Secret sharing schemes

Secret sharing schemes

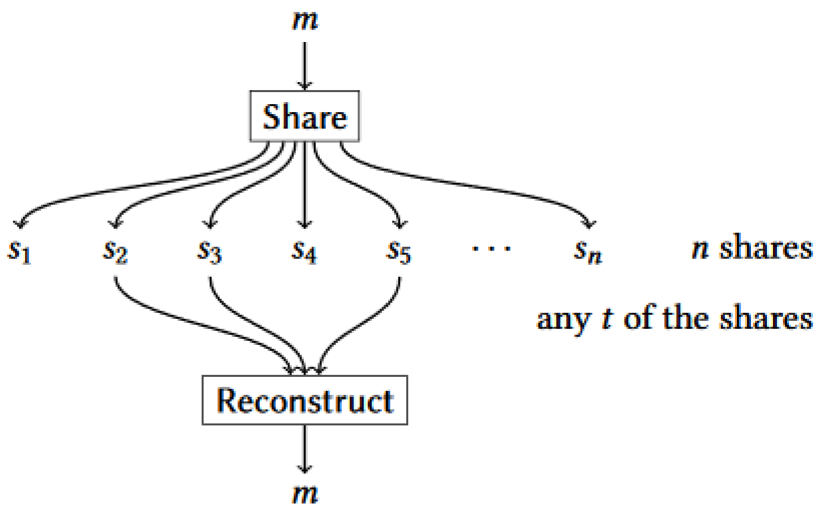
Another type of problem that cryptography tries to solve is the question "Who can be trusted to keep a secret". One of the ways to solve that problem is to split the secret between multiple parties such that no party can compute the secret alone and a minimum amount of parties are needed to compute back the secret.

- https://en.wikipedia.org/wiki/Secret_sharing

t-out-of-n threshhold secret-sharing-scheme(TSSS)

Algorithms needed

> **Share**: A randomized algorithm takes a message $m \in \mathcal{M}$ and splits it into a sequence $s = s_1, ..., s_n)$ of **shares**
> **Reconstruct**: A deterministic algorithm that takes a collection of t or more shares as input and outputs a message



Security

Idea:

> If you know only an unauthorized set of shares, then you learn **no information** about the choice of secret message.
> Someone with fewer than t shares has **no more information** than someone with 0 shares

Notes

The attacker gets the shares through different calls

- Suppose we have a 4 out of 6 sharing scheme and we make two calls.

- We get the shares {1,2,3} the first call
    - We get the shares {4,5,6} the second call
- Although it doesn't seem so, the attacker **should not** be able to compute the message or find any information about it since the shares come from two **independent** calls of the share function

We do not address the problem of who should run the share algorithm or how the shares get to the users

Insecure examples: Addition

Suppose

- we have a message $m \in \{0, 1\}^{500}$
- we split the message into 5 shares of $100b => (s_1, ..., s_5)$
- this is a 5-out-of-5 share

This scheme is insecure: Suppose you have 1 share

- **Knowing 1 share you know more than someone who knows 0 shares**
    - Example: In a brute force attack, you have to brute force less bits
- **Indistinguishability**
    - You queue 1^{500} and 0^{500} and you get s_1 for each of them
    - Your attack scheme is: check if $s_1 == 0^{500}$ and return 1 if true else false
    - You will return 1 with probability 1 when queuing 0^{500}
    - You will return 1 with probability 0 when queuing 1^{500}
    - Therefore you can distinguish which message was shared

Secure example with t = n

The simplest example: Multiple OTP

- Suppose m is the secret that we want to share to n participants
- Generate $n - 1$ random numbers s_i
- the last secret $s_n = m \oplus s_1 \oplus ... \oplus s_{n-1}$

Resources

- https://www.youtube.com/watch?v=iFY5SyY3IMQ
- https://www.youtube.com/watch?v=K54ildEW9-Q&t
- https://web.engr.oregonstate.edu/~rosulekm/crypto/crypto.pdf - SSS chapter