Hash Functions (ctd.)

Lecture 15
Main syntactic feature: Variable input length to fixed length output

Primary requirement: collision-resistance

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

\[ A \rightarrow (x, y); \quad h \leftarrow \mathcal{H} : \text{Combinatorial Hash Functions} \]

\[ A \rightarrow x; \quad h \leftarrow \mathcal{H}; \quad A(h) \rightarrow y : \text{Universal One-Way Hash Functions} \]

\[ h \leftarrow \mathcal{H}; \quad A(h) \rightarrow (x, y) : \text{Collision-Resistant Hash Functions} \]

\[ h \leftarrow \mathcal{H}; \quad A^h \rightarrow (x, y) : \text{Weak Collision-Resistant Hash Functions} \]

\[ h \leftarrow \mathcal{H}; \quad x \leftarrow X; \quad A(h, h(x)) \rightarrow y : \text{One-Way Hash Functions} \quad (x = y \text{ OK}) \]

\[ h \leftarrow \mathcal{H}; \quad x \leftarrow X; \quad A(h, x) \rightarrow y : \text{SPR Hash Functions} \]

Also often required: “unpredictability”

Already saw: a 2-UHF (chop(ax+b) over a field)

Today: UOWHF and CRHF constructions. Domain Extension.
Universal One-Way HF: $A \rightarrow x; \ h \leftarrow \mathcal{U}; \ A(h) \rightarrow y. \ h(x) = h(y) \ \text{w.n.p}$

Since the hash function is compressing, then there will be collisions. So a computationally unbounded adversary can win this game!

Need to rely on computational hardness

UOWHF can be constructed from OWF

Much easier to see $\text{OWP} \Rightarrow \text{UOWHF}$
UOWHF from OWP

\[ F_h(x) = h(f(x)), \text{ where } f \text{ is a OWP and } h \text{ from a UHF family} \]

\[ \text{s.t. } h \text{ compresses by a bit (i.e., is a 2-to-1 map), and} \]

\[ \text{for all } z, z', w, \text{ can efficiently solve for } h \text{ s.t. } h(z) = h(z') = w \]

Is a UOWHF: can choose \( h \) to force UOWHF adversary to invert \( f \)

\[
\text{BreakOWP}(z) \begin{cases} 
\text{Get } x \leftarrow A; \text{ Sample random } w; \text{ Solve } h \text{ s.t. } h(z) = h(f(x)) = w; \\
\text{Give } h \text{ to } A; \text{ Get } y \leftarrow A \text{ and output } y; 
\end{cases}
\]

Only collision \(( y \neq x \text{ s.t. } F_h(x) = F_h(y) ) \) is \( y = f^{-1}(z) \)
**UOWHF from OWP**

- $F_h(x) = h(f(x))$, where $f$ is a OWP and $h$ from a UHF family

- s.t. $h$ compresses by a bit (i.e., is a 2-to-1 map), and

- for all $z, z', w$, can efficiently solve for $h$ s.t. $h(z) = h(z') = w$

Is a UOWHF: can choose $h$ to force UOWHF adversary to invert $f$

```
BreakOWP(z) { Get x ← A; Sample random w; Solve h s.t. h(z) = h(f(x)) = w; Give h to A; Get y ← A and output y; }
```

- Only collision ($y ≠ x$ s.t. $F_h(x) = F_h(y)$) is $y = f^{-1}(z)$

- BreakOWP is efficient as $h$ can be efficiently solved ✓

- BreakOWP has same advantage as $A$ has against UOWHF? Yes, if $h$ is uniform (independent of $x$) [Why?]

- $h$ uniform because $z, w$ picked uniformly ✓
Collision-Resistant HF: $h \leftarrow \#; A(h) \rightarrow (x, y)$. $h(x) = h(y)$ w.n.p

Not known to be possible from OWF/OWP alone

“Impossibility” (blackbox-separation) known

Possible from “claw-free pair of permutations”

In turn from hardness of discrete-log, factoring, and from lattice-based assumptions

Also from “homomorphic one-way permutations”, and from homomorphic encryptions

All candidates use mathematical operations that are considered computationally expensive
CRHF

- CRHF from discrete log assumption:
  - Suppose $G$ a group of prime order $q$, where DL is considered hard (e.g. $\mathbb{QR}_{p^*}$ for $p=2q+1$ a safe prime)
  - $h_{g_1,g_2}(x_1,x_2) = g_1^{x_1}g_2^{x_2}$ (in $G$) where $g_1$, $g_2 \neq 1$ (hence generators)
  - A collision: $(x_1,x_2) \neq (y_1,y_2)$ s.t. $h_{g_1,g_2}(x_1,x_2)= h_{g_1,g_2}(y_1,y_2)$
    - Collision $\Rightarrow x_1 \neq y_1$ and $x_2 \neq y_2$ [Why?] 
  - Then $g_2 = g_1^{(x_1-y_1)/(x_2-y_2)}$ (exponents in $\mathbb{Z}_{q^*}$)
    - i.e., w.r.t. a random base $g_1$, can compute DL of a random element $g_2$. Breaks DL!
  - Hash halves the size of the input
**Domain Extension**

- **Full-domain hash**: hash arbitrarily long strings to a single hash value
  - So far, UOWHF/CRHF which have a fixed domain
  - First, simpler goal: extend to a larger, fixed domain

Assume we are given a hash function from two blocks to one block (a block being, say, k bits)

- What if we can compress by only one bit (e.g., our UOWHF construction)?

  Can just apply repeatedly to compress by \( t \) bits
Domain Extension

Can compose hash functions more efficiently, using a “Merkle tree”

Suppose basic hash from \(\{0,1\}^{2^k}\) to \(\{0,1\}^{2k}\). A hash function from \(\{0,1\}^{8k}\) to \(\{0,1\}^k\) using a tree of depth 3

If basic hash from \(\{0,1\}^{2^k}\) to \(\{0,1\}^{2k-1}\), first construct new basic hash from \(\{0,1\}^{2k}\) to \(\{0,1\}^k\), by repeated hashing

Any tree can be used, with consistent I/O sizes

Independent hashes or same hash?

Depends!
For CRHF, **same basic hash** used throughout the Merkle tree. Hash description same as for a single basic hash.

If a collision \((x_1...x_n), (y_1...y_n)\) over all, then some collision \((x',y')\) for basic hash.

Consider moving a “frontline” from bottom to top. Look for equality on this front.

Collision at some step (different values on \(i^{th}\) front, same on \(i+1^{st}\)); gives a collision for basic hash.

\(A^*(h):\) run \(A(h)\) to get \((x_1...x_n), (y_1...y_n)\). Move frontline to find \((x',y')\)
Domain Extension for UOWHF

- For UOWHF, can’t use same basic hash throughout!
- $A^*$ has to output an $x'$ on getting $(x_1...x_n)$ from $A$, before getting $h$
  - Can guess a random node (i.e., random pair of frontlines) where collision occurs, but if not a leaf, can’t compute $x'$ until $h$ is fixed!
- Solution: a different $h$ for each level of the tree (i.e., no ancestor/successor has same $h$)
  - To compute $x'$: Get $(x_1...x_n)$ from $A$. Then pick a random node (say at level $i$), pick $h_j$ for levels below $i$, and compute input to the node; let this be $x'$.
  - On getting $h$, plug it in as $h_i$, pick $h_j$ for remaining levels; give $h$'s to $A$ and get $(y_1...y_n)$; compute $y'$ and output it.
UOWHF vs. CRHF

- UOWHF has a weaker guarantee than CRHF
- UOWHF can be built based on OWF (we saw based on OWP), whereas CRHF “needs stronger assumptions”
  - But “usual” OWF candidates suffice for CRHF too (we saw construction based on discrete-log)
- Domain extension of CRHF is simpler, with no blow-up in the description size. For UOWHF description increases logarithmically in the input size
- UOWHF theoretically important (based on simpler assumptions, good if paranoid), but CRHF can substitute for it
- Current practice: much less paranoid; faith on efficient, ad hoc (and unkeyed) constructions (though increasingly under attack)
Domain Extension

**Full-domain hash:** hash arbitrarily long strings to a single hash value

- Merkle-Tree construction extends the domain to any fixed input length

- Hash the message length (number of blocks) along with the original hash

- Collision in the new hash function gives either collision at the top level, or if not, collision in the original Merkle tree and for the same message length

![Diagram of Merkle Tree with hash length (|m|) at the top](image)
Hash Functions in Practice

A single function, not a family (e.g. SHA-3, SHA-256, MD4, MD5)

Often from a fixed input-length compression function

Merkle-Damgård iterated hash function, $\text{MD}^f$:

If $f$ is collision resistant then so is $\text{MD}^f$ (for any IV)

If $f$ modelled as a Random Oracle, $\text{MD}^f$ is a “public-use RO.”
If $f$ modelled as an “Ideal Cipher,” $\text{MD}^f$ is “pre-image aware.”