $x_{B+1} \leftarrow L$, where L is encoded as an n-bit string.
  - Set $z_0 \leftarrow 0^n$ (also called IV)
  - Compute $z_i \leftarrow h^s(z_{i-1}||x_i)$, for $i = 1, \cdots, B+1$.
  - Output z_{B+1}.

The Merkle-damgård Transform

[Katz-Lindell]



Theorem

If h is collision-resistant, then so is H.

The Merkle-damgård Transform

Proof.

We show that a collision in H would definitely lead to a collision in h. Suppose that we have $x \neq x'$ of length L and L' such that $H^s(x) = H^s(x')$. We will try to find a collision in h^s. We pad x and x' to get $x_1, \cdots, x_B$ and $x'_1, \cdots, x'_{B'}$, and we distinguish between two cases:

- $L \neq L'$: then $z_B || L \neq z'_{B'} || L'$, but since $H^s(x) = H^s(x')$, then $h^s(z_B||L) = h^s(z'_{B'}||L')$ therefore a collision in h^s is found.
- $L = L'$: in this case $B = B'$. One can compute both $H^s(x)$ and $H^s(x')$ and store all the intermediate values. Compare all the inputs to h^s, i.e. $z_{i-1}||x_i$ and $z'_{i-1}||x'_i)$. We know that $x \neq x'$ but $|x| = |x'|$ therefore there must exist an $1 \leq j \leq B$, for which $x_j \neq x'_j$. Output the pair $z_{j-1}||x_j$ and $z'_{j-1}||x'_j$ as a collusion in h^s.

□

Outline

MAC using Hash Functions

- A different approach to construct a MAC for arbitrary-length messages.
- The idea is simple and widely used in practice (e.g. HMAC).
- Firstly, use a collision resistant hash function H to hash an arbitrary-long message down to a fixed-length $H^s(m)$.
- Secondly, apply a fixed-length MAC to the digest of the hash function.

Hash-and-MAC

Given a message authentication code $S_{mac} = (\text{Mac}, \text{Verify})$ for message of length $\ell(n)$ and a hash function H with output length $\ell(n)$. We define a new MAC $S'_{mac} = (\text{KeyGen}', \text{Mac}', \text{Verify}')$ for arbitrary-length messages as follows:

- KeyGen$'(1^n)$: it takes an security parameter n, and output a uniform key $k \in \{0,1\}^n$ and it runs the key generator of the hash function to get s. the final key will (k, s).
- Mac$'(k, s, m \in \{0,1\}^*)$: it outputs $t \leftarrow \text{Mac}_k(H^s(m))$.
- Verify$'(k, s, m \in \{0,1\}^*, t)$: it outputs 1 iff $\text{Verify}_k(H^s(m), t) = 1$.

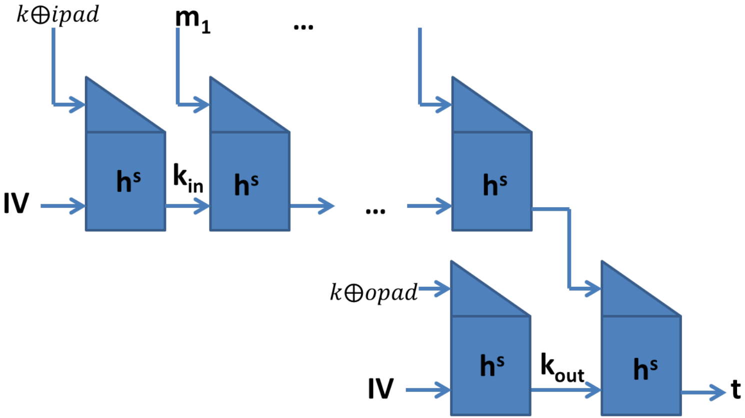
HMAC

- The idea is to build a secure MAC for arbitrary-length messages **directly** from a hash function.
- What about defining $\mathsf{Mac}_k(m) = H(k||m)$?

HMAC

- The idea is to build a secure MAC for arbitrary-length messages **directly** from a hash function.
- What about defining $\mathsf{Mac}_k(m) = H(k||m)$?
- It is NOT secure, why? (exercise)
- HMAC is a secure MAC that uses two layers of hashing.

HMAC



HMAC

Given a compression function h with input length $n + n'$. Let H be a hash function obtained from applying Merkle-Damgård transform on h. Let opad and ipad be two fixed constants of length n'. We define a MAC for arbitrary-length messages as follows:

- KeyGen(n): it runs the key generator of the hash function H to get a key s. It also chooses a uniform $k \in \{0,1\}^{n'}$. It outputs (s, k)
- Mac$(s, k, m \in \{0,1\}^*)$: it outputs

$$t \leftarrow H^s\big((k \oplus \text{opad})||H^s((k \oplus \text{ipad})||m)\big)$$

- Verify$(s, k, m \in \{0,1\}^*, t)$: outputs 1 iff

$$t \stackrel{?}{=} H^s\big((k \oplus \text{opad})||H^s((k \oplus \text{ipad})||m)\big)$$

Analysis of HMAC

- HMAC can be viewed as an instantiation of the hash-and-MAC technique.
- The use of keys in the inner computation allows for hash function with weaker assumptions to be used, namely hash functions that are *weakly* collision resistant (in this case, the adversary has access to a hash oracle to $H^s_{k_{in}}()$, where k_{in} is a secret value that replaces IV).
- The two keys in the inner and outer computations are treated as independent and uniform keys given that k is uniform.
- For efficiency reasons, they used ipad and opad to derive two keys from k.
- HMAC is very efficient and widely used in practice.

Outline

Generic attacks: The Birthday Attack

- Suppose there are q people in a room. What is the probability that two people have the same birthday?
- How many people do we need to have a probability larger than 1/2 ?

Generic attacks: The Birthday Attack

- Suppose there are q people in a room. What is the probability that two people have the same birthday?
- How many people do we need to have a probability larger than 1/2 ?
- Answer is **23**:

$$\Pr[\text{all distinct}] = 1 \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \ldots \cdot \frac{365-22}{365} < \frac{1}{2}$$

Generic attacks: The Birthday Attack

- Suppose you choose q elements randomly in a set of N elements. What is the probability that two elements are equal?
- How large should q be with respect to N to have a probability larger than 50% ?

Generic attacks: The Birthday Attack

- Suppose you choose q elements randomly in a set of N elements. What is the probability that two elements are equal?
- How large should q be with respect to N to have a probability larger than 50% ?
- Answer is $q = \Theta(\sqrt{N})$.
- Note: $f(x) = \Theta(g(x))$ means "f grows asymptotically *as fast as* g.

Generic attacks: The Birthday Attack

- Suppose you choose q elements randomly in a set of N elements. What is the probability that two elements are equal?
- How large should q be with respect to N to have a probability larger than 50% ?
- Answer is $q = \Theta(\sqrt{N})$.
- Note: $f(x) = \Theta(g(x))$ means "f grows asymptotically *as fast as* g.
- Let us try to solve it in a formal way...

The Birthday Problem

- Assume that you are throwing q balls to N bins. Let Coll denote the fact that two balls end up being in the same bin. We can show that

$$1 - e^{-q(q-1)/2N} \leq \Pr[\text{Coll}] \leq q(q-1)/2N$$

- Upper bound: Let Coll_i denote that the i-th ball falls into an already occupied bin, then $\Pr[\text{Coll}_i] \leq (i-1)/N$ as there are at most $i-1$ occupied bins.
- Now

$$\Pr[\text{Coll}] = \Pr[\bigvee_{i=1}^{q} \text{Coll}_i] \leq \sum_{i=1}^{q} \Pr[\text{Coll}_i] \leq 0/N + \cdots + (q-1)/N = \frac{q(q-1)}{2N}$$

The Birthday Problem

Lower bound: Let NoColl_i denote the event of not having any collision after throwing the i-th ball. we have

$$\Pr[\text{NoColl}_i|\text{NoColl}_{i-1}] = (N-(i-1))/N \quad (1)$$

which is the probability of not falling in any the the previous $i-1$ balls with $\Pr[\text{NoColl}_1] = 1$.
One can write

$$\Pr[\bar{\text{Coll}}] = \Pr[\text{NoColl}_q] \quad (2)$$

But

$$\Pr[\text{NoColl}_q] = \Pr[\text{NoColl}_q|\text{NoColl}_{q-1}].\Pr[\text{NoColl}_{q-1}]$$

Eventually, we will have

$$\Pr[\text{NoColl}_q] = \prod_{i=1}^{q-1}\Pr[\text{NoColl}_{i+1}|\text{NoColl}_i] \quad (3)$$

The Birthday Problem

From equations (1), (2) and (3)

$$\Pr[\bar{\text{Coll}}] = \prod_{i=1}^{q-1} \left(1 - \frac{i}{N}\right) \tag{4}$$

But we have $1 - x \leq e^{-x}$ for $x \leq 1$, which is the case for i/N. Thus,

$$\Pr[\bar{\text{Coll}}] \leq e^{-\sum_{i=1}^{q-1}(i/N)} = e^{-q(q-1)/2N}. \tag{5}$$

Therefore

$$\Pr[\text{Coll}] \geq 1 - e^{-q(q-1)/2N}$$

Hash Functions: the Birthday Attack

- How does the birthday attack apply to hash functions?
- We had a probability $\approx$ 1/2 when $q = \Theta(N^{1/2})$.
- If we have a hash function with output length ℓ, its range will be of size 2^ℓ.
- Therefore, if we take $q = \Theta(2^{\ell/2})$, the probability of finding a collision will be $\approx$ 1/2.
- In practice, to make finding collisions as difficult as exaustive search over 128-bit keys, you need a hash function with output length $\geq$ 256 bits.
- This is rather a necessary but not sufficient condition!
- This attack doesn't work for preimage and second preimage resistance!

A Better Birthday Attack

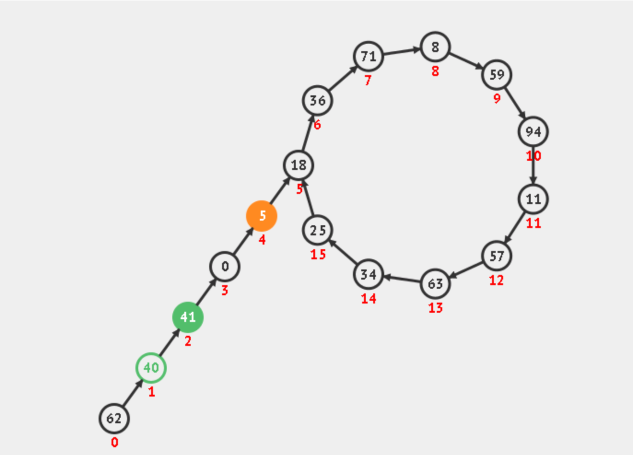
- The original birthday attack uses lots of memory storage. It has to store $\mathcal{O}(q) = \mathcal{O}\left(2^{\ell/2}\right)$ values.
- Managing storage for 2^{60} bytes is often more difficult that executing 2^{60} CPU instructions.
- Can we do better?

A Better Birthday Attack

- It is based on a cycle-finding algorithm of Floyd.
- We choose a random value x_0.
- We compute $x_i \leftarrow H(x_{i-1})$ and $x_{2i} \leftarrow H(H(x_{2(i-1)}))$ for $i = 1, 2, \ldots$, where $x_i = H^{(i)}(x_0)$.
- We compare x_i and x_{2i} after each iteration.
- If they are equal, then the collision happens somewhere in $x_0, \cdots, x_{2i-1}$.
- To find the collision, we try to find the smallest value of j for which $x_j = x_{j+i}$. The collision will then be (x_{j-1}, x_{j+i-1}).
- The algorithm has same time complexity and success probability as the general birthday attack, but only $\mathcal{O}(1)$ memory, namely, storage of two hashes in each iteration!

A better Birthday Attack

Floyd's cycle finding idea:`https://visualgo.net/bn/cyclefinding`



A Better Birthday Attack

We describe here a small-space birthday attack. We are given a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$, and we need to find x, x' s.t. $H(x) = H(x')$.

$x_0 \leftarrow_\$ \{0,1\}^{\ell+1}$

$x', x \leftarrow x_0$

for $i = 1, 2, \cdots$ **do**

  $x \leftarrow H(x) = H^{(i)}(x_0)$

  $x' \leftarrow H(H(x')) = H^{(2i)}(x_0)$

  if $x = x'$ **break**

$x' \leftarrow x, x \leftarrow x_0$

for $j = 1 \cdots, i$

  if $H(x) = H(x')$ **return** x, x'

  else $x \leftarrow H(x) = H^{(j)}(x_0)$

    $x' \leftarrow H(x') = H^{(i+j)}(x_0)$

Further Reading (1)

- Mihir Bellare and Phillip Rogaway.
  Random oracles are practical: A paradigm for designing efficient protocols.
  In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, 1993.
- Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
  Keccak sponge function family main document.
  Submission to NIST (Round 2), 3:30, 2009.

Further Reading (2)

- Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya.
  Merkle-damgård revisited: How to construct a hash function.
  In *Advances in Cryptology–CRYPTO 2005*, pages 430–448. Springer, 2005.
- Pierre Karpman, Thomas Peyrin, and Marc Stevens.
  Practical free-start collision attacks on 76-step sha-1.
  In *Advances in Cryptology–CRYPTO 2015*, pages 623–642. Springer, 2015.
- Neal Koblitz and Alfred J Menezes.
  The random oracle model: a twenty-year retrospective.
  Designs, Codes and Cryptography, pages 1–24, 2015.

Further Reading (3)

- Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone.
  Handbook of applied cryptography.
  CRC press, 1996.
- Marc Stevens.
  New collision attacks on sha-1 based on optimal joint local-collision analysis.
  In *Advances in Cryptology–EUROCRYPT 2013*, pages 245–261. Springer, 2013.