Lecture 9

Douglas Wikström
KTH Stockholm
dog@csc.kth.se

April 10, 2015

Hash Functions

Hash Function

A hash function maps arbitrarily long bit strings into bit strings of fixed length.

The output of a hash function should be "unpredictable".

Wish List

- Finding a pre-image of an output should be hard.
- Finding two inputs giving the same output should be hard.
- The output of the function should be “random”.

etc

Standardized Hash Functions

Despite that theory says it is impossible, in practice people simply live with **fixed** hash functions and use them as if they are randomly chosen functions.

SHA

- Secure Hash Algorithm (SHA-0,1, and the SHA-2 family) are hash functions standardized by NIST to be used in, e.g., signature schemes and random number generation.
- SHA-0 was **weak** and withdrawn by NIST. SHA-1 was **withdrawn** 2010. SHA-2 family is based on similar ideas but seems safe so far...
- All are **iterated** hash functions, starting from a basic **compression function**.

SHA-3

- NIST ran an open competition for the next hash function, named SHA-3. Several groups of famous researchers submitted proposals.
- Call for SHA-3 explicitly asked for “different” hash functions.
- It might be a good idea to read about SHA-1 for comparison.
- The competition ended October 2, 2012, and the hash function **Keccak was selected as the winner**.
- This was constructed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche,

Ensembles of Functions (1/3)

- Let $f : \{0,1\}^* \rightarrow \{0,1\}^*$ be a polynomial time computable function.

Ensembles of Functions (1/3)

- Let $f : \{0,1\}^* \rightarrow \{0,1\}^*$ be a polynomial time computable function.
- We can derive an ensemble $\{f_n\}_{n\in\mathbb{N}}$, with

$$f_n : \{0,1\}^n \rightarrow \{0,1\}^*$$

by setting $f_n(x) = f(x)$.

Ensembles of Functions (1/3)

- Let $f : \{0,1\}^* \rightarrow \{0,1\}^*$ be a polynomial time computable function.
- We can derive an ensemble $\{f_n\}_{n\in\mathbb{N}}$, with

  $$f_n : \{0,1\}^n \rightarrow \{0,1\}^*$$

  by setting $f_n(x) = f(x)$.
- Note that we may recover f from the ensemble by $f(x) = f_{|x|}(x)$.

Ensembles of Functions (1/3)

- Let $f : \{0,1\}^* \rightarrow \{0,1\}^*$ be a polynomial time computable function.
- We can derive an ensemble $\{f_n\}_{n\in\mathbb{N}}$, with

$$f_n : \{0,1\}^n \rightarrow \{0,1\}^*$$

  by setting $f_n(x) = f(x)$.
- Note that we may recover f from the ensemble by $f(x) = f_{|x|}(x)$.
- When convenient we give definitions for a function, but it can be turned into a definition for an ensemble.

Ensembles of Functions (2/3)

- Consider $F = \{f_n\}_{n\in\mathbb{N}}$, where f_n is itself an ensemble $\{f_{n,\alpha_n}\}_{\alpha_n\in\{0,1\}^n}$, with

$$f_{n,\alpha_n} : \{0,1\}^{l(n)} \to \{0,1\}^{l'(n)}$$

for some polynomials $l(n)$ and $l'(n)$.

Ensembles of Functions (2/3)

- Consider $F = \{f_n\}_{n\in\mathbb{N}}$, where f_n is itself an ensemble $\{f_{n,\alpha_n}\}_{\alpha_n\in\{0,1\}^n}$, with

$$f_{n,\alpha_n} : \{0,1\}^{l(n)} \rightarrow \{0,1\}^{l'(n)}$$

  for some polynomials $l(n)$ and $l'(n)$.

- Here n is the security parameter and α is a "key" that is chosen randomly.

Ensembles of Functions (2/3)

- Consider $F = \{f_n\}_{n\in\mathbb{N}}$, where f_n is itself an ensemble $\{f_{n,\alpha_n}\}_{\alpha_n\in\{0,1\}^n}$, with

$$f_{n,\alpha_n} : \{0,1\}^{l(n)} \rightarrow \{0,1\}^{l'(n)}$$

  for some polynomials $l(n)$ and $l'(n)$.

- Here n is the security parameter and α is a "key" that is chosen randomly.

- We may also view F as an ensemble $\{f_\alpha\}$, where $f_\alpha = \{f_{n,\alpha_n}\}_{n\in\mathbb{N}}$ and $\alpha = \{\alpha_n\}_{n\in\mathbb{N}}$.

Ensembles of Functions (3/3)

These conventions allow us to talk about what in everyday language is a "function" f in several convenient ways.

Now you can forget that and assume that everything works!

One-Wayness

Definition. A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is said to be **one-way**[1] if for every polynomial time algorithm A and a random x

$$\Pr[A(f(x)) = x' \wedge f(x') = f(x)] < \epsilon(n)$$

for a negligible function ϵ.

Normally f is computable in polynomial time in its input size.

[1] "Enkelriktad" på svenska **inte** "envägg".

Second Pre-Image Resistance

Definition. A function $h : \{0,1\}^* \rightarrow \{0,1\}^*$ is said to be **second pre-image resistant** if for every polynomial time algorithm A and a random x

$$\Pr[A(x) = x' \wedge x' \neq x \wedge f(x') = f(x)] < \epsilon(n)$$

for a negligible function ϵ.

Note that A is given not only the output of f, but also the **input** x, but it must find a **second** pre-image.

Collision Resistance

Definition. Let $f = \{f_\alpha\}_\alpha$ be an ensemble of functions. The "function" f is said to be **collision resistant** if for every polynomial time algorithm A and randomly chosen α

$$\Pr[A(\alpha) = (x, x') \wedge x \neq x' \wedge f_\alpha(x') = f_\alpha(x)] < \epsilon(n)$$

for a negligible function ϵ.

Collision Resistance

Definition. Let $f = \{f_\alpha\}_\alpha$ be an ensemble of functions. The "function" f is said to be **collision resistant** if for every polynomial time algorithm A and randomly chosen α

$$\Pr[A(\alpha) = (x, x') \wedge x \neq x' \wedge f_\alpha(x') = f_\alpha(x)] < \epsilon(n)$$

for a negligible function ϵ.

An algorithm that gets a small "advice string" for each security parameter can easily hardcode a collision for a fixed function f, which explains the random index α.

Relations for Compressing Hash Functions

- If a function is not pre-image resistant, then it is not collision-resistant.

Relations for Compressing Hash Functions

- If a function is not pre-image resistant, then it is not collision-resistant.
  1. Pick random x.
  2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
  3. Output x' and x.

Relations for Compressing Hash Functions

- If a function is not pre-image resistant, then it is not collision-resistant.
  1. Pick random x.
  2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
  3. Output x' and x.

- If a function is not one-way, then it is not second pre-image resistant.

Relations for Compressing Hash Functions

- If a function is not pre-image resistant, then it is not collision-resistant.
  1. Pick random x.
  2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
  3. Output x' and x.

- If a function is not one-way, then it is not second pre-image resistant.
  1. Given random x, compute $y = f(x)$.
  2. Request pre-image x' of y.
  3. Repeat until $x' \neq x$, and output x'.

Random Oracles

Random Oracle As Hash Function

A random oracle is simply a randomly chosen function with appropriate domain and range.

A random oracle is the **perfect** hash function. Every input is mapped **independently** and **uniformly** in the range.

Let us consider how a random oracle behaves with respect to our notions of security of hash functions.

Pre-Image of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a pre-image, i.e., it queries the random oracle on its output.

Theorem. Let $H : X \to Y$ be a randomly chosen function and let $x \in X$ be randomly chosen. Then for every algorithm A making q oracle queries

$$\Pr[A^{H(\cdot)}(H(x)) = x' \wedge H(x) = H(x')] \leq 1 - \left(1 - \frac{1}{|Y|}\right)^q .$$

Pre-Image of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a pre-image, i.e., it queries the random oracle on its output.

Theorem. Let $H : X \rightarrow Y$ be a randomly chosen function and let $x \in X$ be randomly chosen. Then for every algorithm A making q oracle queries

$$\Pr[A^{H(\cdot)}(H(x)) = x' \wedge H(x) = H(x')] \leq 1 - \left(1 - \frac{1}{|Y|}\right)^q \quad .$$

Proof. Each query x' satisfies $H(x') \neq H(x)$ independently with probability $1 - \frac{1}{|Y|}$.

Second Pre-Image of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a second pre-image, i.e., it queries the random oracle on the input and its output.

Theorem. Let $H : X \to Y$ be a randomly chosen function and let $x \in X$ be randomly chosen. Then for every such algorithm A making q oracle queries

$$\Pr[A^{H(\cdot)}(x) = x' \wedge x \neq x' \wedge H(x) = H(x')] \leq 1 - \left(1 - \frac{1}{|Y|}\right)^{q-1} .$$

Second Pre-Image of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a second pre-image, i.e., it queries the random oracle on the input and its output.

Theorem. Let $H : X \to Y$ be a randomly chosen function and let $x \in X$ be randomly chosen. Then for every such algorithm A making q oracle queries

$$\Pr[A^{H(\cdot)}(x) = x' \wedge x \neq x' \wedge H(x) = H(x')] \leq 1 - \left(1 - \frac{1}{|Y|}\right)^{q-1} .$$

Proof. Same as pre-image case, except we must waste one query on the input value to get the target in Y.

Collision Resistance of Random Oracles

We assume with little loss that an adversary always “knows” if it has found a collision, i.e., it queries the random oracle on its outputs.

Theorem. Let $H : X \to Y$ be a randomly chosen function. Then for every such algorithm A making q oracle queries

$$\Pr[A^{H(\cdot)} = (x, x') \wedge x \neq x' \wedge H(x) = H(x')] \leq 1 - \prod_{i=1}^{q-1} \left(1 - \frac{i}{|Y|}\right) \leq \frac{q(q-1)}{2|Y|} \enspace .$$

Collision Resistance of Random Oracles

We assume with little loss that an adversary always "knows" if it has found a collision, i.e., it queries the random oracle on its outputs.

Theorem. Let $H : X \to Y$ be a randomly chosen function. Then for every such algorithm A making q oracle queries

$$\begin{aligned}\Pr[A^{H(\cdot)} = (x, x') \wedge x \neq x' \wedge H(x) = H(x')] &\leq 1 - \prod_{i=1}^{q-1} \left(1 - \frac{i}{|Y|}\right) \\ &\leq \frac{q(q-1)}{2|Y|} \enspace .\end{aligned}$$

Proof. $1 - \frac{i-1}{|Y|}$ bounds the probability that the ith query does not give a collision for any of the $i - 1$ previous queries, conditioned on no previous collision.

Iterated Hash Functions

Merkle-Damgård (1/3)

Suppose that we are given a collision resistant hash function

$$f : \{0,1\}^{n+t} \rightarrow \{0,1\}^n \ .$$

How can we construct a collision resistant hash function

$$h : \{0,1\}^* \rightarrow \{0,1\}^n$$

mapping any length inputs?

Merkle-Damgård (2/3)

Construction.

1. Let $x = (x_1, \ldots, x_k)$ with $|x_i| = t$ and $0 < |x_k| \leq t$.
2. Let x_{k+1} be the total number of bits in x.
3. Pad x_k with zeros until it has length t.
4. $y_0 = 0^n$, $y_i = f(y_{i-1}, x_i)$ for $i = 1, \ldots, k+1$.
5. Output y_{k+1}

Here the total number of bits is bounded by $2^t - 1$, but this can be relaxed.

Merkle-Damgård (3/3)

Suppose A finds collisions in Merkle-Damgård.

- If the number of bits differ in a collision, then we can derive a collision from the last invocation of f.
- If not, then we move backwards until we get a collision. Since both inputs have the same length, we are guaranteed to find a collision.

Universal Hash Functions

Universal Hash Function

Definition. An ensemble $f = \{f_\alpha\}$ of hash functions $f_\alpha : X \rightarrow Y$ is (strongly) 2-universal if for every $x, x' \in X$ and $y, y' \in Y$ with $x \neq x'$ and a random α

$$\Pr[f_\alpha(x) = y \wedge f_\alpha(x') = y'] = \frac{1}{|Y|^2} \ .$$

Universal Hash Function

Definition. An ensemble $f = \{f_\alpha\}$ of hash functions $f_\alpha : X \to Y$ is (strongly) 2-universal if for every $x, x' \in X$ and $y, y' \in Y$ with $x \neq x'$ and a random α

$$\Pr[f_\alpha(x) = y \wedge f_\alpha(x') = y'] = \frac{1}{|Y|^2} \ .$$

I.e., for any $x' \neq x$, the outputs $f_\alpha(x)$ and $f_\alpha(x')$ are uniformly and independently distributed.

In particular x and x' are both mapped to the same value with probability $1/|Y|$.

Example

Example. The function $f : \mathbb{Z}_p \to \mathbb{Z}_p$ for prime p defined by

$$f(z) = az + b \bmod p$$

is strongly 2-universal.

Proof. Let $x, x', y, y' \in \mathbb{Z}_p$ with $x \neq x'$. Then

$$\begin{pmatrix} x & 1 \\ x' & 1 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} y \\ y' \end{pmatrix}$$

has a unique solution. Random (a, b) satisfies this solution with probability $\frac{1}{p^2}$.

Universal Hash Function

Universal hash functions are **not** one-way or collision resistant!

Message Authentication Code

- Message Authentication Codes (MACs) are used to ensure integrity and authenticity of messages.

Message Authentication Code

- Message Authentication Codes (MACs) are used to ensure integrity and authenticity of messages.
- Scenario:
  1. Alice and Bob share a common key k.
  2. Alice computes an authentication tag $\alpha = \mathsf{MAC}_\mathsf{k}(m)$ and sends (m, α) to Bob.
  3. Bob receives (m', α') from Alice, but before accepting m' as coming from Alice, Bob checks that $\mathsf{MAC}_\mathsf{k}(m') = \alpha'$.

Security of a MAC

Definition. A message authentication code MAC is secure if for a random key k and every polynomial time algorithm A,

$$\Pr[A^{\mathsf{MAC}_\mathsf{k}(\cdot)} = (m, \alpha) \wedge \mathsf{MAC}_\mathsf{k}(m) = \alpha \wedge \forall i : m \neq m_i]$$

is negligible, where m_i is the ith query to the oracle $\mathsf{MAC}_\mathsf{k}(\cdot)$.

Random Oracle As MAC

- Suppose that $H : \{0,1\}^* \rightarrow \{0,1\}^n$ is a random oracle.
- Then we can construct a MAC as $\mathrm{MAC}_{\mathsf{k}}(m) = H(\mathsf{k}, m)$.

Could we plug in an iterated hash function in place of the random oracle?

HMAC

- Let $H : \{0,1\}^* \rightarrow \{0,1\}^n$ be a "cryptographic hashfunction", e.g., SHA-256.
- $\mathrm{HMAC}_{\mathrm{k}_1,\mathrm{k}_2}(x) = H\big(\mathrm{k}_2 \| H(\mathrm{k}_1 \| x)\big)$
- This is provably secure under the assumption that
  - $H(\mathrm{k}_1 \| \cdot)$ is unknown-key collision resistant, and
  - $H(\mathrm{k}_2 \| \cdot)$ is a secure MAC.

CBC-MAC

Let E be a secure block-cipher, and $x = (x_1, \ldots, x_t)$ an input. The MAC-key is simply the block-cipher key.

1. $y_0 = 000\ldots0$
2. For $i = 1, \ldots, t$, $y_i = \mathsf{E_k}(y_{i-1} \oplus x_i)$
3. Return y_t.

Is this secure?

Universal Hashfunction As MAC

Theorem. A t-universal hashfunction f_α for a randomly chosen secret α is an **unconditionally secure** MAC, provided that the number queries is smaller than t.