Secret Sharing: 2 out of N and Beyond

Luke Kowalczyk

September, 13, 2016

Review

- What is secret sharing?
- 2 out of 2 secret sharing

2 out of n secret sharing from 2 out of 2 secret sharing

- Proof by reduction (started last class)

Some Number Theory

t out of n secret sharing (Shamir)

t-out-of-n Secret Sharing Syntax and Correctness

A *t-out-of-n secret sharing scheme* over message space $\mathcal{M}$ is a pair of algorithms $(\mathtt{Share}, \mathtt{Reconstruct})$ such that:

t-out-of-n Secret Sharing Syntax and Correctness

A *t-out-of-n secret sharing scheme* over message space $\mathcal{M}$ is a pair of algorithms $(\texttt{Share}, \texttt{Reconstruct})$ such that:

- $\texttt{Share}$ is a randomized algorithm that on any input $m \in \mathcal{M}$ outputs a n-tuple of shares $(s_1, \ldots, s_n)$.

t-out-of-n Secret Sharing Syntax and Correctness

A *t-out-of-n secret sharing scheme* over message space $\mathcal{M}$ is a pair of algorithms $(\texttt{Share}, \texttt{Reconstruct})$ such that:

- $\texttt{Share}$ is a randomized algorithm that on any input $m \in \mathcal{M}$ outputs a n-tuple of shares $(s_1, \ldots, s_n)$.
- $\texttt{Reconstruct}$ is a deterministic algorithm that given an t-tuple of shares outputs a message in $\mathcal{M}$

t-out-of-n Secret Sharing Syntax and Correctness

A *t-out-of-n secret sharing scheme* over message space $\mathcal{M}$ is a pair of algorithms $(\texttt{Share}, \texttt{Reconstruct})$ such that:

- $\texttt{Share}$ is a randomized algorithm that on any input $m \in \mathcal{M}$ outputs a n-tuple of shares $(s_1, \ldots, s_n)$.
- $\texttt{Reconstruct}$ is a deterministic algorithm that given an t-tuple of shares outputs a message in $\mathcal{M}$

while both satisfy the following *correctness* requirement:

t-out-of-n Secret Sharing Syntax and Correctness

A *t-out-of-n secret sharing scheme* over message space $\mathcal{M}$ is a pair of algorithms $(\texttt{Share}, \texttt{Reconstruct})$ such that:

- $\texttt{Share}$ is a randomized algorithm that on any input $m \in \mathcal{M}$ outputs a n-tuple of shares $(s_1, \ldots, s_n)$.
- $\texttt{Reconstruct}$ is a deterministic algorithm that given an t-tuple of shares outputs a message in $\mathcal{M}$

while both satisfy the following *correctness* requirement:
$\forall m \in \mathcal{M}$,

t-out-of-n Secret Sharing Syntax and Correctness

A *t-out-of-n secret sharing scheme* over message space $\mathcal{M}$ is a pair of algorithms $(\texttt{Share}, \texttt{Reconstruct})$ such that:

- $\texttt{Share}$ is a randomized algorithm that on any input $m \in \mathcal{M}$ outputs a n-tuple of shares $(s_1, \ldots, s_n)$.
- $\texttt{Reconstruct}$ is a deterministic algorithm that given an t-tuple of shares outputs a message in $\mathcal{M}$

while both satisfy the following *correctness* requirement:
$\forall m \in \mathcal{M}$, $\forall S = \{i_1, \ldots, i_t\} \subseteq \{1, \ldots, n\}$ of size t,

t-out-of-n Secret Sharing Syntax and Correctness

A *t-out-of-n secret sharing scheme* over message space $\mathcal{M}$ is a pair of algorithms $(\mathtt{Share}, \mathtt{Reconstruct})$ such that:

- $\mathtt{Share}$ is a randomized algorithm that on any input $m \in \mathcal{M}$ outputs a n-tuple of shares $(s_1, \ldots, s_n)$.
- $\mathtt{Reconstruct}$ is a deterministic algorithm that given an t-tuple of shares outputs a message in $\mathcal{M}$

while both satisfy the following *correctness* requirement:
$\forall m \in \mathcal{M}$, $\forall S = \{i_1, \ldots, i_t\} \subseteq \{1, \ldots, n\}$ of size t,

$$\Pr_{\mathtt{Share}(m) \to (s_1, \ldots, s_n)} [\mathtt{Reconstruct}(s_{i_1}, \ldots, s_{i_t}) = m] = 1$$

t-out-of-n Secret Sharing Security

Definition (secret sharing security with adversaries)

A t-out-of-n secret sharing scheme $(\texttt{Share}, \texttt{Reconstruct})$ over M is perfectly secure if:

t-out-of-n Secret Sharing Security

Definition (secret sharing security with adversaries)

A t-out-of-n secret sharing scheme $(\mathtt{Share}, \mathtt{Reconstruct})$ over M is perfectly secure if:
$\forall m, m' \in \mathcal{M}$,

t-out-of-n Secret Sharing Security

Definition (secret sharing security with adversaries)

A t-out-of-n secret sharing scheme $(\mathtt{Share}, \mathtt{Reconstruct})$ over M is perfectly secure if:

$\forall m, m' \in \mathcal{M}, \;\; \forall S \subseteq \{1, \ldots, n\}$ s.t. $|S| < t$,

t-out-of-n Secret Sharing Security

Definition (secret sharing security with adversaries)

A t-out-of-n secret sharing scheme $(\texttt{Share}, \texttt{Reconstruct})$ over M is perfectly secure if:

$\forall m, m' \in \mathcal{M},\ \ \forall S \subseteq \{1, \ldots, n\}$ s.t. $|S| < t,\ \ \forall A,$

t-out-of-n Secret Sharing Security

Definition (secret sharing security with adversaries)

A t-out-of-n secret sharing scheme $(\mathtt{Share}, \mathtt{Reconstruct})$ over M is perfectly secure if:

$\forall m, m' \in \mathcal{M}, \ \ \forall S \subseteq \{1, \ldots, n\}$ s.t. $|S| < t, \ \ \forall A,$

$$\Pr_{\substack{\mathtt{Share}(m) \\ \to (s_1, \ldots, s_n)}} [A((s_i | i \in S)) = 1] = \Pr_{\substack{\mathtt{Share}(m') \\ \to (s'_1, \ldots, s'_n)}} [A((s'_i | i \in S)) = 1]$$

A 2-out-of-2 Secret Sharing Scheme

A 2-out-of-2 Secret Sharing Scheme

$\texttt{Share}_{2-2}$: On input $m \in \{0,1\}^\ell$,

- select $s_0 \in \{0,1\}^\ell$ uniformly at random.
- set $s_1 = s_0 \oplus m$
- output (s_0, s_1)

A 2-out-of-2 Secret Sharing Scheme

$\texttt{Share}_{2-2}$: On input $m \in \{0,1\}^\ell$,

- select $s_0 \in \{0,1\}^\ell$ uniformly at random.
- set $s_1 = s_0 \oplus m$
- output (s_0, s_1)

$\texttt{Reconstruct}_{2-2}$: On input $(s_0, s_1) \in \{0,1\}^\ell \times \{0,1\}^\ell$,

- output $s_0 \oplus s_1$

A 2-out-of-2 Secret Sharing Scheme

$\texttt{Share}_{2-2}$: On input $m \in \{0,1\}^\ell$,

- select $s_0 \in \{0,1\}^\ell$ uniformly at random.
- set $s_1 = s_0 \oplus m$
- output (s_0, s_1)

$\texttt{Reconstruct}_{2-2}$: On input $(s_0, s_1) \in \{0,1\}^\ell \times \{0,1\}^\ell$,

- output $s_0 \oplus s_1$

(proved perfect security using identical distributions security definition)

A 2-out-of-n Secret Sharing Scheme

$\texttt{Share}_{2-n}$: On input $m \in \{0,1\}^\ell$,

A 2-out-of-n Secret Sharing Scheme

$\texttt{Share}_{2-n}$: On input $m \in \{0,1\}^{\ell}$,

- For $k = 1$ to $\log n$

A 2-out-of-n Secret Sharing Scheme

$\texttt{Share}_{2-n}$: On input $m \in \{0,1\}^{\ell}$,

- For $k = 1$ to $\log n$
  - Run $\texttt{Share}_{2-2}(m) \rightarrow (s_0^k, s_1^k)$

A 2-out-of-n Secret Sharing Scheme

$\texttt{Share}_{2-n}$: On input $m \in \{0,1\}^\ell$,

- For $k = 1$ to $\log n$
  - Run $\texttt{Share}_{2-2}(m) \to (s_0^k, s_1^k)$
- For $i = 0$ to $n-1$, if binary representation of i is $i_1 \ldots i_{\log n}$, set $S_i = (i, s_{i_1}^1, \ldots, s_{i_{\log n}}^{\log n})$

A 2-out-of-n Secret Sharing Scheme

$\texttt{Share}_{2-n}$: On input $m \in \{0,1\}^\ell$,

- For $k = 1$ to $\log n$
  - Run $\texttt{Share}_{2-2}(m) \to (s_0^k, s_1^k)$
- For $i = 0$ to $n-1$, if binary representation of i is $i_1 \ldots i_{\log n}$, set $S_i = (i, s_{i_1}^1, \ldots, s_{i_{\log n}}^{\log n})$
- Output $(S_0, \ldots, S_{n-1})$.

A 2-out-of-n Secret Sharing Scheme

$\texttt{Share}_{2-n}$: On input $m \in \{0,1\}^\ell$,

- For $k = 1$ to $\log n$
  - Run $\texttt{Share}_{2-2}(m) \to (s_0^k, s_1^k)$
- For $i = 0$ to $n-1$, if binary representation of i is $i_1 \ldots i_{\log n}$, set $S_i = (i, s_{i_1}^1, \ldots, s_{i_{\log n}}^{\log n})$
- Output $(S_0, \ldots, S_{n-1})$.

$\texttt{Reconstruct}_{2-n}$: On input (S_i, S_j),

A 2-out-of-n Secret Sharing Scheme

$\texttt{Share}_{2-n}$: On input $m \in \{0,1\}^\ell$,

- For $k = 1$ to $\log n$
  - Run $\texttt{Share}_{2-2}(m) \to (s_0^k, s_1^k)$
- For $i = 0$ to $n-1$, if binary representation of i is $i_1 \dots i_{\log n}$, set $S_i = (i, s_{i_1}^1, \dots, s_{i_{\log n}}^{\log n})$
- Output $(S_0, \dots, S_{n-1})$.

$\texttt{Reconstruct}_{2-n}$: On input (S_i, S_j),

- Consider the binary representations of the indices $i = i_1 \dots i_{\log n}$ and $j = j_1 \dots j_{\log n}$.

A 2-out-of-n Secret Sharing Scheme

$\texttt{Share}_{2-n}$: On input $m \in \{0,1\}^\ell$,

- For $k = 1$ to $\log n$
  - Run $\texttt{Share}_{2-2}(m) \to (s_0^k, s_1^k)$
- For $i = 0$ to $n-1$, if binary representation of i is $i_1 \ldots i_{\log n}$, set $S_i = (i, s_{i_1}^1, \ldots, s_{i_{\log n}}^{\log n})$
- Output $(S_0, \ldots, S_{n-1})$.

$\texttt{Reconstruct}_{2-n}$: On input (S_i, S_j),

- Consider the binary representations of the indices $i = i_1 \ldots i_{\log n}$ and $j = j_1 \ldots j_{\log n}$.
- Find a bit position k where they differ, namely $i_k \neq j_k$ (thus, $s_{i_k}^k = s_0^k, s_{j_k}^k = s_1^k$ or vice versa).

A 2-out-of-n Secret Sharing Scheme

$\texttt{Share}_{2-n}$: On input $m \in \{0,1\}^\ell$,

- For $k = 1$ to $\log n$
  - Run $\texttt{Share}_{2-2}(m) \to (s_0^k, s_1^k)$
- For $i = 0$ to $n-1$, if binary representation of i is $i_1 \dots i_{\log n}$, set $S_i = (i, s_{i_1}^1, \dots, s_{i_{\log n}}^{\log n})$
- Output $(S_0, \dots, S_{n-1})$.

$\texttt{Reconstruct}_{2-n}$: On input (S_i, S_j),

- Consider the binary representations of the indices $i = i_1 \dots i_{\log n}$ and $j = j_1 \dots j_{\log n}$.
- Find a bit position k where they differ, namely $i_k \neq j_k$ (thus, $s_{i_k}^k = s_0^k, s_{j_k}^k = s_1^k$ or vice versa).
- Run $\texttt{Reconstruct}_{2-2}(s_0^k, s_1^k)$ and output the same.

2-out-of-n Scheme: Hybrid Proof

Main idea:
Assume 2-out-of-n scheme is *not* perfectly secure. We will show that this implies that the 2-out-of-2 scheme must not be perfectly secure.

2-out-of-n Scheme: Hybrid Proof

Main idea:
Assume 2-out-of-n scheme is *not* perfectly secure. We will show that this implies that the 2-out-of-2 scheme must not be perfectly secure.

We know that the 2-out-of-two scheme *is* perfectly secure (proved last class). So, this means that our assumption must have been false and it must be the case that the 2-out-of-n scheme *is* perfectly secure.

2-out-of-n Scheme: Hybrid Proof

Assume that the 2-out-of-n scheme above is not secure.

2-out-of-n Scheme: Hybrid Proof

Assume that the 2-out-of-n scheme above is not secure. This means (by negation of the security definition) that:

2-out-of-n Scheme: Hybrid Proof

Assume that the 2-out-of-n scheme above is not secure. This means (by negation of the security definition) that:
$\exists m, m' \in \{0,1\}^\ell$,

2-out-of-n Scheme: Hybrid Proof

Assume that the 2-out-of-n scheme above is not secure. This means (by negation of the security definition) that:
$\exists m, m' \in \{0,1\}^\ell$, $\exists i \in \{0,\ldots,n-1\}$,

2-out-of-n Scheme: Hybrid Proof

Assume that the 2-out-of-n scheme above is not secure. This means (by negation of the security definition) that:
$\exists m, m' \in \{0,1\}^{\ell}$, $\exists i \in \{0, \ldots, n-1\}$, $\exists A$,

2-out-of-n Scheme: Hybrid Proof

Assume that the 2-out-of-n scheme above is not secure. This means (by negation of the security definition) that:
$\exists m, m' \in \{0,1\}^\ell$, $\exists i \in \{0, \ldots, n-1\}$, $\exists A$, such that

$$\Pr_{\texttt{Share}_{2-n}(m) \to (S_1,\ldots,S_n)}[A(S_i) = 1] \neq \Pr_{\texttt{Share}_{2-n}(m') \to (S'_1,\ldots,S'_n)}[A(S'_i) = 1]$$

Our goal is to use this to construct an adversary B that breaks the 2-out-of-2 scheme.

2-out-of-n Scheme: Hybrid Proof

Assume that the 2-out-of-n scheme above is not secure. This means (by negation of the security definition) that:
$\exists m, m' \in \{0,1\}^{\ell}$, $\exists i \in \{0, \ldots, n-1\}$, $\exists A$, such that

$$\Pr_{\mathtt{Share}_{2-n}(m) \to (S_1, \ldots, S_n)}[A(S_i) = 1] \neq \Pr_{\mathtt{Share}_{2-n}(m') \to (S'_1, \ldots, S'_n)}[A(S'_i) = 1]$$

Our goal is to use this to construct an adversary B that breaks the 2-out-of-2 scheme.

Notice we have two distributions (a subset of the outputs of `Share` called on m vs m') such that when A is called on one it outputs 1 with a different probability than when it's called on the other.

2-out-of-n Scheme: Hybrid Proof

In the lefthandside distribution we have $S_i = (s^1_{i_1}, \ldots, s^{\log n}_{i_{\log n}})$, where each (s^k_0, s^k_1) is the output of $\texttt{Share}_{2-2}(m)$.

2-out-of-n Scheme: Hybrid Proof

In the lefthandside distribution we have $S_i = (s^1_{i_1}, \ldots, s^{\log n}_{i_{\log n}})$, where each (s^k_0, s^k_1) is the output of $\mathtt{Share}_{2-2}(m)$.
Let's call this distribution H^0.

2-out-of-n Scheme: Hybrid Proof

In the lefthandside distribution we have $S_i = (s^1_{i_1}, \ldots, s^{\log n}_{i_{\log n}})$, where each (s^k_0, s^k_1) is the output of $\texttt{Share}_{2-2}(m)$.
Let's call this distribution H^0.

In the righthandside distribution, we have $S'_i = (s'^1_{i_1}, \ldots, s'^{\log n}_{i_{\log n}})$, where each (s'^k_0, s'^k_1) is the output of $\texttt{Share}_{2-2}(m')$.

2-out-of-n Scheme: Hybrid Proof

In the lefthandside distribution we have $S_i = (s^1_{i_1}, \ldots, s^{\log n}_{i_{\log n}})$, where each (s^k_0, s^k_1) is the output of $\texttt{Share}_{2-2}(m)$.
Let's call this distribution H^0.

In the righthandside distribution, we have $S'_i = (s'^1_{i_1}, \ldots, s'^{\log n}_{i_{\log n}})$, where each (s'^k_0, s'^k_1) is the output of $\texttt{Share}_{2-2}(m')$.
Let's call this distribution $H^{\log n}$

2-out-of-n Scheme: Hybrid Proof

In the lefthandside distribution we have $S_i = (s^1_{i_1}, \ldots, s^{\log n}_{i_{\log n}})$, where each (s^k_0, s^k_1) is the output of $\mathtt{Share}_{2-2}(m)$.
Let's call this distribution H^0.

In the righthandside distribution, we have $S'_i = (s'^1_{i_1}, \ldots, s'^{\log n}_{i_{\log n}})$, where each (s'^k_0, s'^k_1) is the output of $\mathtt{Share}_{2-2}(m')$.
Let's call this distribution $H^{\log n}$

Using this notation, the previous statement that our scheme is not perfectly secure can be written as:
$\Pr[A(H^0) = 1] \neq \Pr[A(H^{\log n}) = 1]$.

2-out-of-n Scheme: Hybrid Proof

Let's define some more distributions H^j that A *could* be called on (we call these *hybrid distributions*).

2-out-of-n Scheme: Hybrid Proof

Let's define some more distributions H^j that A *could* be called on (we call these *hybrid distributions*).

H^j: for each share, the first j components are taken from m', while the rest are taken from m.

2-out-of-n Scheme: Hybrid Proof

Let's define some more distributions H^j that A *could* be called on (we call these *hybrid distributions*).

H^j: for each share, the first j components are taken from m', while the rest are taken from m.

That is, for every $j \in \{0, \ldots, \log n\}$, we define

$$H^j = \{(s_{i_1}^{\prime 1}, \ldots, s_{i_j}^{\prime j}, s_{i_{j+1}}^{j+1}, \ldots, s_{i_{\log n}}^{\log n}) : \substack{(s_0^k, s_1^k) \leftarrow \mathtt{Share}_{2-2}^k(m) \\ (s_0^{\prime k}, s_1^{\prime k}) \leftarrow \mathtt{Share}_{2-2}^k(m')} \ \forall k\}$$

2-out-of-n Scheme: Hybrid Proof

Let's define some more distributions H^j that A *could* be called on (we call these *hybrid distributions*).

H^j: for each share, the first j components are taken from m', while the rest are taken from m.

That is, for every $j \in \{0, \ldots, \log n\}$, we define

$$H^j = \{(s_{i_1}^{\prime 1}, \ldots, s_{i_j}^{\prime j}, s_{i_{j+1}}^{j+1}, \ldots, s_{i_{\log n}}^{\log n}) : \substack{(s_0^k, s_1^k) \leftarrow \mathtt{Share}_{2-2}^k(m) \\ (s_0^{\prime k}, s_1^{\prime k}) \leftarrow \mathtt{Share}_{2-2}^k(m')} \ \forall k\}$$

Note that our names for H^0 and $H^{\log n}$ match this definition.

2-out-of-n Scheme: Hybrid Proof

Assuming our scheme is not perfectly secure, we know:
$\Pr[A(H^0) = 1] \neq \Pr[A(H^{\log n}) = 1]$.

2-out-of-n Scheme: Hybrid Proof

Assuming our scheme is not perfectly secure, we know:
$\Pr[A(H^0) = 1] \neq \Pr[A(H^{\log n}) = 1]$.

It follows that there must exist a $j \in \{1, \ldots, \log n\}$ such that

$$\Pr[A(H^{j-1}) = 1] \neq \Pr[A(H^j) = 1]$$

(otherwise, if all adjacent hybrids produce equal probabilities, the end hybrids would also have equal probabilities)

2-out-of-n Scheme: Hybrid Proof

So, A outputs 1 with different probabilities when applied to

$$H^{j-1} \rightarrow (s'^{1}_{i_1}, \ldots, s'^{j-1}_{i_{j-1}}, s^{j}_{i_j}, s^{j+1}_{i_{j+1}}, \ldots, s^{\log n}_{i_{\log n}})$$

vs. when applied to

$$H^{j} \rightarrow (s'^{1}_{i_1}, \ldots, s'^{j-1}_{i_{j-1}}, s'^{j}_{i_j}, s^{j+1}_{i_{j+1}}, \ldots, s^{\log n}_{i_{\log n}})$$

2-out-of-n Scheme: Hybrid Proof

So, A outputs 1 with different probabilities when applied to

$$H^{j-1} \rightarrow (s_{i_1}^{\prime 1}, \ldots, s_{i_{j-1}}^{\prime j-1}, s_{i_j}^{j}, s_{i_{j+1}}^{j+1}, \ldots, s_{i_{\log n}}^{\log n})$$

vs. when applied to

$$H^{j} \rightarrow (s_{i_1}^{\prime 1}, \ldots, s_{i_{j-1}}^{\prime j-1}, s_{i_j}^{\prime j}, s_{i_{j+1}}^{j+1}, \ldots, s_{i_{\log n}}^{\log n})$$

These hybrids are "adjacent" in a sense, differing in only one location (j), with A still behaving differently on their distributions.

2-out-of-n Scheme: Hybrid Proof

So, A outputs 1 with different probabilities when applied to

$$H^{j-1} \rightarrow (s'^{1}_{i_1}, \ldots, s'^{j-1}_{i_{j-1}}, s^{j}_{i_j}, s^{j+1}_{i_{j+1}}, \ldots, s^{\log n}_{i_{\log n}})$$

vs. when applied to

$$H^{j} \rightarrow (s'^{1}_{i_1}, \ldots, s'^{j-1}_{i_{j-1}}, s'^{j}_{i_j}, s^{j+1}_{i_{j+1}}, \ldots, s^{\log n}_{i_{\log n}})$$

These hybrids are "adjacent" in a sense, differing in only one location (j), with A still behaving differently on their distributions. We are now ready to define B, the algorithm that uses A to break the 2-out-of-2 scheme by "plugging it in" that location.

2-out-of-n Scheme: Hybrid Proof

We define B as follows (where i, j, m, m' are all hard-coded into B):

B: chooses to attack messages m, m' with share i_j.
On input $\mathbf{s} = s_{i_j}$,

2-out-of-n Scheme: Hybrid Proof

We define B as follows (where i, j, m, m' are all hard-coded into B):

B: chooses to attack messages m, m' with share i_j.
On input $\mathbf{s} = s_{i_j}$,

- For $k = 1, \ldots, j-1$, run $\mathtt{Share}_{2-2}(m') \to (s_0'^k, s_1'^k)$.
- For $k = j+1, \ldots, \log n$, run $\mathtt{Share}_{2-2}(m) \to (s_0^k, s_1^k)$
- Set $S_i = (s_{i_1}'^1, \ldots, s_{i_{j-1}}'^{j-1}, \mathbf{s}, s_{i_{j+1}}^{j+1}, \ldots, s_{i_{\log n}}^{\log n})$

2-out-of-n Scheme: Hybrid Proof

We define B as follows (where i, j, m, m' are all hard-coded into B):

B: chooses to attack messages m, m' with share i_j.
On input $\mathbf{s} = s_{i_j}$,

- For $k = 1, \ldots, j-1$, run $\texttt{Share}_{2-2}(m') \rightarrow ({s'}_0^k, {s'}_1^k)$.
- For $k = j+1, \ldots, \log n$, run $\texttt{Share}_{2-2}(m) \rightarrow (s_0^k, s_1^k)$
- Set $S_i = ({s'}_{i_1}^1, \ldots, {s'}_{i_{j-1}}^{j-1}, \mathbf{s}, s_{i_{j+1}}^{j+1}, \ldots, s_{i_{\log n}}^{\log n})$
- Run $A(S_i)$ and output the same.

If $\mathbf{s}$ came from running $\texttt{Share}_{2-2}$ on m, then S_i is drawn from the H^{j-1} distribution.

2-out-of-n Scheme: Hybrid Proof

We define B as follows (where i, j, m, m' are all hard-coded into B):

B: chooses to attack messages m, m' with share i_j.
On input $\mathbf{s} = s_{i_j}$,

- For $k = 1, \ldots, j-1$, run $\texttt{Share}_{2-2}(m') \to ({s'}_0^k, {s'}_1^k)$.
- For $k = j+1, \ldots, \log n$, run $\texttt{Share}_{2-2}(m) \to (s_0^k, s_1^k)$
- Set $S_i = ({s'}_{i_1}^1, \ldots, {s'}_{i_{j-1}}^{j-1}, \mathbf{s}, s_{i_{j+1}}^{j+1}, \ldots, s_{i_{\log n}}^{\log n})$
- Run $A(S_i)$ and output the same.

If $\mathbf{s}$ came from running $\texttt{Share}_{2-2}$ on m, then S_i is drawn from the H^{j-1} distribution.

If $\mathbf{s}$ came from running $\texttt{Share}_{2-2}$ on m', then S_i is drawn from the H^j distribution.

2-out-of-n Scheme: Hybrid Proof

So,

$$\Pr_{\mathtt{Share}_{2-2}(m)\rightarrow(s_0,s_1)}[B(s_{i_j}) = 1] = \Pr[A(H^{j-1}) = 1]$$

while

$$\Pr_{\mathtt{Share}_{2-2}(m')\rightarrow(s'_0,s'_1)}[B(s'_{i_j}) = 1] = \Pr[A(H^{j}) = 1]$$

We know that $\Pr[A(H^{j-1}) = 1] \neq \Pr[A(H^{j}) = 1]$

2-out-of-n Scheme: Hybrid Proof

So,

$$\Pr_{\mathtt{Share}_{2-2}(m)\to(s_0,s_1)}[B(s_{i_j})=1]=\Pr[A(H^{j-1})=1]$$

while

$$\Pr_{\mathtt{Share}_{2-2}(m')\to(s'_0,s'_1)}[B(s'_{i_j})=1]=\Pr[A(H^{j})=1]$$

We know that $\Pr[A(H^{j-1})=1]\neq\Pr[A(H^{j})=1]$, so

$$\Pr_{\mathtt{Share}_{2-2}(m)\to(s_0,s_1)}[B(s_{i_j})=1]\neq\Pr_{\mathtt{Share}_{2-2}(m')\to(s'_0,s'_1)}[B(s'_{i_j})=1]$$

2-out-of-n Scheme: Hybrid Proof

So,

$$\Pr_{\mathtt{Share}_{2-2}(m)\rightarrow(s_0,s_1)}[B(s_{i_j})=1]=\Pr[A(H^{j-1})=1]$$

while

$$\Pr_{\mathtt{Share}_{2-2}(m')\rightarrow(s'_0,s'_1)}[B(s'_{i_j})=1]=\Pr[A(H^{j})=1]$$

We know that $\Pr[A(H^{j-1})=1]\neq\Pr[A(H^{j})=1]$, so

$$\Pr_{\mathtt{Share}_{2-2}(m)\rightarrow(s_0,s_1)}[B(s_{i_j})=1]\neq\Pr_{\mathtt{Share}_{2-2}(m')\rightarrow(s'_0,s'_1)}[B(s'_{i_j})=1]$$

(so B breaks the perfect security of the 2-out-of-2 scheme – there exists m, m', an index i_j and an algorithm B such that the above probability holds.)

2-out-of-n Scheme: Hybrid Proof

This is a contradiction. We know from last class that the 2-out-of-2 scheme is perfectly secure.

So our original assumption (that there exists an A that breaks the perfect security of the 2-out-of-n scheme) must be false, and therefore the 2-out-of-n scheme is perfectly secure.

Some Number Theory

Everybody knows that “two points determine a line” (this is a postulate of Euclidean geometry).

Some Number Theory

Everybody knows that “two points determine a line” (this is a postulate of Euclidean geometry).

It is also true that 3 points determine a parabola, and so on.

Some Number Theory

Everybody knows that "two points determine a line" (this is a postulate of Euclidean geometry).

It is also true that 3 points determine a parabola, and so on.

Namely: $d + 1$ points determine a unique degree-d polynomial, and this is true even working modulo a prime.

Some Number Theory

$$\mathbb{Z}_p = \{0, ..., p - 1\}$$

Combined with modular addition and multiplication, $\mathbb{Z}_p$ is a *field* when p is prime. (every nonzero element has an additive and multiplicative inverse)

Some Number Theory

$$\mathbb{Z}_p = \{0, ..., p-1\}$$

Combined with modular addition and multiplication, $\mathbb{Z}_p$ is a *field* when p is prime. (every nonzero element has an additive and multiplicative inverse)

$\mathbb{Z}_p$ has p elements. A particular x is therefore drawn with probability $\frac{1}{p}$.

Some Number Theory

$$\mathbb{Z}_p = \{0, ..., p-1\}$$

Combined with modular addition and multiplication, $\mathbb{Z}_p$ is a *field* when p is prime. (every nonzero element has an additive and multiplicative inverse)

$\mathbb{Z}_p$ has p elements. A particular x is therefore drawn with probability $\frac{1}{p}$.

A degree-d polynomial $f(x) = \sum_{i=0}^{d} f_i x^i$ can be evaluated on inputs in both $\mathbb{R}$ and $\mathbb{Z}_p$

Some Number Theory

$$\mathbb{Z}_p = \{0, ..., p-1\}$$

Combined with modular addition and multiplication, $\mathbb{Z}_p$ is a *field* when p is prime. (every nonzero element has an additive and multiplicative inverse)

$\mathbb{Z}_p$ has p elements. A particular x is therefore drawn with probability $\frac{1}{p}$.

A degree-d polynomial $f(x) = \sum_{i=0}^{d} f_i x^i$ can be evaluated on inputs in both $\mathbb{R}$ and $\mathbb{Z}_p$

Some Number Theory

Theorem (Polynomial Uniqueness and Interpolation)

Let p be a prime, and let $\{(x_1, y_1), ..., (x_{d+1}, y_{d+1})\} \subseteq \mathbb{Z}_p \times \mathbb{Z}_p$ be a set of points whose x_i values are all distinct.
Then there is a unique degree-d polynomial f with coefficients in $\mathbb{Z}_p$ that satisfies $y_i = f(x_i)$ for all i.
(This f can be obtained from the $d + 1$ points via polynomial interpolation).

Shamir's Secret Sharing Scheme

We would like to have a t out of n secret sharing scheme. We just saw that $d + 1$ points are enough to uniquely define a degree d polynomial (the polynomial can be reconstructed via polynomial interpolation).

Shamir's Secret Sharing Scheme

We would like to have a t out of n secret sharing scheme. We just saw that $d+1$ points are enough to uniquely define a degree d polynomial (the polynomial can be reconstructed via polynomial interpolation).

A natural approach to build a secret sharing scheme is to let each user's share be a point on a polynomial. This is exactly what Shamir Secret Sharing does.

Shamir's Secret Sharing Scheme

We would like to have a t out of n secret sharing scheme. We just saw that $d+1$ points are enough to uniquely define a degree d polynomial (the polynomial can be reconstructed via polynomial interpolation).

A natural approach to build a secret sharing scheme is to let each user's share be a point on a polynomial. This is exactly what Shamir Secret Sharing does.

To share a secret $m \in \mathbb{Z}_p$ with threshold t out of n to reconstruct, we choose a degree $t-1$ polynomial that satisfies $f(0) = m$, with all other coefficients chosen uniformly at random from $\mathbb{Z}_p$. The share of the ith user is $(i, f(i))$.

Shamir's Secret Sharing Scheme

We would like to have a t out of n secret sharing scheme. We just saw that $d+1$ points are enough to uniquely define a degree d polynomial (the polynomial can be reconstructed via polynomial interpolation).

A natural approach to build a secret sharing scheme is to let each user's share be a point on a polynomial. This is exactly what Shamir Secret Sharing does.

To share a secret $m \in \mathbb{Z}_p$ with threshold t out of n to reconstruct, we choose a degree $t-1$ polynomial that satisfies $f(0) = m$, with all other coefficients chosen uniformly at random from $\mathbb{Z}_p$. The share of the ith user is $(i, f(i))$.

The interpolation theorem says any t shares can uniquely determine f, and hence recover the secret $f(0) = m$.

Shamir's Secret Sharing Scheme

$\texttt{Share}_{shamir}$: On input $m \in \mathbb{Z}_p$,

- select $f_1, ..., f_{t-1}$ uniformly at random from $\mathbb{Z}_p$.

Shamir's Secret Sharing Scheme

$\texttt{Share}_{shamir}$: On input $m \in \mathbb{Z}_p$,

- select $f_1, ..., f_{t-1}$ uniformly at random from $\mathbb{Z}_p$.
- define $f(x) = m + \sum_{i=1}^{t-1} f_i x^i$

Shamir's Secret Sharing Scheme

$\texttt{Share}_{shamir}$: On input $m \in \mathbb{Z}_p$,

- select $f_1, ..., f_{t-1}$ uniformly at random from $\mathbb{Z}_p$.
- define $f(x) = m + \sum_{i=1}^{t-1} f_i x^i$
- for $i = 1$ to n:
  - create share $s_i = (i, f(i))$.
- output: $(s_1, ..., s_n)$

Shamir's Secret Sharing Scheme

$\texttt{Share}_{shamir}$: On input $m \in \mathbb{Z}_p$,

- select $f_1, ..., f_{t-1}$ uniformly at random from $\mathbb{Z}_p$.
- define $f(x) = m + \sum_{i=1}^{t-1} f_i x^i$
- for $i = 1$ to n:
  - create share $s_i = (i, f(i))$.
- output: $(s_1, ..., s_n)$

$\texttt{Reconstruct}_{shamir}$: On input $(s_i : i \in S)$

- interpolate t points of s_i to obtain f, the unique degree $t-1$ polynomial passing through these points.
- output $f(0)$

Shamir's Secret Sharing Scheme

$\texttt{Share}_{shamir}$: On input $m \in \mathbb{Z}_p$,

- select $f_1, ..., f_{t-1}$ uniformly at random from $\mathbb{Z}_p$.
- define $f(x) = m + \sum_{i=1}^{t-1} f_i x^i$
- for $i = 1$ to n:
  - create share $s_i = (i, f(i))$.
- output: $(s_1, ..., s_n)$

$\texttt{Reconstruct}_{shamir}$: On input $(s_i : i \in S)$

- interpolate t points of s_i to obtain f, the unique degree $t-1$ polynomial passing through these points.
- output $f(0)$

(correctness follows from interpolation theorem)

Shamir Security

Recall the perfect security definition:

Definition (secret sharing security via identical distributions)

A t-out-of-n secret sharing scheme $(\texttt{Share}, \texttt{Reconstruct})$ over M is perfectly secure if:
$\forall m, m' \in \mathcal{M}$, $\forall S \subseteq \{1, \ldots, n\}$ s.t. $|S| < t$, the following distributions are identical:

$$\{(s_i | i \in S) : (s_1, \ldots, s_n) \leftarrow \texttt{Share}(m)\}$$

$$\{(s'_i | i \in S) : (s'_1, \ldots, s'_n) \leftarrow \texttt{Share}(m')\}$$

Shamir Security

Equivalently: $\forall m, m' \in \mathcal{M}$, $\forall S \subseteq \{1, \ldots, n\}$ s.t. $|S| < t$, and for any set $\alpha = (\alpha_1, \ldots, \alpha_{|S|})$, we have that

$$\Pr_{\texttt{Share}(m) \to (s_1, \ldots, s_n)}[(s_i | i \in S) = \alpha] = \Pr_{\texttt{Share}(m') \to (s'_1, \ldots, s'_n)}[(s'_i | i \in S) = \alpha]$$

Shamir Security

Consider the distribution of $\texttt{Share}_{shamir}(m) \rightarrow (s_1, \ldots, s_n)$. Then, for any $\alpha = (\alpha_1, \ldots, \alpha_{|S|})$, consider:

$$\Pr_{\texttt{Share}_{shamir}(m) \rightarrow (s_1, \ldots, s_n)} [(s_i | i \in S) = \alpha]$$

for an unauthorized set S of size $t - 1$.

Shamir Security

Consider the distribution of $\texttt{Share}_{shamir}(m) \to (s_1, \ldots, s_n)$. Then, for any $\alpha = (\alpha_1, \ldots, \alpha_{|S|})$, consider:

$$\Pr_{\texttt{Share}_{shamir}(m) \to (s_1, \ldots, s_n)}[(s_i | i \in S) = \alpha]$$

for an unauthorized set S of size $t - 1$.

$(s_i | i \in S) = \alpha$ happens if and only if the polynomial chosen by $\texttt{Share}_{shamir}$ happens to have $f(i) = \alpha_i$ for each $i \in S$ and $f(0) = m$.

Shamir Security

Consider the distribution of $\texttt{Share}_{shamir}(m) \to (s_1, \ldots, s_n)$. Then, for any $\alpha = (\alpha_1, \ldots, \alpha_{|S|})$, consider:

$$\Pr_{\texttt{Share}_{shamir}(m) \to (s_1, \ldots, s_n)}[(s_i | i \in S) = \alpha]$$

for an unauthorized set S of size $t - 1$.

$(s_i | i \in S) = \alpha$ happens if and only if the polynomial chosen by $\texttt{Share}_{shamir}$ happens to have $f(i) = \alpha_i$ for each $i \in S$ and $f(0) = m$.

By the polynomial interpolation theorem, there is one unique degree $t - 1$ polynomial that satisfies these t constraints. The $\texttt{Share}_{shamir}$ chooses a degree $t - 1$ polynomial uniformly from the set of p^{t-1} polynomials that satisfy $f(0) = m$ (this is done by choosing f_i at random from $\mathbb{Z}_p$ for $i = 1, ..., t - 1$). So, this probability is $\frac{1}{p^{t-1}}$.

Shamir Security

So we have that:

$$\Pr_{\texttt{Share}_{shamir}(m) \to (s_1,\ldots,s_n)}[(s_i | i \in S) = \alpha] = \frac{1}{p^{t-1}}$$

for an unauthorized set S of size $t-1$.

Shamir Security

So we have that:

$$\Pr_{\texttt{Share}_{shamir}(m)\to(s_1,\ldots,s_n)}[(s_i | i \in S) = \alpha] = \frac{1}{p^{t-1}}$$

for an unauthorized set S of size $t-1$.

Notice that we can repeat this argument for the distribution of $\texttt{Share}_{shamir}(m') \to (s'_1, \ldots, s'_n)$! (Nothing in the argument depended on the particular value for m).

Shamir Security

So we have that:

$$\Pr_{\texttt{Share}_{shamir}(m)\to(s_1,\ldots,s_n)}[(s_i|i \in S) = \alpha] = \frac{1}{p^{t-1}}$$

for an unauthorized set S of size $t-1$.

Notice that we can repeat this argument for the distribution of $\texttt{Share}_{shamir}(m') \to (s'_1, \ldots, s'_n)$! (Nothing in the argument depended on the particular value for m).

So we also have that:

$$\Pr_{\texttt{Share}_{shamir}(m')\to(s'_1,\ldots,s'_n)}[(s'_i|i \in S) = \alpha] = \frac{1}{p^{t-1}}$$

for an unauthorized set S of size $t-1$.

Shamir Security

Therefore, for any m, m', for any α, and for any unauthorized set S of size $t-1$, we have that:

$$\Pr_{\substack{\texttt{Share}_{shamir}(m) \\ \rightarrow (s_1,\ldots,s_n)}}[(s_i | i \in S) = \alpha] = \frac{1}{p^{t-1}} = \Pr_{\substack{\texttt{Share}_{shamir}(m') \\ \rightarrow (s'_1,\ldots,s'_n)}}[(s'_i | i \in S) = \alpha]$$

and therefore Shamir t-out-of-n secret sharing satisfies perfect security.