Secret sharing: Suppose we have a secret and want to distribute it among n parties such that any t of them can subsequently recover the secret and any $(t-1)$ subset cannot [e.g., Board of directors at Coca-Cola want to protect Coca-Cola recipe]

Two algorithms: $\text{Share}(m) \to \{s_i\}_{i \in [n]}$: takes a message m and outputs a collection of n shares

$\text{Reconstruct}(\{s_j\}) \to m / \perp$: takes a set of shares and reconstructs the message

How NOT to share a secret:

Secret $s \in \{0,1\}^\ell$ $\quad s = s_1 \| s_2 \| \cdots \| s_n$ where $s_i \in \{0,1\}^{\ell/n}$

shares: $s_1, s_2, \ldots, s_n$

Given subset of the shares $s_1, \ldots, s_{n-1}$, can learn most of the secret

↳ if secret sharing scheme is n-out-of-n, should not learn anything given just $n-1$ shares

Examples: Additive secret sharing [n out of n] : messages are over $\mathbb{Z}_p$ (for any modulus p, not necessarily prime)

- $\text{Share}(m)$: Sample $r_1, \ldots, r_{n-1} \xleftarrow{R} \mathbb{Z}_p$ and set $r_n = m - \sum_{i=1}^{n-1} r_i \in \mathbb{Z}_p$
- $\text{Reconstruct}(r_1, \ldots, r_n)$: Output $\sum_{i=1}^{n} r_i$

Combinatorial secret sharing [t out of n]: Will use a symmetric encryption scheme (satisfying CPA-security) over $\{0,1\}^*$ (e.g. AES-CTR)

- $\text{Share}(m)$: Sample n keys $k_1, \ldots, k_n$ for encryption scheme

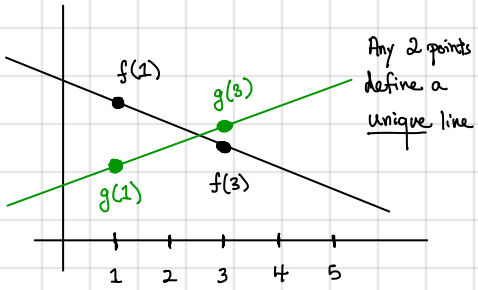
  For every t-subset $\{i_1, \ldots, i_t\} \subseteq [n]$, encrypt m using $k_{i_1}, \ldots, k_{i_t}$ (e.g. $\text{Enc}(k_{i_1}, \text{Enc}(k_{i_2}, \cdots, \text{Enc}(k_{i_t}, m) \cdots))$)

  Let $\{ct\}$ be the collection of ciphertexts

  Output shares $\left((k_1, \{ct\}), \ldots, (k_n, \{ct\})\right)$
- $\text{Reconstruct}\left((k_1, \{ct\}), \ldots (k_t, \{ct\})\right)$: by construction, there is one $ct \in \{ct\}$ encrypted under $k_1, \ldots, k_t$ (relies on computational assumptions), so decrypt accordingly

Shamir secret sharing [t out of n]: will work over $\mathbb{Z}_p$ (for prime p) (more generally, any finite field)

↳ beautiful construction based on polynomials (very useful mechanism for sharing data)



Key idea: Any d points uniquely define a degree-$(d-1)$ polynomial over $\mathbb{Z}_p$

- e.g. 2 points define a line, 3 points define a parabola, etc.
- given d points, can efficiently obtain entire polynomial [Lagrange interpolation]

- $\text{Share}(m)$: Choose $y_1, \ldots, y_{t-1} \xleftarrow{R} \mathbb{Z}_p$

  Let $f: \mathbb{Z}_p \to \mathbb{Z}_p$ be the unique polynomial of degree $t-1$

  $f(0) = m$ and $f(i) = y_i \quad \forall i \in [t-1]$ [t points uniquely define the polynomial f]

  Output shares $(i, f(i)) \quad \forall i \in [n]$ Each share is just 2 field elements (independent of threshold t or # parties n)
- $\text{Reconstruct}((i_1, y_1), \ldots, (i_t, y_t))$: Interpolate the unique polynomial f of degree $(t-1)$ defined by the points $(i_1, y_1), \ldots, (i_t, y_t)$

  Output $f(0)$

A little more detail... how to construct the polynomial f. Lagrange interpolation.

Let $(x_0, y_0), \ldots, (x_t, y_t)$ be a collection of $t+1$ points. To find the polynomial f of degree t that interpolates these points, we can write

$$f(x) = a_0 + a_1 x + \cdots + a_t x^t, \text{ where } a_0, \ldots, a_t \in \mathbb{Z}_p$$

Then, we can write

$$\begin{aligned} f(x_0) &= a_0 + a_1 x_0 + \cdots + a_t x_0^t = y_0 \\ &\vdots \\ f(x_t) &= a_0 + a_1 x_t + \cdots + a_t x_t^t = y_t \end{aligned} \quad \Longrightarrow \quad \underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^t \\ 1 & x_1 & x_1^2 & \cdots & x_1^t \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_t & x_t^2 & \cdots & x_t^t \end{bmatrix}}_{\text{"Vandermonde matrix" of dimension } t+1} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_t \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_t \end{bmatrix}$$

Interpolating a polynomial over $\mathbb{Z}_p$ just corresponds to solving a linear system over $\mathbb{Z}_p$. A unique solution exists as long as the Vandermonde matrix is invertible. It turns out that you can show (via linear algebra) that

$$\det \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^t \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_t & x_t^2 & \cdots & x_t^t \end{pmatrix} = \prod_{0 \le i < j \le t} (x_j - x_i)$$

[If x_i, x_j are all distinct, and we work over a finite field (e.g., there are no non-trivial divisors of 0), then this matrix is invertible and we can interpolate efficiently]

Let us now analyze the properties of Shamir's secret sharing scheme:

Correctness: Follows by uniqueness of interpolating polynomial (e.g., t shares uniquely define a polynomial of degree $t-1$)

Security: Given any subset of $(t-1)$ shares $(i_1, y_1), \ldots, (i_{t-1}, y_{i-1})$, and any message $m \in \mathbb{Z}_p$, there is a unique polynomial f of degree $t-1$ where

$$f(i_1) = y_1, \ \ldots, \ f(i_{t-1}) = y_{i-1} \text{ and } f(0) = m$$

Thus, any $(t-1)$ shares can be consistent with secret-sharing of any message $m \Rightarrow$ information-theoretic security

Efficiency: Both share-generation and share reconstruction consist of polynomial evaluation and interpolation, both of which are efficiently computable (see above)

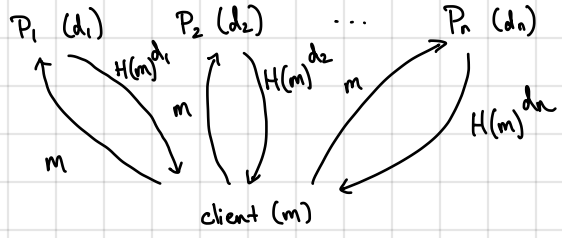
Secret sharing is very useful for building threshold cryptosystems. Here, we describe one example with threshold RSA signatures:

Setup: sample primes p, q, set $N = pq$
choose e, d such that $ed = 1 \pmod{\varphi(N)}$ → vk: (N, e), sk: d (useful for protecting signing keys)

Sign(sk, m): Output $\sigma \leftarrow H(m)^d$

Verify(vk, m, σ): Output 1 if $H(m) = \sigma^e$

Can apply n-out-of-n secret sharing to signing key d: sample $d_1, \ldots, d_n \xleftarrow{R} \mathbb{Z}_{\varphi(N)}$ such that $\sum_{i=1}^{n} d_i = d \pmod{\varphi(N)}$:



to steal signing key, now need to compromise n parties rather than single party

Combine signatures: $\prod_{i=1}^{n} H(m)^{d_i} = H(m)^{\sum_{i=1}^{n} d_i} = H(m)^d = \sigma$