2/13/19

Admin: Pset #1 due 2/26
Pset #2 out 2/26

Today: Cryptographic Hash Functions

- Definition
- Random Oracle Model (ROM)
- Properties: OW, CR, TCR,
- Applications

Reading: Katz-Lindell (2nd edition) - Chapter 5

Def: Hash family is a family of functions $\{h_s\}$ s.t. $h_s : \{0,1\}^* \longrightarrow \{0,1\}^d$

bit strings of any length.

and given (s,x) one can efficiently compute $h_s(x)$.

s is called "the seed" and is public.

$h_s(x)$ is called "hash value" or "message digest".

An ideal hash function : A "Random Oracle" (RO)

- Theoretical model, not achievable in practice, called Random Oracle Model (ROM) :

- Model the hash function as an oracle (Black Box), that on any input $x \in \{0,1\}^*$:
  - If x was not queried before output a truly random value in $\{0,1\}^d$, denoted by $h(x)$
  - O.w. if x was already queried, return the same answer.

The oracle records all queries, and returns a truly random value in $\{0,1\}^d$.

* Many cryptographic primitives use hash functions, and security is proved in the ROM.

* In practice the hash function is implented using one of the standardized hash functions, such as SHA-256 (SHA: Secure Hash Algorithm)

SHA-256 is not a random oracle, but is hopefully "pseudo random enough" that an adv cannot exploit any flaws in it.

Desirable properties for hash functions :

(1) One-way (OW) : Given random $y \in \{0,1\}^d$ hard to find $x \in \{0,1\}^*$ s.t. $h(x) = y$

[Note that such x exists (w.h.p.) and can be found via "brute force" in time $\Theta(2^d)$ (even in ROM)]

(2) Collision Resistance (CR)

Given a seed s it is hard to find any distinct $x, x' \in \{0,1\}^*$ s.t. $h_s(x) = h_s(x')$.

- This property cannot hold unless the hash is seeded (i.e. chosen at random from a family of functions) since o.w. a collision can simply be "hard wired"

[In the ROM collisions can be found in time $\approx 2^{d/2}$: Query $x_1, x_2, \ldots$ until a pair x_i, x_j collide. This is called the "birthday paradox"]

3 Target Collision Resistance (TCR):

Given any $x \in \{0,1\}^n$ and given a random seed s it is hard to find $x' \neq x$ s.t. $h_s(x) = h_s(x')$

Similar to CR but one preimage is fixed and known.

[In ROM can find x' in time $O(2^d)$, similar to OW, since knowing x does not help finding x' in the ROM]

4 Pseudo-randomness (PRF):

Obtaining black box access to h_s (for random s) is computationally indistinguishable (i.e., indisting. by poly-bounded adv.) from a RO.

- This property cannot hold unless the hash is seeded (i.e., is chosen randomly from a family of functions).

Informal...

5 Non-malleability (NM):

Given $h(x)$ for a randomly chosen x, it is hard to produce $h(x')$ where x' is related to x (e.g. $x' = x+1$)

Thm: 1. $\{h_s\}$ is CR $\Longrightarrow$ $\{h_s\}$ is TCR.

(the converse does not hold).

2. $\{h_s\}$ is CR $\Longrightarrow$ $\{h_s\}$ is OW

since h_s compresses.

(the converse does not hold).

Example: ($\not\Leftarrow$) Consider $\{h_s\}$ that is OW and TCR s.t.

$h_s(0) = h_s(1)$ or $h_s(s) = h_s(s+1)$.

Hash Function Applications:

(1) Password Storage (for login)

- Store $h(pw)$, rather than pw
- When user logs in, check that hash of pw is consisten with stored value
- Security: $h(pw)$ should not reveal pw or any preimage that hashes to $h(pw)$.

Need OW

2. File modification detector:

- For each file F store $h(F)$ securely.
- Can check if F was modified by computing $h(F)$.
- Security: Given F and h should be hard to find F' s.t. $h(F) = h(F')$

Need **TCR**

3. Digital signatures (hash & sign):

Each user, say Alice has keys (PK_A, SK_A).

PK_A = Alice's public key (used to verify Alice's signature

SK_A = Alice's secret key (used for signing)

Signing: $\sigma = \text{sign}(SK_A, m)$ can be randomized

$\text{Verify}(PK_A, m, \sigma) \in \{acc/rej\}$

If m is very long this can be quite inefficient.

The hash-&-sign paradigm:

Sign $h(m)$ (as opposed to m),

Intuitively $h(m)$ is a "proxy" for m.

<u>Security</u>: An adversary cannot forge a signature to <u>any</u> message even if he sees signatures of many (other) messages of <u>his choice</u>.

Need <u>CR</u>

Else, an adv. can find $m \neq m'$ s.t. $h(m) = h(m')$ and ask Alice to sign m, and then can use this same signature as a (valid) signature for m'.

4) <u>Commitments</u>:

A commitment scheme allows any user, Alice, to commit to a value x (eg., an auction bid), denoted by $com(x, r)$ s.t.

randomized alg

• <u>Binding property</u>: Alice should not be able to open the commitment in more than one way eg. it is hard to find (x,r) & (x',r') s.t.

$x \neq x'$ and $com(x,r) = com(x';r')$.

<u>hiding property</u>: $com(x,r)$ shoueld reveal no information about x. Namely $\forall x, x'$ (of same size) $com(x,r) \cong com(x',r')$

looks the same in the eye of a poly-time adv.

<u>Non-malleability</u>: Given $com(x;r)$ it should be hard to compute a commitment to a related value, say $com(x+1;r')$.

Idea: $com(x;r) = h(x,r)$

<u>Need</u>: For binding - <u>CR</u>

For non-malleability - <u>NM</u>

For hiding: ?