6.857 Computer and Network Security September 18, 2001

Lecture Notes 4 : Frogs/Voting and Hash Codes

Lecturer: Ron Rivest *Scribe: Gilliland/Li/Lustbader/Wagner*

Outline

- Frogs
- Collision-Resistant Hash Functions
- Digital Signatures

1 Frogs - Comments

See the 6.857 Web page for a copy of the presentation on Voting/Frogs. Each ballot is recorded on an object called a "frog." This is not an acronym; it was chosen to be a neutral term with convenient clip-art for slides...

- A frog cannot simply be copied over and over because a randomized procedure is used for each frog. Two exact copies can be detected.
- Frogs for different voters should look different even if the voters vote the same.
- The signature schemes would have to be certified – they are part of the critical voting process.
- Vote generation hardware is important for security, but not critical. The casting hardware *is* critical.
- The voter may only approach the vote casting hardware once.

2 Hash Functions

Hash functions generally have the function signature:

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

That is, a hash function transforms an input of any length into a constant-length string.

For the MD5 algorithm, $n = 128$, and for SHA-1, $n = 160$. Cryptographic hash functions are easy to compute in the forward direction. Other properties such as collision resistance and one-wayness deserve further discussion.

2.1 Hash Function Properties

One-way

A function h is one-way (or pre-image resistant) if no efficient adversary program $\mathcal{A}$ exists which can invert h on more than a negligible fraction of its outputs. More formally, we say that a function $h : \{0,1\}^* \rightarrow \{0,1\}^n$ is one-way if $Prob[h(\mathcal{A}(y)) = y]$ is negligible where $y \in \{0,1\}^n$ and $y = h(x)$ for some $x \in \{0,1\}^*$.

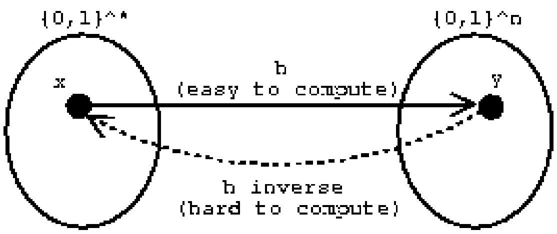

Figure 1: A one-way function is hard to invert.

Collision Resistance

A function is collision resistant (also known as strong collision resistance) if it is infeasible to find any two distinct inputs x and x' such that $h(x) = h(x')$. While we know such x, x' pairs must exist by the pigeonhole principle, it may be computationally difficult to find these pairs.

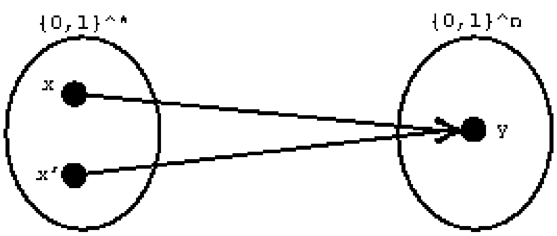

Figure 2: In [strong] collision resistance, it is difficult to find any pair of inputs which produce the same output.

No one has ever found a collision for MD5 or SHA-1. Collisions have been found for MD4.

Theorem 1 : *One-Way $\not\Rightarrow$ Collision Resistant*

Proof: Assume g is a one-way hash function. Define $h(z \parallel 0) = h(z \parallel 1) = g(z) = y$.

Assume for the purpose of contradiction that there exists an adversary program $\mathcal{A}$ which can invert h. Then it is easy to find x, x' such that $h(x) = h(x')$. Flip the last bit of x to get x'. However, this would allow us to invert g by simply chopping off the rightmost bit output by $\mathcal{A}$. Since this violates our assumption on g, such an adversary cannot exist. h is therefore one-way. But it is clearly not collision resistant since any pair of inputs of equal length but having a different rightmost bits will collide.

■

Theorem 2 *Collision Resistant* $\not\Rightarrow$ *One-Way*

Proof:

Let g be a collision resistant hash function $\{0,1\}^* \to \{0,1\}^n$. Define $h(x) : \{0,1\}^* \to \{0,1\}^{n+1}$:

$$h(x) = \begin{cases} 0 \parallel x & \text{if } \mid x \mid = n \\ 1 \parallel g(x) & \text{otherwise} \end{cases}$$

Assume for the purpose of contradiction that h is not collision resistant. Then an adversary program $\mathcal{A}$ exists which can find a pair x, x' such that $h(x) = h(x')$. But if $h(x)$ and $h(x')$ start with 0, then $x = x'$. This is not a collision. If $h(x)$ and $h(x')$ start with 1, then x, x' are also collisions on g. This violates our assumption. Therefore h must be collision resistant. However, h is not one-way because:

$$h^{-1}(0 \parallel x) = x$$

We can invert half of the output range (of any hash output that starts with 0). ■

Weak Collision Resistance

Weak collision resistance is also known as second pre-image resistance. It is infeasible given an input x, to find an $x' \neq x$ such that $h(x) = h(x')$. Here the adversary has a more difficult time finding collisions because it must collide on a *particular* input rather than any input. WCR vs. CR is analogous to a selective forgery vs. existential forgery.

Theorem 3 *Collision Resistance* $\Rightarrow$ *Weak Collision Resistance*

Proof: If a hash function g is not WCR, then an adversary program $\mathcal{A}$ can find an $x' \neq x$ such that $g(x) = g(x')$. Therefore a different adversary can find a pair w, w' such that $h(w) = h(w')$. Simply pick a random w and run $\mathcal{A}$ to find w'. Therefore g is not CR. ■

Theorem 4 *Weak Collision Resistance* $\not\Rightarrow$ *Collision Resistance*

Proof: By example, let g be a one-way hash function that is also collision-resistant.

Let $g^{(i)}(x)$ denote $g(g(g(...(x)))$ – the result of applying g i times, starting with x.

Define the hash function h to work on pairs of values (x, y), as follows. Given an input pair (x, y), h does the following:

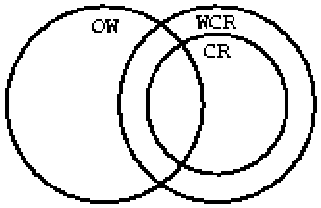

Figure 3: A collision resistant function is also weakly collision resistant.

- Finds the least $i > 0$ such that the binary representation of $g^{(i)}(x)$ ends in 4 zeros.
- Finds the least $j > 0$ such that the binary representation of $g^{(j)}(y)$ ends in 4 zeros. (Actually, we can fix an upper bound, say 100, on these values, just to handle the case that this iteration would otherwise loop forever. This is a small point.)

$h(x, y)$ then returns as output the triple: $(g^{(i)}(x), g^{(j)}(y), i + j)$

This function can be shown to be WCR, since finding a second input that hashes to a given output (corresponding to a first given input) would require inverting g or finding a collision on g.

But the function is also easily seen to be non-collision-resistant, since $(x, g(y))$ and $(g(x), y)$ will very often give the same hash function output. ■

2.2 Applications of Hash Functions

MD5 is OW and WCR. SHA-1 is OW and CR.

- **Password Storage:** *Needs OW and maybe WCR.*
- **Intrusion Detection:** For each file F, store $h(F)$, for example on a CD-R. Keep the CD-R secure. Later, one can see if the file has been modified by recomputing $h(F)$ for each file. The means that an intruder would have to change F but leave $h(F)$. *Needs WCR.*
- **Secure URL (link):** Give an MD5 hash of the target page with the link in the form <a href="..." md5=D4B2...>. *Needs WCR.*
- **Commitment:** One example is a sealed bid auction. Alice wants to bid $\$d$. She submits "Alice", $h(d)$ to the auction, saying she is willing to pay d dollars for Jack. When bidding closes, Alice reveals d. A problem with this scheme is that $h(d)$ may reveal d, since a small range of likely d values can be inverted. We can solve this problem by concatenating a random number in the hash, so we have $h(d \parallel random)$. *Needs CR so Alice cannot change her bid, OW so no one can read the bids, and non-malleability, so, by seeing $h(d)$, Bob cannot create $h(d+1)$.*