6.857 Computer and Network Security **Fall Term, 1997**

Lecture 10 : October 7, 1997

Lecturer: Ron Rivest *Scribe: Edmond Kayi Lee*

1 Topics Covered

- One-way Hash Functions and Applications
- Randomized Hash Functions
- Auction and Commitment

2 Review: CvHP

CvHP stands for Chaum-vanHeijst-Pfitzmann. It is a one-way hash function which is based on the assumption of that solving the discrete-log problem is difficult.

We pick primes p, q such that $p = 2q + 1$. Let G_q be a subgroup of Z_p^* such that $x \in G_q \leftrightarrow x^q \equiv 1 \pmod p$.

Theorem 1 $x^q \equiv 1 \pmod p$ *if and only if* x *is a square. Thus,* $x \in G_q$ *if and only if* x *is a square.*

Proof: Let $x = g^{2k+i}$ for some generator g of Z_p^*, where $k \in \{0, 1, \ldots, q\}, i \in \{0, 1\}$.

$$\begin{aligned} x^q &\equiv (g^{2k+i})^q \pmod p \\ &\equiv (g^{2q})^k g^{qi} \pmod p \\ &\equiv (g^{p-1})^k g^{qi} \pmod p \\ &\equiv g^{qi} \pmod p \end{aligned}$$

If x is a square, $i = 0$. Therefore $g^{qi} = 1$. If x is not a square, $i = 1$. And $g^{qi} = g^q \neq 1$, since g is a generator. ■

Therefore, we can pick two elements $\alpha, \beta \in G_q$ by simply choosing any element in Z_p^* and squaring it. Alternatively, we can find a generator g of Z_p^* and get a even power of g modulo p.

Now assume we get $\alpha, \beta \neq 1$ such that it is presumed hard to compute $\log_\alpha \beta$, the message $x = x_1, x_2$ is hashed as $h(x_1, x_2) = \alpha^{x_1}\beta^{x_2} \bmod p$.

Here p, α and β is public, nothing is secret. So everybody can create a message digest or verify a message digest. As opposed to MAC or Digital Signature, it cannot be used for authentication. In summary:

	Who can verify	Who can create
One-way Function	Anyone	Anyone
MAC	Shared secret key holders only	Shared secret key holders only
Digital Signature	Anyone	Private key holder only

3 Applications of One-way Hash Functions

- Virus Checking
  - virus modifies file contents
  - producer makes checksum $c = h(F)$ where F is the original file content and h is a weakly collision-free one-way hash function
  - c is distributed to users via secure channels (e.g. floppy disk, CD)
  - users compare c with $h(F')$, where F' is the suspected copy of the file
- Software/Public Key Distribution
  - objective: get the "true" copy of software or public key
  - same idea as virus checking
  - obtain a hash value of the software/public key via a secure channel and compare it to the hash value of user's copy.
- Swapping to insecure storage
  - keep a table of hash values of the pages before swapping them out to an insecure storage
  - check the hash value of the page when retrieved back next time

- Secure Reference or Pointer
  - used as web page reference
  - URLs are hashed
- Time stamp
  - files or documents are hashed and then time-stamped by some time-stamping services
- One-time Password (S-Key)
  - pick a seed x_0 to create a chain of hash values: $x_0 \overset{h(x_0)}{\longrightarrow} x_1 \overset{h(x_1)}{\longrightarrow} x_2 \cdots \overset{h(x_n)}{\longrightarrow} x_n$
  - originally system stores x_n, user enters password x_{n-1}
  - system verifies $h(x_{n-1}) \overset{?}{=} x_n$
  - adversaries cannot find x_{n-1} even if knowing x_n, since h is weakly collision-free
  - system then stores h_{n-1}, next time user enters h_{n-2} as password

4 Randomized Hash Functions

Sometimes we are concerned about whether $h(x)$ reveals any information of x for a one-way hash function h. For example, in a multi-user system, users' passwords are usually hashed by a certain hash function and stored in a password file. Is it totally safe then?

Adversaries may get their way to the password file and read the list of the hashed passwords. It is common (though strongly discouraged) that English words are used as passwords. As a result, adversaries can create a table of hash values for all the dictionary words in advance and look up the hashed passwords in the table. This is known as the *dictionary attack*.

Also, if two people have the same password, the adversary knows that, since the hash function is deterministic.

As a solution, we introduce random factors. To do this, every time a user password is created, a random number r (called a *salt*) is picked and hashed with the password. The random number r is kept for verification. Now the adversary only sees a "random" number.

e.g. $h(x) = (r, MD5(r, x))$ or $h(x) = (r, r^{MD5(x)})$

In this case, the adversary needs to generate a separate dictionary hash table for each random number (i.e. user password). Same passwords are very unlikely to give the same hash values because of the salt.

5 Application: Auction & Commitment

Imagine we are designing a computer system for secure auction. The auction works this way:

1. bidders submit "sealed" bids
2. auction over
3. bidders "open" bids, auctioneer announces the winning bid

All bidders do not want the auctioneer and other people know what their bids are. On the other hand, the auctioneer needs to prevent the bidders from changing their bids once they have submitted. Such a scenario is called *commitment*.

To keep confidentiality, bidders need to somehow encrypt their bids before submitting to the auctioneer and keep the key until the auction is over. To avoid changing bids, the auctioneer has to prevent the bidders from changing their keys because changing the keys essentially changes the bids as well.

One solution may come. The auctioneer chooses a public, randomized, weakly collision-free, one-way hash function h. And each bidder picks a random salt r for himself and makes his bid b. Bidders submit $h(r, b)$ to the auctioneer. At the end, bidders open their bids by disclosing r and b.

In this way, the auctioneer cannot get any information of the bids because the submitted hash values are randomized. On the other hand, bidders cannot change their bids because it is hard to compute another pair (r', b') such that $h(r', b') = h(r, b)$.

One example of the hash functions that can be used is the randomized CvHP. The auctioneer picks p, α and β and bidders choose their own random salt r and their bids b. Bidders submit $\alpha^r \beta^b$ modulo p to the auctioneer. At the end of the auction, the bidders disclose r and b and the auctioneer verifies that $\alpha^r \beta^b$ modulo p is the submitted value.