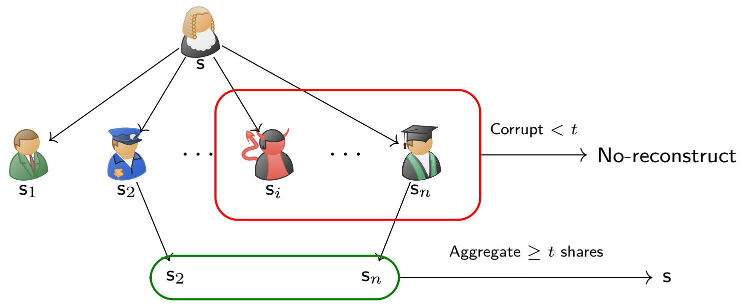
CS 171 - Cryptography

Sanjam Garg

Lecture 23

$(t, n)-$Threshold Secret Sharing

- A (t, n) threshold secret sharing scheme allows one to split a secret s into n pieces so that one will need at least t shares to reconstruct s.
- A dealer takes s as input and uses a sharing algorithm to split the secret s into parts $s_1 \dots s_n$ to be given parties $P_1, \dots P_n$.



- Correctness: Any t parties can reconstruct s.
- Security: No collusion of $< t$ parties can reconstruct s.

$(t, n)-$Threshold Secret Sharing

A (t, n)-secret sharing scheme $(\mathsf{Share}, \mathsf{Reconstruct})$ is defined as follows.

- $\mathsf{Share}(s)$: On input a secret s it outputs shares $s_1, \ldots s_n$.
- $\mathsf{Reconstruct}(\{s_i\}_{i \in T})$: Outputs s or $\perp$.
- Correctness: For any T such that $|T| \geq t$ and secret s we have that $\mathsf{Reconstruct}(\{s_i\}_{i \in T}) = s$.
- Security: For any T such that $|T| < t$, secrets s, s' and adversary $\mathcal{A}$ we have that $p = p'$ where

$$p = Pr[\mathcal{A}(\{s_i\}_{i \in T}) \mid (s_1, \ldots s_n) \leftarrow \mathsf{Share}(s)],$$
$$p' = Pr[\mathcal{A}(\{s'_i\}_{i \in T}) \mid (s'_1, \ldots s'_n) \leftarrow \mathsf{Share}(s')].$$

$(2, 2)-$ Threshold Secret Sharing

- Let $s \in \{0, 1\}^m$. How do we $(2, 2)$-secret share s?
- $\mathsf{Share}(s)$: Sample $r \leftarrow \{0, 1\}^m$ and output $s_1 = r$ and $s_2 = s \oplus r$.
- $\mathsf{Reconstruct}(s_1, s_2)$: Outputs $s_1 \oplus s_2$.
- Correctness: By constrcution, $s = s_1 \oplus s_2$.
- Security: For any s, each individual s_1 or s_2 is uniformaly random. Thus, $p = p' = q$ where:

$$q = Pr[\mathcal{A}(r) \mid r \leftarrow \{0, 1\}^m].$$

$(n, n)-$ Threshold Secret Sharing

- Let $s \in \{0,1\}^m$. How do we $(2,2)$-secret share s?
- Share(s) : Sample $r_1 \dots r_{n-1} \leftarrow \{0,1\}^m$ and output $s_1 = r_1$, $s_2 = r_2 \dots s_{n-1} = r_{n-1}$ and $s_n = s \oplus_{i=1}^{n-1} r_i$.
- Reconstruct$(s_1, s_2 \dots s_n)$: Outputs $\oplus_{i=1}^{m} s_i$.
- Correctness: By constrcution, $s = \oplus_{i=1}^{m} s_i$.
- Security: For any s, T such that $|T| < n$, $\{s_i\}_{i \in T}$ is uniformaly random. Thus, $p = p' = q$ where:

$$q = Pr[\mathcal{A}(\{r_i\}) \mid r_1 \dots r_{|T|} \leftarrow \{0,1\}^m].$$

$(3, 3)-$Threshold Secret Sharing

- Let $s \in \{0,1\}^m$. How do we $(3,3)$-secret share s?
- Share(s) : Sample $r_1, r_2 \leftarrow \{0,1\}^m$ and output $s_1 = r_1, s_2 = r_2$ and $s_3 = s \oplus r_1 \oplus r_2$.
- Reconstruct(s_1, s_2, s_3): Outputs $s_1 \oplus s_2 \oplus s_3$.
- Correctness: By construction, $s = s_1 \oplus s_2 \oplus s_3$.
- Security: For any s, s_i, s_j for any $i, j \in \{1,2,3\}$ are uniformaly random. Thus, $p = p' = q$ where:

$$q = Pr[\mathcal{A}(r_1, r_2) \mid r_1, r_2 \leftarrow \{0,1\}^m].$$

$(2,3)-$Threshold Secret Sharing

- Let $s \in \{0,1\}^m$. How do we $(2,3)$-secret share s?
- Share(s) : Sample $r_1, r_2 \leftarrow \{0,1\}^m$. Set $r_3 = s \oplus r_1 \oplus r_2$ and output $s_1 = (r_1, r_2), s_2 = (r_2, r_3)$ and $s_3 = (r_3, r_1)$.
- Reconstruct(s_i, s_j): Outputs $r_1 \oplus r_2 \oplus r_3$ where r_1, r_2, r_3 can be recovered from s_i, s_j.
- Correctness: By construction, $s = r_1 \oplus r_2 \oplus r_3$.
- Security: For any s, s_i for any $i \in \{1,2,3\}$ is uniformaly random. Thus, $p = p' = q$ where:

$$q = Pr[\mathcal{A}(r_1, r_2) \mid r_1, r_2 \leftarrow \{0,1\}^m].$$

$(2,n)-$Threshold Secret Sharing

- Let $s \in \{0,1\}^m$. How do we $(2,n)$-secret share s (assume $n = 2^k$)?
- Share(s) : Sample $r_1, \ldots r_k \leftarrow \{0,1\}^m$. For each $i = i_1 \ldots i_k$ and $j = 1 \ldots k$ generate

$$s_{i,j} = r_j$$

  if $i_j = 0$ and as

$$s_{i,j} = r_j \oplus s$$

  if $i_j = 1$. Output $s_i = (s_{i,1} \ldots s_{i,k})$
- Reconstruct$(s_i = (s_{i,1} \ldots s_{i,k}), s_{i'} = (s_{i',1} \ldots s_{i',k}))$: Outputs $s_{i,j} \oplus s_{i',j}$ for a j such that $i_j \neq i'_j$.
- Correctness: This can be checked by construction.
- Security: For any s, s_i is uniformaly random vector of k strings. Thus, $p = p' = q$ where:

$$q = Pr[\mathcal{A}(r_1, \ldots r_k) \mid r_1, \ldots r_k \leftarrow \{0,1\}^m].$$

Can we build (t, n)-secret sharing for any t, n such that $t \le n$?

Yes! Shamir's Secret Sharing Scheme.

Shamir's Secret Sharing: Background

- We consider a polynomials $p(x) \in \mathbb{Z}_q[x]$ where q is a prime.
- $p(x)$ is denoted as $a_0 + a_1 x \ldots a_t x^t \mod q$. If $a_t \neq 0$ then $p(x)$ has degree t.
- $p(x) = p'(x)$ if they have the same degree and agree on all coefficients.

Theorem: Any two distinct degree-t polynomials agree on at most t points.

- Proof: Suppose $p(z_i) = p'(z_i)$ for $i \in \{1 \ldots t+1\}$.
- Let $q(x) = p(x) - p'(x)$. Then we have that $q(x) = 0$ for all $x \in \{z_1 \ldots z_{t+1}\}$.
- However, $q(x)$ is of degree $\le t$ and has $t+1$ root. Contradiction!

Shamir's Secret Sharing

Key idea:

- If we have t points of a polynomial of degree $t-1$, we can reconstruct the polynomial. Moreover, the polynomial is unique.

Theorem: Given t distinct input/output points $(x_1, y_1) \dots (x_t, y_t)$, we can find in poly time the unique degree-$(t-1)$ polynomial $p(x)$, where $p(x_i) = y_i$ for $i \in \{1 \dots t\}$.

$(t, n)-$Shamir's Secret Sharing

Main Idea: To share $s \in \mathbb{Z}_q$: choose a random degree $t-1$ polynomial $p(x)$ such that $p(0) = s$. Give out the shares $(p(1), \dots, p(n))$.

- Given t shares, we can reconstruct $p(x)$, and can then recover $p(0)$.

Sharing:

- Given a secret $s \in \mathbb{Z}_q$, choose $p(x) = s + a_1 x + \dots a_{t-1} x^{t-1}$, where a_i's are chosen randomly in $\mathbb{Z}_q$. Give out the shares $(p(1), \dots, p(n))$.

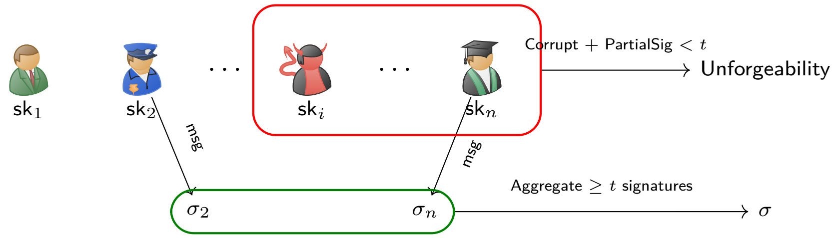
Reconstruct:

- Given t values $(i_1, p(i_1)), \dots, (t, p(i_t))$, reconstruct p and output $p(0)$.

Practice Problem

- Given encryption schemes $\Pi_1 \dots \Pi_n$ (where $\Pi_i = (Gen_i, Enc_i, Dec_i)$) such that at least t of them are CPA-secure. Construct an encryption scheme that is CPA-secure.

(t, n)-Threshold Signature [Desmedt'87, Desmedt-Frankel'89]



- A succinct (constant-size) public/verification key vk.
- Aggregated signatures σ are succinct (constant-size).
- Widely used in blockchain applications.

BLS Signature [Boneh-Lynn-Shacham'01]

- $\mathsf{s} \leftarrow \mathbb{Z}_q$, $\mathsf{vk} = g^{\mathsf{s}}$.
- Signature is $\sigma = \mathsf{H}(\mathsf{msg})^{\mathsf{s}}$.
- Verify signature: $e(\mathsf{H}(\mathsf{msg}), \mathsf{vk}) \stackrel{?}{=} e(\sigma, g)$

BLS Multisignature: n-out-of-n threshold signature

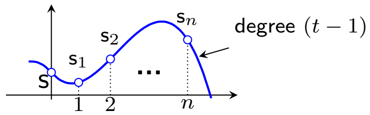
- Each party picks $\mathsf{s}_i \leftarrow \mathbb{Z}_q$, $\mathsf{vk}_i = g^{\mathsf{s}_i}$
- Partial signature $\sigma_i = \mathsf{H}(\mathsf{msg})^{\mathsf{s}_i}$

$$\begin{cases} e(\mathsf{H}(\mathsf{msg}), \mathsf{vk}_1) \stackrel{?}{=} e(\sigma_1, g) \\ \vdots \\ e(\mathsf{H}(\mathsf{msg}), \mathsf{vk}_n) \stackrel{?}{=} e(\sigma_n, g) \end{cases}$$

- Verification key $\mathsf{aVK} = \prod_i \mathsf{vk}_i$
- Aggregated Signature $\sigma = \prod_i \sigma_i$
- Verify signature: $e(\mathsf{H}(\mathsf{msg}), \mathsf{aVK}) \stackrel{?}{=} e(\sigma, g)$

BLS t-out-of-n threshold signature

- Generate $\mathsf{s} \leftarrow \mathbb{Z}_q$, $\mathsf{vk} = g^{\mathsf{s}_i}$.
- vk is published, i^{th} party receives s_i.
- $\mathsf{s}_1, \ldots, \mathsf{s}_n$ forms a t-out-of-n linear secret sharing of s.



Signing and Aggregation

- Signing: Partial signature $\sigma_i = (\mathsf{H}(\mathsf{msg}))^{\mathsf{s}_i}$ for message msg.
- Linear secret sharing property: For any set $T \subseteq \{1 \ldots n\}$ such that $|T| \geq t$ we have constants $\{\alpha_i^T\}_{i \in T}$ such that $\mathsf{s} = \sum_{i \in T} \alpha_i^T \cdot \mathsf{s}_i$.
- Given $\{\sigma_i\}_{i \in T}$ compute $\sigma = \mathsf{H}(\mathsf{msg})^{\mathsf{s}}$ as

$$\mathsf{H}(\mathsf{msg})^{\mathsf{s}} = \mathsf{H}(\mathsf{msg})^{\sum \alpha_i^T \cdot \mathsf{s}_i} = \prod_{i \in T} (\mathsf{H}(\mathsf{msg})^{\mathsf{s}_i})^{\alpha_i^T} = \prod_{i \in T} \sigma_i^{\alpha_i^T}$$