CS 594 – Advanced Crypto (Spring 2026)

Alex Block

Lecture 7

February 09, 2026

Secret Sharing

Last Time: t-out-of-n Secret Sharing

- We built a (t, n) secret-sharing scheme from a simple (n, n) additive secret-sharing scheme.

Last Time: t-out-of-n Secret Sharing

- We built a (t, n) secret-sharing scheme from a simple (n, n) additive secret-sharing scheme.

$\Sigma = (\mathsf{Share}, \mathsf{Reconstruct})$
$\mathcal{M} = \mathcal{S} = \{0,1\}^\ell$

$S_{T,i_1}, \ldots, S_{T,i_t}$

$\mathsf{Share}(m)$:
$\forall T \subset [n]$ s.t. $|T| = t$,
let $T = \{i_1, \ldots, i_t\}$
  for $j = 1, \ldots, t-1$
    $S_{T,i_j} \xleftarrow{\$} \{0,1\}^\ell$
  $S_{T,i_t} = m \oplus \bigoplus_{j=1}^{t-1} S_{T,i_j}$
  for $j = 1, \ldots, t$
    $\mathbb{S}_{i_j} = \mathbb{S}_{i_j} \cup \{S_{T,i_j}\}$
 return $(\mathbb{S}_1, \ldots, \mathbb{S}_n)$

$\mathsf{Reconstruct}(\mathbb{S}_{i_1}, \ldots, \mathbb{S}_{i_t})$:
Compute $T \subset [n]$ s.t.
$T = \{i_1, \ldots, i_t\}$
 return $\bigoplus_{j=1}^{t} S_{T,i_j}$
$//S_{T,i_j} \in \mathbb{S}_{i_j}$ *for all* j

Last Time: t-out-of-n Secret Sharing

Pros

- Simple construction

Cons

Inefficient (horrible) space & time

Better (t, n) Secret Sharing?

Better (t, n) Secret Sharing?

Can we make a *more efficient* (t, n) secret-sharing scheme?

poly-time (n, t)
- Share
- Reconstruct

$\longmapsto$ parties only have poly-size shares

Constant-sized?

Better (t, n) Secret Sharing?

Can we make a *more efficient* (t, n) secret-sharing scheme?

Efficiency of Secret Sharing

Better (t, n) Secret Sharing?

Can we make a *more efficient* (t, n) secret-sharing scheme?

Efficiency of Secret Sharing

- *Time* to generate secret shares;

Better (t, n) Secret Sharing?

Can we make a *more efficient* (t, n) secret-sharing scheme?

Efficiency of Secret Sharing

- *Time* to generate secret shares;
- *Time* to reconstruct secret from shares;

Better (t, n) Secret Sharing?

Can we make a *more efficient* (t, n) secret-sharing scheme?

Efficiency of Secret Sharing

- *Time* to generate secret shares;
- *Time* to reconstruct secret from shares;
- *Size of shares* required by each party to store.

Better (t, n) Secret Sharing?

Can we make a *more efficient* (t, n) secret-sharing scheme?

Efficiency of Secret Sharing

- *Time* to generate secret shares;
- *Time* to reconstruct secret from shares;
- *Size of shares* required by each party to store.
  - This includes number of shares.

Better (t, n) Secret Sharing?

Can we make a *more efficient* (t, n) secret-sharing scheme?

Efficiency of Secret Sharing

- *Time* to generate secret shares;
- *Time* to reconstruct secret from shares;
- *Size of shares* required by each party to store.
  - This includes number of shares.

Our Targets

Better (t, n) Secret Sharing?

Can we make a *more efficient* (t, n) secret-sharing scheme?

Efficiency of Secret Sharing

- *Time* to generate secret shares;
- *Time* to reconstruct secret from shares;
- *Size of shares* required by each party to store.
  - This includes number of shares.

Our Targets

- *Polynomial-time* share and reconstruction algorithms;

Better (t, n) Secret Sharing?

Can we make a *more efficient* (t, n) secret-sharing scheme?

Efficiency of Secret Sharing

- *Time* to generate secret shares;
- *Time* to reconstruct secret from shares;
- *Size of shares* required by each party to store.
  - This includes number of shares.

Our Targets

- *Polynomial-time* share and reconstruction algorithms;
- *Constant-sized* shares (with respect to some security parameter).

Shamir Secret Sharing

Shamir Secret Sharing

- Idea: use *random polynomials* to share a secret!

Shamir Secret Sharing

- Idea: use *random polynomials* to share a secret!

- Fix (t, n).

Shamir Secret Sharing

- Idea: use *random polynomials* to share a secret!
- Fix (t, n).
- Take $\mathbb{F}$ such that $|\mathbb{F}| > n$.

Shamir Secret Sharing

- Idea: use *random polynomials* to share a secret!
- Fix (t, n).
- Take $\mathbb{F}$ such that $|\mathbb{F}| > n$.
  - For our purposes, we always set $\mathbb{F} = \mathbb{Z}_p$ for large enough prime p.

Shamir Secret Sharing

- Idea: use *random polynomials* to share a secret!
- Fix (t, n).
- Take $\mathbb{F}$ such that $|\mathbb{F}| > n$.
  - For our purposes, we always set $\mathbb{F} = \mathbb{Z}_p$ for large enough prime p.
- Main Idea: for a secret $s \in \mathbb{F}$, sample random degree $t-1$ polynomial $f \in \mathbb{F}[X]$ such that $f(0) = s$. $f_0 + f_1 X + \ldots + f_{t-1} X^{t-1}$

Shamir Secret Sharing

- Idea: use *random polynomials* to share a secret!
- Fix (t, n).
- Take $\mathbb{F}$ such that $|\mathbb{F}| > n$.
  - For our purposes, we always set $\mathbb{F} = \mathbb{Z}_p$ for large enough prime p.
- Main Idea: for a secret $s \in \mathbb{F}$, sample random degree $t-1$ polynomial $f \in \mathbb{F}[X]$ such that $f(0) = s$.
- Secret shares are $(f(1), \ldots, f(n))$

Shamir Secret Sharing

- Idea: use *random polynomials* to share a secret!
- Fix (t, n).
- Take $\mathbb{F}$ such that $|\mathbb{F}| > n$.
  - For our purposes, we always set $\mathbb{F} = \mathbb{Z}_p$ for large enough prime p.
- Main Idea: for a secret $s \in \mathbb{F}$, sample random degree $t-1$ polynomial $f \in \mathbb{F}[X]$ such that $f(0) = s$.
- Secret shares are $(f(1), \ldots, f(n))$
- Reconstruction is just *Lagrange Interpolation*.

Shamir Secret Sharing

Shamir Secret Sharing

$\Sigma = (\mathsf{Share}, \mathsf{Reconstruct})$

$\mathcal{M} = \mathbb{Z}_p; \mathcal{S} = [n] \times \mathbb{Z}_p$

which party

Shamir Secret Sharing

$\Sigma = (\mathsf{Share}, \mathsf{Reconstruct})$
$\mathcal{M} = \mathbb{Z}_p; \mathcal{S} = [n] \times \mathbb{Z}_p$

$\mathsf{Share}(m)$:
$f(X) = s$
for $i = 1, \ldots, t-1$:
$f_i \xleftarrow{\$} \mathbb{Z}_p$
$f = f + f_i \cdot X^i$
return $[(i, f(i))$ for $i \in [n]]$

Shamir Secret Sharing

$\Sigma = (\mathsf{Share}, \mathsf{Reconstruct})$
$\mathcal{M} = \mathbb{Z}_p; \mathcal{S} = [n] \times \mathbb{Z}_p$

$\mathsf{Share}(m)$:
$f(X) = s$
for $i = 1, \ldots, t-1$:
$f_i \xleftarrow{\$} \mathbb{Z}_p$
$f = f + f_i \cdot X^i$
return $[(i, f(i))$ for $i \in [n]]$

$\mathsf{Reconstruct}(S_{i_1}, \ldots, S_{i_m})$:
Parse $S_{i_j} = (i_j, y_{i_j}) \forall j \in [m]$
$f(X) =$
$\mathsf{Interpolate}(S_{i_1}, \ldots, S_{i_m})$
return $f(0)$

Shamir Secret Sharing: Example

Shamir Secret Sharing: Example

- Set $p = 37$; so we operate over $\mathbb{Z}_{37}$.

Shamir Secret Sharing: Example

- Set $p = 37$; so we operate over $\mathbb{Z}_{37}$.
- Each of you receives one share of a $(3, 13)$ Shamir Secret Sharing.

Shamir Secret Sharing: Example

- Set $p = 37$; so we operate over $\mathbb{Z}_{37}$.
- Each of you receives one share of a $(3, 13)$ Shamir Secret Sharing.
  - Shares are of the form $S_i = (i, f(i))$.

Shamir Secret Sharing: Example

- Set $p = 37$; so we operate over $\mathbb{Z}_{37}$.
- Each of you receives one share of a $(3, 13)$ Shamir Secret Sharing.
  - Shares are of the form $S_i = (i, f(i))$.
- Goal: be first to tell me what the secret is!

Shamir Secret Sharing: Example

- Set $p = 37$; so we operate over $\mathbb{Z}_{37}$.
- Each of you receives one share of a $(3, 13)$ Shamir Secret Sharing.
  - Shares are of the form $S_i = (i, f(i))$.
- Goal: be first to tell me what the secret is!
- Helpful Reminders:

Shamir Secret Sharing: Example

- Set $p = 37$; so we operate over $\mathbb{Z}_{37}$.
- Each of you receives one share of a $(3, 13)$ Shamir Secret Sharing.
  - Shares are of the form $S_i = (i, f(i))$.
- Goal: be first to tell me what the secret is!
- Helpful Reminders:
  - $f(X) = \sum_{i=0}^{t-1} (f_i \cdot X^i \bmod p) \bmod p$

Shamir Secret Sharing: Example

- Set $p = 37$; so we operate over $\mathbb{Z}_{37}$.
- Each of you receives one share of a $(3, 13)$ Shamir Secret Sharing.
  - Shares are of the form $S_i = (i, f(i))$.
- Goal: be first to tell me what the secret is!
- Helpful Reminders:
  - $f(X) = \sum_{i=0}^{t-1} (f_i \cdot X^i \bmod p) \bmod p$
  - Lagrange Interpolation on set $\{(i_j, y_{i_j})\}_{j \in [m]}$ s.t. $i_j \neq i_{j'}$ for all $j \neq j'$:

Shamir Secret Sharing: Example

- Set $p = 37$; so we operate over $\mathbb{Z}_{37}$.
- Each of you receives one share of a $(3, 13)$ Shamir Secret Sharing.
  - Shares are of the form $S_i = (i, f(i))$.
- Goal: be first to tell me what the secret is!
- Helpful Reminders:
  - $f(X) = \sum_{i=0}^{t-1} (f_i \cdot X^i \bmod p) \bmod p$
  - Lagrange Interpolation on set $\{(i_j, y_{i_j})\}_{j \in [m]}$ s.t. $i_j \neq i_{j'}$ for all $j \neq j'$:

$$\delta_k(X) = \prod_{\substack{j \in [m] \\ j \neq k}} \frac{X - i_j}{(i_k - i_j)} \qquad L(X) = \sum_{k=1}^{m} y_{i_k} \cdot \delta_k(X)$$

f(i) m=3 your party # inverse over $\mathbb{Z}_p$

Shamir Secret Sharing: Correctness

Shamir Secret Sharing: Correctness

- Let's prove correctness of Shamir's scheme.

$$\forall\, m \in \mathbb{F}$$

$$\forall\, T \subseteq [n]\,,\ |T| \geq t$$

$$\Pr\left[\ \text{Reconstruct}\left(\{\text{Share}(m)_i\}_{i \in T}\right) = m\ \right]$$

$$= 1$$

Shamir Secret Sharing: Correctness

- Let's prove correctness of Shamir's scheme.

- Need to show that the reconstruction algorithm outputs a *unique* polynomial $f(X)$ of degree $\leq t-1$ such that $f(0) = s$ for secret s.

Shamir Secret Sharing: Correctness

- Let's prove correctness of Shamir's scheme.
- Need to show that the reconstruction algorithm outputs a *unique* polynomial $f(X)$ of degree $t-1$ such that $f(0) = s$ for secret s.
- Useful fact about polynomials over fields:

Shamir Secret Sharing: Correctness

- Let's prove correctness of Shamir's scheme.
- Need to show that the reconstruction algorithm outputs a *unique* polynomial $f(X)$ of degree $t-1$ such that $f(0) = s$ for secret s.
- Useful fact about polynomials over fields:

Lemma 1

Let $\mathbb{F}$ be a (finite) field and let $f \in \mathbb{F}[X]$ be a non-zero polynomial of degree at most t. Then, $f(X)$ has at most t roots (i.e., $|\{\alpha \in \mathbb{F} : f(\alpha) = 0\}| \leqslant t$).

$\mathbb{F}_q \cong \mathbb{F}_p[X]/P(X)$

$q = p^a$

$P(X)$ is irreducible in $\mathbb{F}_p[X]$ and degree $= a$

Shamir Secret Sharing: Correctness

- Let's prove correctness of Shamir's scheme.
- Need to show that the reconstruction algorithm outputs a *unique* polynomial $f(X)$ of degree $t-1$ such that $f(0) = s$ for secret s.
- Useful fact about polynomials over fields:

Lemma 1

Let $\mathbb{F}$ be a (finite) field and let $f \in \mathbb{F}[X]$ be a non-zero polynomial of degree at most t. Then, $f(X)$ has at most t roots (i.e., $|\{\alpha \in \mathbb{F} \colon f(\alpha) = 0\}| \leqslant t$).

Corollary 1

For any (finite) field $\mathbb{F}$, any two distinct degree-t polynomials $p, q \in \mathbb{F}[X]$ agree on at most t points.

Shamir Secret Sharing: Correctness

Shamir Secret Sharing: Correctness

- Correctness of (t, n) Shamir follows from the following lemma.

Shamir Secret Sharing: Correctness

- Correctness of (t, n) Shamir follows from the following lemma.

Lemma 2

Let $x_1, \ldots, x_t, y_1, \ldots, y_t \in \mathbb{F}$ such that x_i are all distinct. Then, there is a unique polynomial $p(X)$ of degree at most $t-1$ such that $p(x_i) = y_i$ for all i.

Shamir Secret Sharing: Correctness

- Correctness of (t, n) Shamir follows from the following lemma.

Lemma 2

Let $x_1, \ldots, x_t, y_1, \ldots, y_t \in \mathbb{F}$ such that x_i are all distinct. Then, there is a unique polynomial $p(X)$ of degree at most $t-1$ such that $p(x_i) = y_i$ for all i.

Proof:

Shamir Secret Sharing: Correctness

- Correctness of (t, n) Shamir follows from the following lemma.

Lemma 2

Let $x_1, \ldots, x_t, y_1, \ldots, y_t \in \mathbb{F}$ such that x_i are all distinct. Then, there is a unique polynomial $p(X)$ of degree at most $t-1$ such that $p(x_i) = y_i$ for all i.

Proof:

- Follows via Lagrange Interpolation!

Shamir Secret Sharing: Correctness

- Correctness of (t, n) Shamir follows from the following lemma.

Lemma 2

Let $x_1, \ldots, x_t, y_1, \ldots, y_t \in \mathbb{F}$ such that x_i are all distinct. Then, there is a unique polynomial $p(X)$ of degree at most $t-1$ such that $p(x_i) = y_i$ for all i.

Proof:

- Follows via Lagrange Interpolation!
- Let $f(X) = \mathsf{Lagrange}((x_i, y_i))$. $= \sum_{i=1}^{t} y_i \delta_i(X)$

$\deg \le t-1$

$$\delta_i(X) = \prod_{j=1, j \neq i}^{t} \frac{(X - x_j)}{(x_i - x_j)}$$

↙ degree $\le (t-1)$

Shamir Secret Sharing: Correctness

- Correctness of (t, n) Shamir follows from the following lemma.

Lemma 2

Let $x_1, \ldots, x_t, y_1, \ldots, y_t \in \mathbb{F}$ such that x_i are all distinct. Then, there is a unique polynomial $p(X)$ of degree at most $t-1$ such that $p(x_i) = y_i$ for all i.

Proof:

- Follows via Lagrange Interpolation!
- Let $f(X) = \mathsf{Lagrange}((x_i, y_i))$.
- Then, $\mathsf{deg}(f) \leqslant t-1$ by definition.

Q: does there $\exists\ g(x)$ s.t.

$g(x_i) = y_i \quad \forall\ i = 1, \ldots, t$?

$\deg(g) \leq t-1$

Shamir Secret Sharing: Correctness

- Correctness of (t, n) Shamir follows from the following lemma.

Lemma 2

Let $x_1, \ldots, x_t, y_1, \ldots, y_t \in \mathbb{F}$ such that x_i are all distinct. Then, there is a unique polynomial $p(X)$ of degree at most $t-1$ such that $p(x_i) = y_i$ for all i.

Proof:

- Follows via Lagrange Interpolation!
- Let $f(X) = \mathsf{Lagrange}((x_i, y_i))$.
- Then, $\mathsf{deg}(f) \leqslant t - 1$ by definition.
- Uniqueness follows via Lemma 1! □

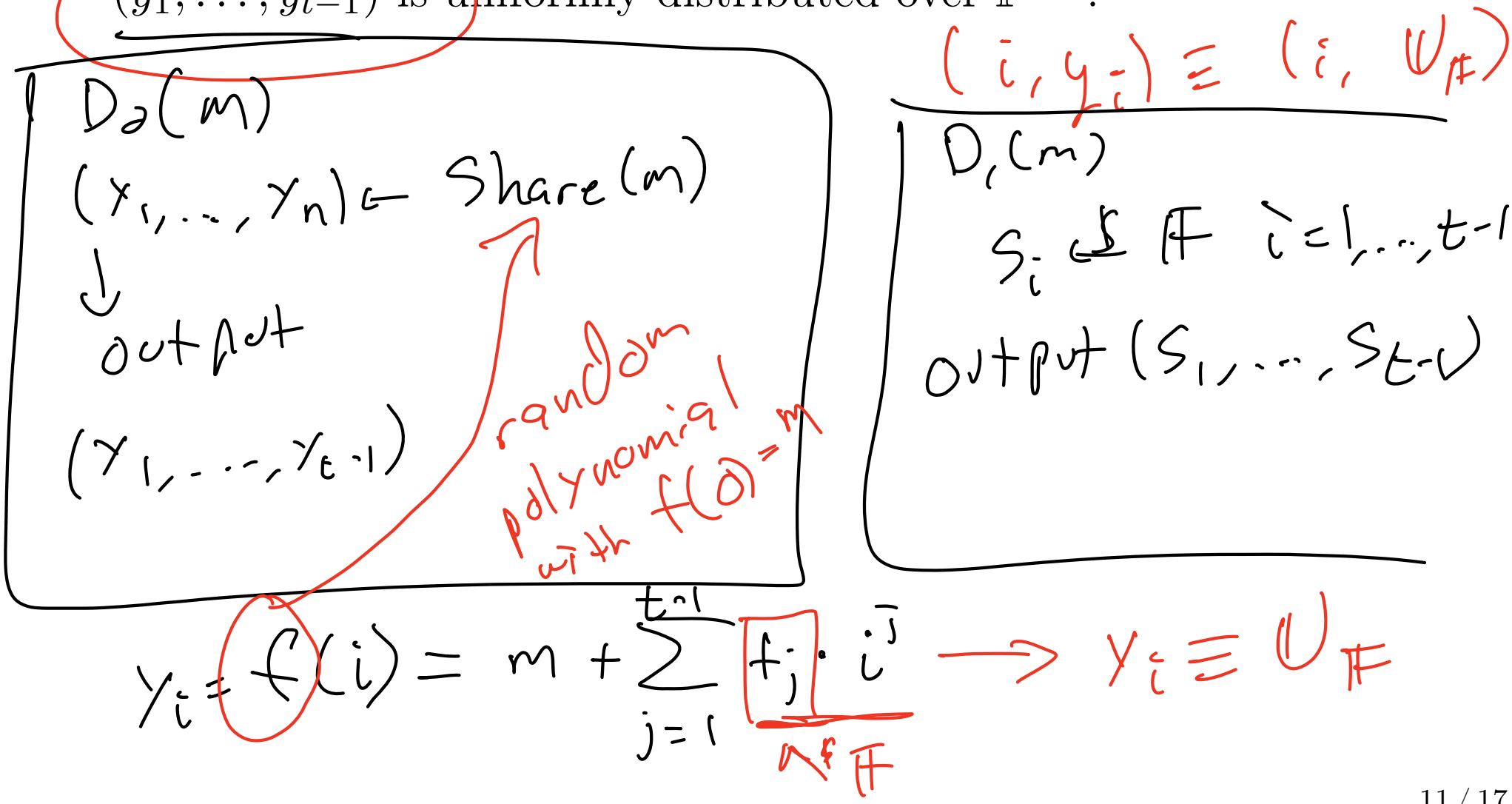
Shamir Secret Sharing: Security

Shamir Secret Sharing: Security

- Need to show that any $t-1$ users "learn nothing" about the secret s.

Shamir Secret Sharing: Security

- Need to show that any $t-1$ users "learn nothing" about the secret s.
- Suffices to consider $\{(1, y_1), \ldots, (t-1, y_{t-1})\}$ and show that $(y_1, \ldots, y_{t-1})$ is uniformly distributed over $\mathbb{F}^{t-1}$.



Shamir Secret Sharing: Efficiency

Shamir Secret Sharing: Efficiency

$\Sigma = (\mathsf{Share}, \mathsf{Reconstruct})$
$\mathcal{M} = \mathbb{Z}_p; \mathcal{S} = [n] \times \mathbb{Z}_p$

$\mathsf{Share}(m)$:
$f(X) = s$
for $i = 1, \ldots, t-1$:
$f_i \xleftarrow{\$} \mathbb{Z}_p$
$f = f + f_i \cdot X^i$
return $[(i, f(i))$ for $i \in [n]]$

$\mathsf{Reconstruct}(S_{i_1}, \ldots, S_{i_m})$:
Parse $S_{i_j} = (i_j, y_{i_j}) \forall j \in [m]$
$f(X) =$
$\mathsf{Interpolate}(S_{i_1}, \ldots, S_{i_m})$
return $f(0)$

O(t) field ops

Space: Every party stores $(i, f(i)) \in [n] \times \mathbb{F}$
↳ has size $O(\log(n) + \log|\mathbb{F}|)$ bits
$= \in_{\mathbb{F}} (1)$

Time

Shamir Secret Sharing: Efficiency

Shamir Secret Sharing: Efficiency

- Share Size

Shamir Secret Sharing: Efficiency

- Share Size
  - One $\mathbb{F}$ element per party, which is $\Theta(1)$ with respect to fixed $\mathbb{F}$

Shamir Secret Sharing: Efficiency

- Share Size
  - One $\mathbb{F}$ element per party, which is $\Theta(1)$ with respect to fixed $\mathbb{F}$
- Time of $\mathsf{Share}(m)$

Shamir Secret Sharing: Efficiency

- Share Size
  - One $\mathbb{F}$ element per party, which is $\Theta(1)$ with respect to fixed $\mathbb{F}$
- Time of $\mathsf{Share}(m)$
  - Samples $t-1$ random $\mathbb{F}$ elements as coefficients; $O(t)$ time

Shamir Secret Sharing: Efficiency

- Share Size
  - One $\mathbb{F}$ element per party, which is $\Theta(1)$ with respect to fixed $\mathbb{F}$
- Time of $\mathsf{Share}(m)$
  - Samples $t-1$ random $\mathbb{F}$ elements as coefficients; $O(t)$ time
  - Computes $(f(1), \ldots, f(n))$

Shamir Secret Sharing: Efficiency

- Share Size
  - One $\mathbb{F}$ element per party, which is $\Theta(1)$ with respect to fixed $\mathbb{F}$
- Time of $\mathsf{Share}(m)$
  - Samples $t-1$ random $\mathbb{F}$ elements as coefficients; $O(t)$ time
  - Computes $(f(1), \ldots, f(n))$
    - Naïve implementation: $f(i) = \langle (f_0, \ldots, f_{t-1}), (1, i, i^2, \ldots, i^{t-1}) \rangle$ for every i; $O(n \cdot t)$ field operations to compute $(f(1), \ldots, f(n))$

Shamir Secret Sharing: Efficiency

- Share Size
  - One $\mathbb{F}$ element per party, which is $\Theta(1)$ with respect to fixed $\mathbb{F}$
- Time of $\mathsf{Share}(m)$
  - Samples $t-1$ random $\mathbb{F}$ elements as coefficients; $O(t)$ time
  - Computes $(f(1), \ldots, f(n))$
    - Naïve implementation: $f(i) = \langle (f_0, \ldots, f_{t-1}), (1, i, i^2, \ldots, i^{t-1}) \rangle$ for every i; $O(n \cdot t)$ field operations to compute $(f(1), \ldots, f(n))$
    - *Fast Fourier Transform*: computes $(f(1), \ldots, f(n))$ in $O(n \log(t))$ field operations[1]

Shamir Secret Sharing: Efficiency

- Share Size
  - One $\mathbb{F}$ element per party, which is $\Theta(1)$ with respect to fixed $\mathbb{F}$
- Time of $\mathsf{Share}(m)$
  - Samples $t-1$ random $\mathbb{F}$ elements as coefficients; $O(t)$ time
  - Computes $(f(1), \ldots, f(n))$
    - Naïve implementation: $f(i) = \langle (f_0, \ldots, f_{t-1}), (1, i, i^2, \ldots, i^{t-1}) \rangle$ for every i; $O(n \cdot t)$ field operations to compute $(f(1), \ldots, f(n))$
    - *Fast Fourier Transform*: computes $(f(1), \ldots, f(n))$ in $O(n \log(t))$ field operations[1]

[1] Actually, we need some special structure.

Shamir Secret Sharing: Efficiency

- Share Size
  - One $\mathbb{F}$ element per party, which is $\Theta(1)$ with respect to fixed $\mathbb{F}$
- Time of $\mathsf{Share}(m)$
  - Samples $t-1$ random $\mathbb{F}$ elements as coefficients; $O(t)$ time
  - Computes $(f(1), \ldots, f(n))$
    - Naïve implementation: $f(i) = \langle (f_0, \ldots, f_{t-1}), (1, i, i^2, \ldots, i^{t-1}) \rangle$ for every i; $O(n \cdot t)$ field operations to compute $(f(1), \ldots, f(n))$
    - *Fast Fourier Transform*: computes $(f(1), \ldots, f(n))$ in $O(n \log(t))$ field operations[1]
- Time of $\mathsf{Reconstruct}$

[1] Actually, we need some special structure.

Shamir Secret Sharing: Efficiency

- Share Size
  - One $\mathbb{F}$ element per party, which is $\Theta(1)$ with respect to fixed $\mathbb{F}$
- Time of Share(m)
  - Samples $t-1$ random $\mathbb{F}$ elements as coefficients; $O(t)$ time
  - Computes $(f(1), \ldots, f(n))$
    - Naïve implementation: $f(i) = \langle (f_0, \ldots, f_{t-1}), (1, i, i^2, \ldots, i^{t-1}) \rangle$ for every i; $O(n \cdot t)$ field operations to compute $(f(1), \ldots, f(n))$
    - *Fast Fourier Transform*: computes $(f(1), \ldots, f(n))$ in $O(n \log(t))$ field operations[1]
- Time of Reconstruct
  - Lagrange Interpolation

[1] Actually, we need some special structure.

Shamir Secret Sharing: Efficiency

- Share Size
  - One $\mathbb{F}$ element per party, which is $\Theta(1)$ with respect to fixed $\mathbb{F}$
- Time of $\mathsf{Share}(m)$
  - Samples $t-1$ random $\mathbb{F}$ elements as coefficients; $O(t)$ time
  - Computes $(f(1), \ldots, f(n))$
    - Naïve implementation: $f(i) = \langle (f_0, \ldots, f_{t-1}), (1, i, i^2, \ldots, i^{t-1}) \rangle$ for every i; $O(n \cdot t)$ field operations to compute $(f(1), \ldots, f(n))$
    - *Fast Fourier Transform*: computes $(f(1), \ldots, f(n))$ in $O(n \log(t))$ field operations[1]
- Time of $\mathsf{Reconstruct}$
  - Lagrange Interpolation
  - Naïve implementation: $O(nt)$ field operations

[1] Actually, we need some special structure.

Shamir Secret Sharing: Efficiency

- Share Size
  - One $\mathbb{F}$ element per party, which is $\Theta(1)$ with respect to fixed $\mathbb{F}$
- Time of $\mathsf{Share}(m)$
  - Samples $t-1$ random $\mathbb{F}$ elements as coefficients; $O(t)$ time
  - Computes $(f(1), \ldots, f(n))$
    - Naïve implementation: $f(i) = \langle (f_0, \ldots, f_{t-1}), (1, i, i^2, \ldots, i^{t-1}) \rangle$ for every i; $O(n \cdot t)$ field operations to compute $(f(1), \ldots, f(n))$
    - *Fast Fourier Transform*: computes $(f(1), \ldots, f(n))$ in $O(n \log(t))$ field operations[1]
- Time of $\mathsf{Reconstruct}$
  - Lagrange Interpolation
  - Naïve implementation: $O(nt)$ field operations
  - Inverse FFTs: $O(n \log(t))$ field operations

[1] Actually, we need some special structure.

Efficient Shamir via Fast Fourier Transform

Efficient Shamir via Fast Fourier Transform

- The famous *Fast Fourier Transform* (drastically) improves the efficiency of Shamir Secret Sharing: from $O(nt)$ field operations to $O(n \log(t))$ for both Sharing and Reconstruction.

Efficient Shamir via Fast Fourier Transform

- The famous *Fast Fourier Transform* (drastically) improves the efficiency of Shamir Secret Sharing: from $O(nt)$ field operations to $O(n \log(t))$ for both Sharing and Reconstruction.

- However, to get the most out of it, we need *specific structure* to get these efficiency gains.

Efficient Shamir via Fast Fourier Transform

- The famous *Fast Fourier Transform* (drastically) improves the efficiency of Shamir Secret Sharing: from $O(nt)$ field operations to $O(n \log(t))$ for both Sharing and Reconstruction.

- However, to get the most out of it, we need *specific structure* to get these efficiency gains.
  1. $p = q \cdot n + 1$ for some $q \in \mathbb{N}$, and $n = 2^N$.

Efficient Shamir via Fast Fourier Transform

- The famous *Fast Fourier Transform* (drastically) improves the efficiency of Shamir Secret Sharing: from $O(nt)$ field operations to $O(n \log(t))$ for both Sharing and Reconstruction.

- However, to get the most out of it, we need *specific structure* to get these efficiency gains.
  1. $p = q \cdot n + 1$ for some $q \in \mathbb{N}$, and $n = 2^N$.
  2. Shares are of the form $(i, \omega^i, f(\omega^i))$, where

Efficient Shamir via Fast Fourier Transform

- The famous *Fast Fourier Transform* (drastically) improves the efficiency of Shamir Secret Sharing: from $O(nt)$ field operations to $O(n \log(t))$ for both Sharing and Reconstruction.

- However, to get the most out of it, we need *specific structure* to get these efficiency gains.
  1. $p = q \cdot n + 1$ for some $q \in \mathbb{N}$, and $n = 2^N$.
  2. Shares are of the form $(i, \omega^i, f(\omega^i))$, where
     - f is the polynomial sampled by the share algorithm; and

Efficient Shamir via Fast Fourier Transform

- The famous *Fast Fourier Transform* (drastically) improves the efficiency of Shamir Secret Sharing: from $O(nt)$ field operations to $O(n \log(t))$ for both Sharing and Reconstruction.

- However, to get the most out of it, we need *specific structure* to get these efficiency gains.
  1. $p = q \cdot n + 1$ for some $q \in \mathbb{N}$, and $n = 2^N$.
  2. Shares are of the form $(i, \omega^i, f(\omega^i))$, where
     - f is the polynomial sampled by the share algorithm; and
     - ω is a *n-th primitive root of unity*.

Efficient Shamir via Fast Fourier Transform

- The famous *Fast Fourier Transform* (drastically) improves the efficiency of Shamir Secret Sharing: from $O(nt)$ field operations to $O(n \log(t))$ for both Sharing and Reconstruction.

- However, to get the most out of it, we need *specific structure* to get these efficiency gains.
  1. $p = q \cdot n + 1$ for some $q \in \mathbb{N}$, and $n = 2^N$.
  2. Shares are of the form $(i, \omega^i, f(\omega^i))$, where
     - f is the polynomial sampled by the share algorithm; and
     - ω is a *n-th primitive root of unity*.

- Given this setup, we can perform Shamir Secret Sharing in $O(n \log(t))$ time!

Linear Secret Sharing

Linear Secret Sharing

- Shamir Secret Sharing is an example of *linear secret sharing*.

Linear Secret Sharing

- Shamir Secret Sharing is an example of *linear secret sharing*.

Definition 1 (Linear Secret Sharing)

A (t, n) secret-sharing scheme $\Sigma = (\mathsf{Share}, \mathsf{Reconstruct})$ is a *linear secret-sharing scheme* if $\mathsf{Share}, \mathsf{Reconstruct}$ are linear maps.

Linear Secret Sharing

- Shamir Secret Sharing is an example of *linear secret sharing*.

Definition 1 (Linear Secret Sharing)

A (t, n) secret-sharing scheme $\Sigma = (\mathsf{Share}, \mathsf{Reconstruct})$ is a *linear secret-sharing scheme* if $\mathsf{Share}, \mathsf{Reconstruct}$ are linear maps.

- What does this mean?

Linear Secret Sharing

- Shamir Secret Sharing is an example of *linear secret sharing*.

Definition 1 (Linear Secret Sharing)

A (t, n) secret-sharing scheme $\Sigma = (\mathsf{Share}, \mathsf{Reconstruct})$ is a *linear secret-sharing scheme* if $\mathsf{Share}, \mathsf{Reconstruct}$ are linear maps.

- What does this mean?
- Let $m \in \mathbb{F}$ and $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(m)$.

Linear Secret Sharing

- Shamir Secret Sharing is an example of *linear secret sharing*.

Definition 1 (Linear Secret Sharing)

A (t, n) secret-sharing scheme $\Sigma = (\mathsf{Share}, \mathsf{Reconstruct})$ is a *linear secret-sharing scheme* if $\mathsf{Share}, \mathsf{Reconstruct}$ are linear maps.

- What does this mean?
- Let $m \in \mathbb{F}$ and $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(m)$.
- For $\alpha, \beta \in \mathbb{F}$, $(\alpha \cdot s_i + \beta)_{i \in [n]}$ is a valid secret sharing of $\alpha \cdot m + \beta$

Linear Secret Sharing

- Shamir Secret Sharing is an example of *linear secret sharing*.

Definition 1 (Linear Secret Sharing)

A (t, n) secret-sharing scheme $\Sigma = (\mathsf{Share}, \mathsf{Reconstruct})$ is a *linear secret-sharing scheme* if $\mathsf{Share}, \mathsf{Reconstruct}$ are linear maps.

- What does this mean?
- Let $m \in \mathbb{F}$ and $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(m)$.
- For $\alpha, \beta \in \mathbb{F}$, $(\alpha \cdot s_i + \beta)_{i \in [n]}$ is a valid secret sharing of $\alpha \cdot m + \beta$
- Additionally, for $m' \in \mathbb{F}$ and $(s'_1, \ldots, s'_n) \leftarrow \mathsf{Share}(m')$, $(s_i + s'_i)_{i \in [n]}$ is a valid secret sharing of $m + m'$.

Example with our Shares

Example with our Shares

- Confirm with our in-class example.

Example with our Shares

- Confirm with our in-class example.
- Transform all your shares $s_i \mapsto 5 \cdot s_i + 1 \bmod p$ and perform reconstruction.

Example with our Shares

- Confirm with our in-class example.
- Transform all your shares $s_i \mapsto 5 \cdot s_i + 1 \bmod p$ and perform reconstruction.
- Another example from Sage terminal!

Next Time: Blakley Secret Sharing, General Linear Secret Sharing, and Verifiable Secret Sharing