CS 346 Class Notes

Mark Lindberg

Mar 9, 2016

Last Time:

Apparently ϵ is the empty string.

This Time:

To find the loop in a linked list (useful for finding collisions in hash function), maintain 2 pointers, A and B. Each step, move A forward 1 step, and B forward 2 steps. Repeat this process until the pointers hold the same value. At time t, then B has advance $2t$ steps, and A has advance t steps. When they collide, they must both be somewhere in the cycle itself.

Because B has advanced t more steps than A, then t must be a multiple of s, the cycle length. The first multiple of $s \geq r$, where r is the number of steps from the beginning of the list until the beginning of the cycle is reached.

Let $k = \left\lfloor \frac{r}{s} \right\rfloor$, and $\ell = r \mod s$. Then $r = ks + \ell$.

If $\ell = 0$, then r is a multiple of s, and $t = ks$.

Else, $\ell > 0$, then they meet at $t = (k+1)s$. Then the loop index of the meeting point is $(k+1)s - (ks + \ell) = s - \ell$.

Now run pointers C and D at speed 1, one from the head of the list, one from the collision point.

C reaches the loop at step $i \cdot s + \ell$ for $i \geq k$.

D reaches loop index 0 at step $i \cdot s + \ell$ for $i \geq 0$.

They will meet at step $ks + \ell$.

Random Oracle Model:

The Algorithm has access to an oracle for a random function from $\ell_{\in}(n)$-bits to $\ell_{out}(n)$-bits. Compare this to f in the definition of a PRF.

We can easily construct a PRG, collision resistant hash function, or a PRF in the random-oracle model.

1. $\ell_{in}(n) < \ell_{out}(n)$, get a PRG such that $\left|\Pr[D^H(y) = 1] - \Pr[D^H(H(x)) = 1]\right| \leq \texttt{negl}(n)$.

2. $\ell_{in} > \ell_{out}(n)$, $\Omega\left(2^{\ell_{out}(n)/2}\right)$.

3. $\ell_{in}(n) = 2n$. $\ell_{out}(n) = n$. $F_k(x) = F(k, x) = y$, $F_k(x) = H(k||x)$.

Additional applications of hash functions:

1. We can track hash values of known viruses.
2. Deduplication in cloud storage.
3. File location/load balancing in P2P systems.
4. Merkle trees.

Password Hashing!

- Store a hash of each user's PWD in the PWD file.
- This means that is it sufficient for the attacker to find *a* preimage of your PWD, which may or may not be your password.

The second item may be weak to something called Hellman's scheme.
Preprocessing time 2^{ℓ}. Uses $2^{\frac{2}{3}\ell}$ space.
Subsequently, can answer a preimage query in $2^{\frac{2}{3}\ell}$ time.
Mitigation techniques to fight these attacks:

1. Use a slow hash function–make it take maybe half a second.
2. Add a salt to every password.