Security Notions, Random Oracles, Iterated Hashfunctions, Universal Hashfunctions

Douglas Wikström
KTH Stockholm
dog@csc.kth.se

February 23

- Security Notions
- Random Oracles
- Iterated Hashfunctions
- Universal Hashfunctions

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.
- For each polynomial $l(n)$ we can derive an ensemble $f = \{f_n\}_{n \in \mathbb{N}}$, with

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

for some polynomial $l'(n)$, 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.
- For each polynomial $l(n)$ we can derive an ensemble $f = \{f_n\}_{n \in \mathbb{N}}$, with

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

  for some polynomial $l'(n)$, by setting $f_n(x) = f(x)$.
- When convenient we give definitions for a function, but it can be turned into a definition for an ensemble.

Ensembles of Functions (1/3)

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

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

  for some polynomial $l'(n)$, by setting $f_n(x) = f(x)$.
- When convenient we give definitions for a function, but it can be turned into a definition for an ensemble.
- When $l(n) > l'(n)$ we say that f is compressing.

Ensembles of Functions (2/3)

- We let f be an ensemble $\{f_n\}_{n\in\mathbb{N}}$, where f_n is itself an ensemble $\{f_{n,\alpha_n}\}_{\alpha_n\in\{0,1\}^n}$, with

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

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)

- We let f be an ensemble $\{f_n\}_{n\in\mathbb{N}}$, 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 $f = \{f_\alpha\}$, where $f_\alpha = \{f_{n,\alpha_n}\}_{n\in\mathbb{N}}$ and $\alpha = \{\alpha_n\}$.

Ensembles of Functions (2/3)

- We let f be an ensemble $\{f_n\}_{n\in\mathbb{N}}$, 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 $f = \{f_\alpha\}$, where $f_\alpha = \{f_{n,\alpha_n}\}_{n\in\mathbb{N}}$ and $\alpha = \{\alpha_n\}$.

- These conventions allow us to talk about a "random function" f in several convenient ways.

Ensembles of Functions (3/3)

Now you can forget that and assume that everything works!

One-Wayness

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

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

for a negligible function ϵ.

Normally f is computable in polynomial time in its input.

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

One-Wayness

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

$$\Pr[A(1^n, f(x)) = x' \wedge f(x') = f(x)] < \epsilon(n)$$

for a negligible function ϵ.

Normally f is computable in polynomial time in its input.

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

Second Pre-Image Resistance

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

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

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 **another** preimage.

Collision Resistance

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

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

for a negligible function ϵ.

Collision Resistance

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

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

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 Between Notions

- Collision resistance implies pre-image resistance.

Relations Between Notions

- Collision resistance implies pre-image resistance.
  1. Pick random x.
  2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
  3. Output x' and x.

Relations Between Notions

- Collision resistance implies pre-image resistance.
  1. Pick random x.
  2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
  3. Output x' and x.

- Second pre-image resistance implies one-wayness.

Relations Between Notions

- Collision resistance implies pre-image resistance.
  1. Pick random x.
  2. Request second pre-image $x' \neq x$ with $f(x') = f(x)$.
  3. Output x' and x.

- Second pre-image resistance implies one-wayness.
  1. Given random x, compute $y = f(x)$.
  2. Request preimage x' of y.
  3. Repeat until $x' \neq x$, and output x'.

Random Oracle As Hashfunction

A random oracle is the **perfect** hashfunction. 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 hashfunctions.

Preimage of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a preimage, 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 such algorithm A making q oracle queries

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

Preimage of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a preimage, 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 such algorithm A making q oracle queries

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

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

Second Preimage of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a second preimage, 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

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

Second Preimage of Random Oracle

We assume with little loss that an adversary always "knows" if it has found a second preimage, 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

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

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

Random Oracle Viewed As Hashfunction (3/3)

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

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

Random Oracle Viewed As Hashfunction (3/3)

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

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

Proof. The $\left(1 - \frac{i-1}{|X|}\right)$ is the probability that the ith query does not give a collision for any of the $i - 1$ previous queries.

Merkle-Damgård (1/3)

Suppose that we are given a collision resistant hashfunction

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

How can we construct a collision resistant hashfunction

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

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. $y_0 = 0^n$, $y_i = f(y_{i-1}, x_i)$ for $i = 1, \ldots, k+1$.
4. 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 we can derive a collision from the last invocation of f.
- If not we move backwards until we get a collision. Since both inputs have the same length, we are guaranteed to find a collision.

Pedersen's Compression Function

- Let G_q be a cyclic group of order q in which the discrete logarithm assumption holds.

Pedersen's Compression Function

- Let G_q be a cyclic group of order q in which the discrete logarithm assumption holds.
- Let g and h be random generators in G_q.

Pedersen's Compression Function

- Let G_q be a cyclic group of order q in which the discrete logarithm assumption holds.
- Let g and h be random generators in G_q.
- Then the function $f_{g,h} : \mathbb{Z}_q \times \mathbb{Z}_q \rightarrow G_q$ defined by

$$f_{g,h}(x, y) = g^x h^y$$

is collision resistant (and compressing).

SHA

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

SHA-3

Call for SHA-3 explicitly asked for "different" hashfunctions. You will tell us all about the candidates!

It might be a good idea to read about SHA-1 for comparison...

Universal Hashfunction

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

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

Universal Hashfunction

Definition. An ensemble $f = \{f_\alpha\}$ of hashfunctions $f_\alpha : X \to Y$ 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 y 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 Hashfunction

Universal hashfunctions are **not** one-way or collision resistant!

So, why are they useful to us in cryptography?