<u>Today</u>

3/15/2021

Lec 8

* Shamir Secret Sharing
* Diffie Hellman key exchange

- Secrets are paramount in cryptography.
- How do we store a secret securely?
  * A computer can be compromised, and thus the secret can leak
  * A secret can be lost.

<u>Shamir's idea:</u> Share the secret among several computers s.t. even if "some" are compromised the secret does not leak, and even if "some" are lost the secret can be recovered.

More formally, share a secret s to n parts $s_1, \ldots, s_n$ s.t. given more than t shares one can efficiently reconstruct s, and given less than t shares, s is information theoretically hidden.

Def: A (n,t) secret sharing scheme consists of 2 PPT alg (Share, Reconstruct) s.t.

- Share(s) outputs $(s_1,\dots,s_n)$
- Reconstruct $(I, \{s_i\}_{i\in I})$ outputs s if $I \subseteq [n]$ is of size $\geq t$
- **Security**: $\forall I \subseteq [n]$ of size $\leq t-1$, $\{s_i\}_{i\in I}$ reveals no information about s; i.e. $\forall s, s'$, $\forall I \subseteq [n]$ of size $< t$

$$\{s_i\}_{i\in I} \equiv \{s_i'\}_{i\in I}$$

where $(s_1 \dots s_n) \leftarrow$ Share(s) & $(s_1' \dots s_n') \leftarrow$ Share(s')

Easy cases:

$t=1$: $s_i = s$

$t=n$: $s_1,\dots,s_n$ random s.t. $s_1 \oplus \dots \oplus s_n = s$

This can be done by choosing $s_1,\dots,s_{n-1}$ at random & setting $s_n = s \oplus s_1 \oplus \dots \oplus s_{n-1}$

What about $1 < t < n$?

Shamir Secret Sharing scheme ("How to Share a Secret" 1979)

Suppose $s \in GF[P]$ for some prime p.

Share(s) :
- Choose at random $a_1, .., a_{t-1} \leftarrow GF[P]$, and set $a_0 = S$
- Let $f(x) = \sum_{i=0}^{t-1} a_i x^i$
- Let $S_i = \underbrace{f(i)}_{y_i} \quad \forall i \in [n]$

Reconstruct(I, $\{S_i\}_{i \in I}$): Via interpolation :

Given t evalution points on a deg $t-1$ polynomial f, one can efficiently compute f, and in particular compute $f(0)$.

Formally, given (x_i, y_i) $1 \le i \le t$ (wlog)

let
$$f_i(x) = \begin{cases} 1 & \text{if } x = x_i \\ 0 & \text{o.w.} \end{cases}$$

$$f_i(x) = \frac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} (x_i - x_j)}$$

we assume $x_1, .., x_t$ are distinct

f_i is a degree $t-1$ polynomial.

Let $f(x) = \sum_{j=1}^{t} f_j(x) y_j$

Note that $f(x_i) = \sum_{j=1}^{t} f_j(x_i) y_j = y_i \quad \forall i \in [t]$

Output $s = f(0)$

Correctness follows from the fact that there exists a <u>unique</u> deg $\le t-1$ polynomial f s.t. $f(x_i) = y_i \ \forall i \in [t]$.

(This is the case since o.w. $\exists$ deg $\le t-1$ polynomial with t roots — contradiction.)

<u>Thm</u>: Shamir's scheme is (information theoretically) secure

(I.e., an adv w. $< t$ shares has no information about s)

<u>Proof</u>: Fix any s, and any non-zero $x_1, \ldots, x_{t-1} \leftarrow GF[p]$.

For random $a_1, \ldots, a_{t-1} \leftarrow GF[p]$

let $y_i = \sum_{i=1}^{t-1} a_i x^i + s$

Then $y_1, \ldots, y_{t-1}$ are random.

Follows from linear algebra:

Let $a = (s, a_1, \ldots, a_{t-1}) \in GF[p]^t$

$\forall i \in \{0, 1, \ldots, t-1\}$, let

$v_i = (1, x_i, \ldots, x_i^{t-1}) \in GF[p]^t$

where $x_0 = 0$

$v_0, v_1, \ldots, v_{t-1}$ are linearly independent

(Vandermonde matrix)

$\Rightarrow \underbrace{v_1 \cdot a}_{= y_1}, \ldots, \underbrace{v_{t-1} \cdot a}_{= y_{t-1}}$ are random in $GF[p]$

since $a_1, \ldots a_{t-1}$ are random in $GF[p]$

Diffie-Hellman Key Exchange

- We discussed secret key cryptography, allows Alice & Bob to send msgs securely (private & authenticated) assuming they share a secret key

<u>Q</u>: How can they establish a shared secret key in the presence of a (passive) eavesdropper?

<u>A</u>: Using a key-exchange protocol, a precursor to public key cryptography.

Let G be a finite cyclic group with generator g

$$G = \{1, g, g^2, \dots, g^{|G|-1}\}$$

G & g are fixed and public

A		B
Choose a random secret $x \leftarrow \{0,1,\dots,|G|-1\}$		choose a random secret $y \leftarrow \{0,1,\dots,|G|-1\}$
	$\xrightarrow{\quad g^x \quad}$	
Compute $K = (g^y)^x$	$\xleftarrow{\quad g^y \quad}$	Compute $K = (g^x)^y$

The shared secret is $K = g^{x \cdot y}$

The eavesdropper Eve only learns (g^x, g^y)

__Assumption__ : Decisional Diffie Hellman (DDH)

$$(g^x, g^y, g^{x \cdot y}) \cong (g^x, g^y, g^r)$$

$$x, y, r \leftarrow \{0, 1, \ldots, |G| - 1\}$$

__Thm__: DH Key Exchange is secure against passive attacks under the DDH assumption.

A weaker (insufficient) assumption:

__Computational Diffie-Hellman__ (CDH):

Given g^x, g^y it is hard to compute g^{xy}.

Formally, $\forall$ PPT $\mathcal{A}$ $\exists$ negl μ s.t.

$$\Pr[\mathcal{A}(g^x, g^y) = g^{x \cdot y}] \leq \mu(\underbrace{K}_{\log|G|})$$