Announcements
Some Project Ideas

Read & Write about something not covered in class
- **Constructions:** e.g., CCA secure PKE schemes, lattice-based PKE, more block-cipher modes, ...
- **Concepts:** e.g., Key management, Double-Ratcheting, Searchable Encryption, Onion Routing/Mix-Nets, Homomorphic Encryption, ...
- **Proofs:** e.g., Goldreich-Levin predicate, Fujisaki-Okamoto, security of TLS,...

Implementation project

Make something
- Slow and secure crypto (e.g., SKE and/or Digital Signatures from OWP, full-domain CRHF from DL,...)
- Higher-level applications (e.g., “simple-TLS”, Off-the-record messaging, things you can do with a block-cipher...)
- A library with a cleaner API for encryption/authentication

Break something
- e.g., use a constraint-solver to break (broken) block-ciphers
Hash Functions

Lecture 14
Flavours of collision resistance
A Tale of Two Boxes

The bulk of today’s applied cryptography works with two magic boxes

- Block Ciphers
- Hash Functions

Block Ciphers: Best modeled as (strong) Pseudorandom Permutations, with inversion trapdoors

- Often more than needed (e.g. SKE needs only PRF)

Hash Functions:

- Some times modelled as Random Oracles!
- Use at your own risk! No guarantees in the standard model.

Today: understanding security requirements on hash functions
Hash Functions

“Randomised” mapping of inputs to shorter hash-values

Hash functions are useful in various places

In data-structures: for efficiency

Intuition: hashing removes worst-case effects

In cryptography: for “integrity”

Primary use: Domain extension (compress long inputs, and feed them into boxes that can take only short inputs)

Typical security requirement: “collision resistance”

Different flavours: some imply one-wayness

Also sometimes: some kind of unpredictability
Hash Function Family

- Hash function \( h : \{0,1\}^{n(k)} \rightarrow \{0,1\}^{t(k)} \)
- Compresses
- A family
  - Alternately, takes two inputs, the index of the member of the family, and the real input
- Efficient sampling and evaluation
- Idea: when the hash function is randomly chosen, "behaves randomly"
- Main goal: to "avoid collisions". Will see several variants of the problem

<table>
<thead>
<tr>
<th>( x )</th>
<th>( h_1(x) )</th>
<th>( h_2(x) )</th>
<th>( h_3(x) )</th>
<th>( h_4(x) )</th>
<th>( h_N(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Hash Functions in Crypto Practice

- A single fixed function
  - e.g. SHA-3, SHA-256, SHA-1, MD5, MD4
  - Not a family ("unkeyed")
  - (And no security parameter knob)

- Not collision-resistant under any of the following definitions

- Alternately, could be considered as having already been randomly chosen from a family (and security parameter fixed too)

- Usually involves hand-picked values (e.g. "I.V." or "round constants") built into the standard
Degrees of Collision-Resistance

If for all PPT $A$, $\Pr[x \neq y \text{ and } h(x) = h(y)]$ is negligible in the following experiment:

- $A \rightarrow (x, y); h \leftarrow \$ : Combinatorial Hash Functions (even non-PPT $A$)
- $A \rightarrow x; h \leftarrow \$; A(h) \rightarrow y : Universal One-Way Hash Functions
- $h \leftarrow \$; A(h) \rightarrow (x, y) : Collision-Resistant Hash Functions

CRHF the strongest; UOWHF still powerful (will be enough for digital signatures)

Useful variants: $A$ gets only oracle access to $h(\cdot)$ (weaker). Or, $A$ gets any coins used for sampling $h$ (stronger).
Degrees of Collision-Resistance

Variants of CRHF/UOWHF where \( x \) is random

\[ h \leftarrow \mathcal{H}; \ x \leftarrow \mathcal{X}; \ A(h, h(x)) \rightarrow y \quad (y=x \text{ allowed}) \]

Pre-image collision resistance if \( h(x) = h(y) \) w.n.p

i.e., \( f(h, x) := (h, h(x)) \) is a OWF (and \( h \) compresses)

\[ h \leftarrow \mathcal{H}; \ x \leftarrow \mathcal{X}; \ A(h, x) \rightarrow y \quad (y \neq x) \]

Second Pre-image collision resistance if \( h(x) = h(y) \) w.n.p

Incomparable (neither implies the other) \([\text{Exercise}]\)

CRHF implies second pre-image collision resistance and, if compressing, then pre-image collision resistance \([\text{Exercise}]\)
Hash Length

- If range of the hash function is too small, not collision-resistant
- If range poly(k)-size (i.e. hash is logarithmically long), then non-negligible probability that two random x, y provide collision
- In practice interested in minimising the hash length (for efficiency)
- Generic attack on a CRHF: birthday attack
  - Look for a collision in a set of random inputs (needs only oracle access to the hash function)
  - Expected size of the set before collision: $O(\sqrt{|\text{range}|})$
- Birthday attack effectively halves the security (hash length) of a CRHF compared to a generic attack on UOWHF
Universal Hashing

Combinatorial HF: $A \rightarrow (x,y); h \leftarrow \mathcal{H}. h(x)=h(y)$ w.n.p

Even better: 2-Universal Hash Functions

“Uniform” and “Pairwise-independent”

$\forall x,z \Pr_{h \leftarrow \mathcal{H}} [ h(x)=z ] = 1/|Z| \text{ (where } h: X \rightarrow Z)$

$\forall x \neq y, w, z \Pr_{h \leftarrow \mathcal{H}} [ h(x)=w, h(y)=z ] = \Pr_{h \leftarrow \mathcal{H}} [ h(x)=w ] \cdot \Pr_{h \leftarrow \mathcal{H}} [ h(y)=z ]$

$\Rightarrow \forall x \neq y \Pr_{h \leftarrow \mathcal{H}} [ h(x)=h(y) ] = 1/|Z|$

k-Universal:

$\forall x_1 \ldots x_k \text{ (distinct), } z_1 \ldots z_k, \Pr_{h \leftarrow \mathcal{H}} [ \forall i h(x_i)=z_i ] = 1/|Z|^k$

Inefficient example: $\mathcal{H}$ set of all functions from $X$ to $Z$

But we will need all $h \in \mathcal{H}$ to be succinctly described and efficiently evaluable
Universal Hashing

Combinatorial HF: \( A \rightarrow (x, y); \ h \leftarrow \mathcal{U}. \ h(x) = h(y) \) w.n.p

Even better: \( 2 \)-Universal Hash Functions

"Uniform" and "Pairwise-independent"

\[ \forall x \neq y, w, z \ \Pr_{h \leftarrow \mathcal{U}} [ h(x) = w, h(y) = z ] = 1/|Z|^2 \]

\[ \Rightarrow \forall x \neq y \ \Pr_{h \leftarrow \mathcal{U}} [ h(x) = h(y) ] = 1/|Z| \]

e.g. \( h_{a,b}(x) = ax + b \) (in a finite field, \( X = \mathbb{Z} \))

Uniform

\[ \Pr_{a,b} [ ax + b = z ] = \Pr_{a,b} [ b = z - ax ] = 1/|Z| \]

\[ \Pr_{a,b} [ ax + b = w, ay + b = z ] = ? \text{ Exactly one } (a, b) \text{ satisfying the two equations (for } x \neq y) \]

\[ \Pr_{a,b} [ ax + b = w, ay + b = z ] = 1/|Z|^2 \]

But does not compress!

Negligible collision-probability if super-polynomial-sized range
Universal Hashing

Combinatorial HF: \( A \to (x,y); h \leftarrow \mathcal{H}. \ h(x)=h(y) \) w.n.p

Even better: 2-Universal Hash Functions

“Uniform” and “Pairwise-independent”

\[ \forall x \neq y, w, z \ Pr_h [ h(x)=w, h(y)=z ] = 1/|Z|^2 \]

\[ \Rightarrow \forall x \neq y \ Pr_h [ h(x)=h(y) ] = 1/|Z| \]

e.g. Chop(h(x)) where

h from a (possibly non-compressing) 2-universal HF

Chop a t-to-1 map from Z to Z'

e.g. with |Z|=2^k, removing last bit gives a 2-to-1 mapping

\[ Pr_h [ Chop(h(x)) = w, Chop(h(y)) = z ] = Pr_h [ h(x) = w0 \text{ or } w1, h(y) = z0 \text{ or } z1 ] = 4/|Z|^2 = 1/|Z'|^2 \]
Today

Combinatorial hash functions, UOWHF and CRHF

(And weaker variants of CRHF: pre-image collision resistance and second-pre-image collision resistance)

Collision-resistant combinatorial HF from 2-Universal Hash Functions

Next:

UOWHF from 2-Universal HF and OWP (possible from OWF)

A candidate CRHF construction

Domain extension