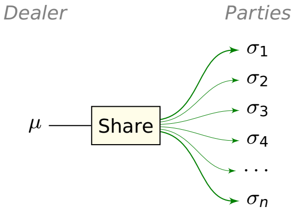
Leakage Resilient Non-Malleable Secret Sharing

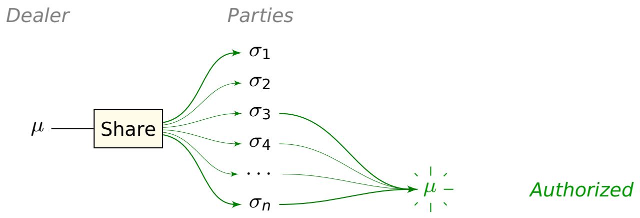
Gianluca Brian
Sapienza University of Rome
Rome, Italy

12-13 october 2020
Part 1

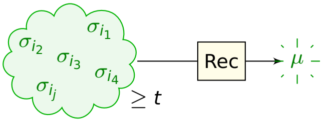
Secret Sharing



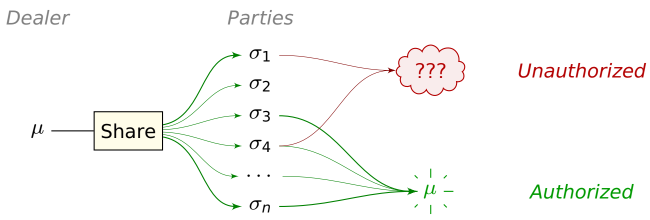
Secret Sharing



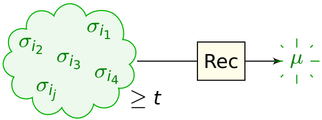
- **Correctness:** at least t parties are required to reconstruct the secret.
- t is the *threshold*.

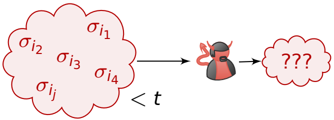


Secret Sharing



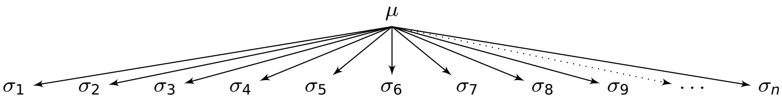
- **Correctness:** at least t parties are required to reconstruct the secret.
- **Privacy:** less than t parties should not be able to learn any information about the secret.
- t is the *threshold*.





Additional properties

- Homomorphic Secret Sharing



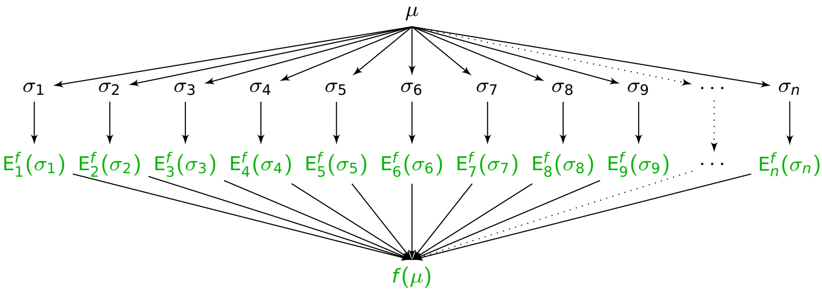
Additional properties

- Homomorphic Secret Sharing

$$
\begin{tikzcd}[column sep=small]
 & & & & & \mu \arrow[dlllll] \arrow[dllll] \arrow[dlll] \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] \arrow[drrr] \arrow[drrrr, dotted] \arrow[drrrrr] & & & & & \\
\sigma_1 \arrow[d] & \sigma_2 \arrow[d] & \sigma_3 \arrow[d] & \sigma_4 \arrow[d] & \sigma_5 \arrow[d] & \sigma_6 \arrow[d] & \sigma_7 \arrow[d] & \sigma_8 \arrow[d] & \sigma_9 \arrow[d] & \cdots \arrow[d, dotted] & \sigma_n \arrow[d] \\
\mathsf{E}^f_1(\sigma_1) & \mathsf{E}^f_2(\sigma_2) & \mathsf{E}^f_3(\sigma_3) & \mathsf{E}^f_4(\sigma_4) & \mathsf{E}^f_5(\sigma_5) & \mathsf{E}^f_6(\sigma_6) & \mathsf{E}^f_7(\sigma_7) & \mathsf{E}^f_8(\sigma_8) & \mathsf{E}^f_9(\sigma_9) & \cdots & \mathsf{E}^f_n(\sigma_n)
\end{tikzcd}
$$

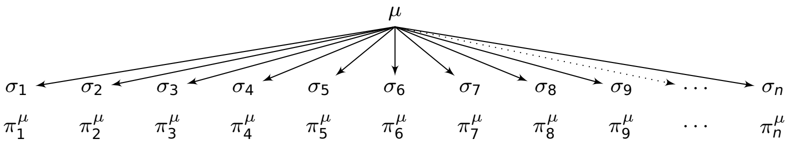
Additional properties

- Homomorphic Secret Sharing



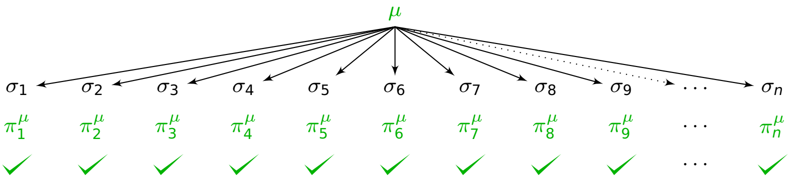
Additional properties

- Homomorphic Secret Sharing
- Verifiable Secret Sharing



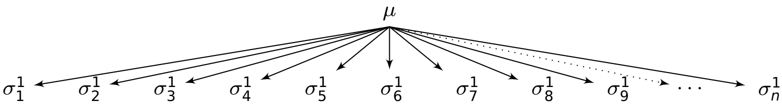
Additional properties

- Homomorphic Secret Sharing
- Verifiable Secret Sharing



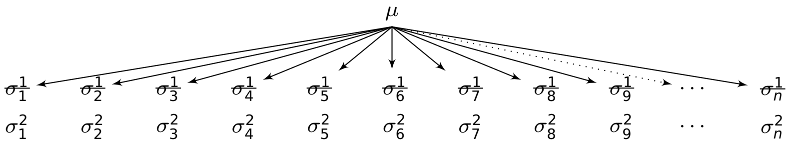
Additional properties

- Homomorphic Secret Sharing
- Verifiable Secret Sharing
- Proactive Secret Sharing



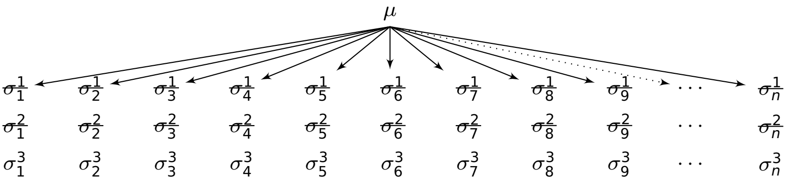
Additional properties

- Homomorphic Secret Sharing
- Verifiable Secret Sharing
- Proactive Secret Sharing



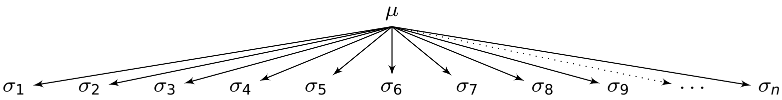
Additional properties

- Homomorphic Secret Sharing
- Verifiable Secret Sharing
- Proactive Secret Sharing



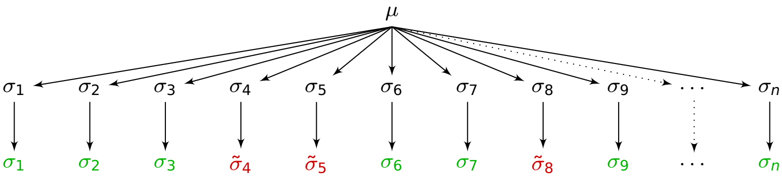
Additional properties

- Homomorphic Secret Sharing
- Verifiable Secret Sharing
- Proactive Secret Sharing
- Robust Secret Sharing



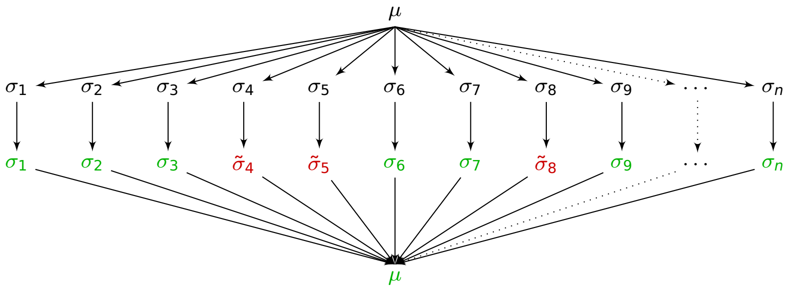
Additional properties

- Homomorphic Secret Sharing
- Verifiable Secret Sharing
- Proactive Secret Sharing
- Robust Secret Sharing



Additional properties

- Homomorphic Secret Sharing
- Verifiable Secret Sharing
- Proactive Secret Sharing
- Robust Secret Sharing



Additional properties

- Homomorphic Secret Sharing
- Verifiable Secret Sharing
- Proactive Secret Sharing
- Robust Secret Sharing
- Leakage Resilient Secret Sharing
- Non-Malleable Secret Sharing

LATER IN THE TALK...

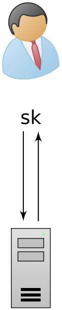
Motivation

- Secure & reliable storage

sk

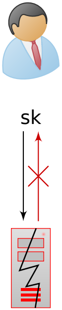
Motivation

- Secure & reliable storage



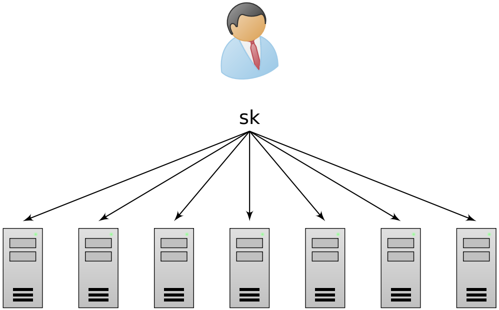
Motivation

- Secure & reliable storage



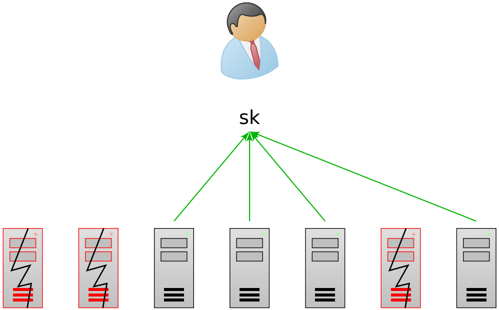
Motivation

- Secure & reliable storage



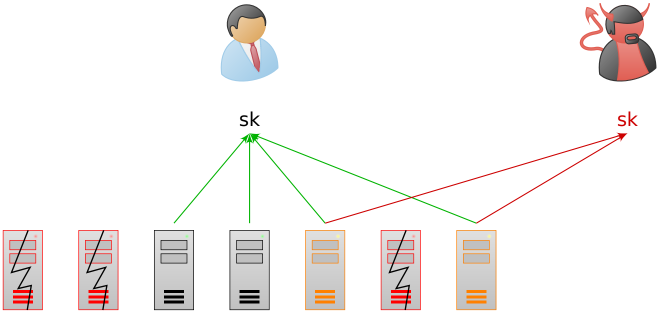
Motivation

- Secure & reliable storage



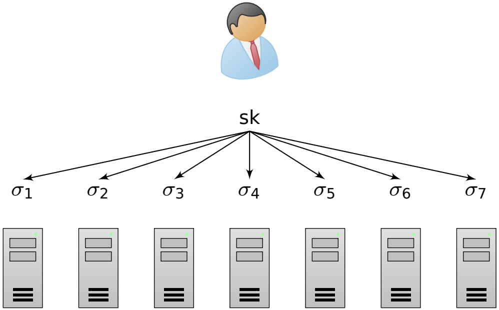
Motivation

- Secure & reliable storage



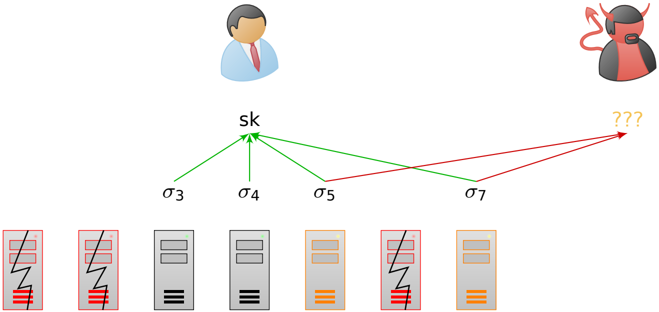
Motivation

- Secure & reliable storage



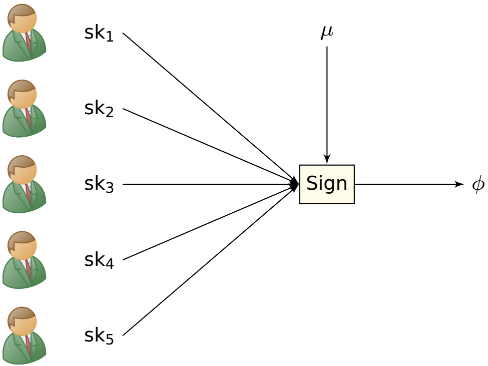
Motivation

- Secure & reliable storage



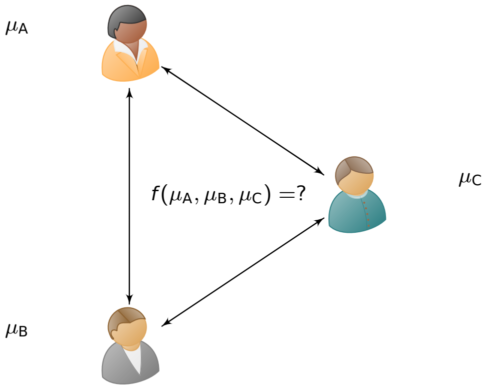
Motivation

- Secure & reliable storage
- Threshold Cryptography



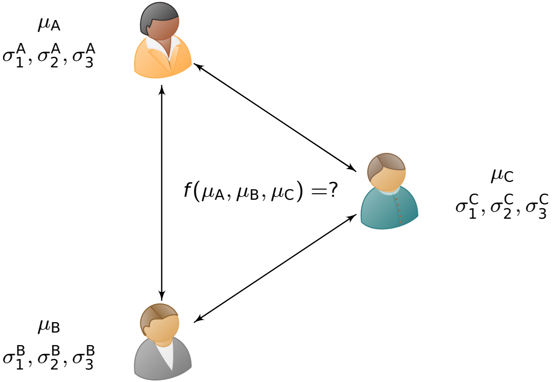
Motivation

- Secure & reliable storage
- Threshold Cryptography
- Multi-Party Computation



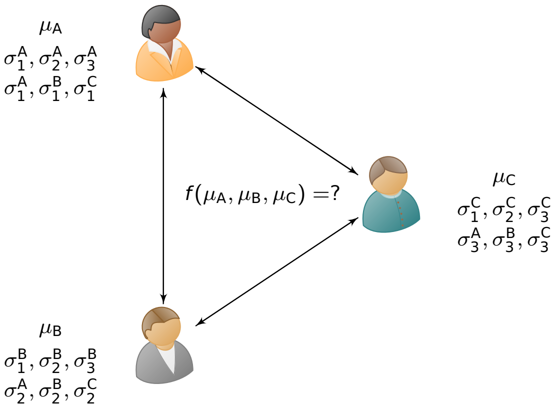
Motivation

- Secure & reliable storage
- Threshold Cryptography
- Multi-Party Computation



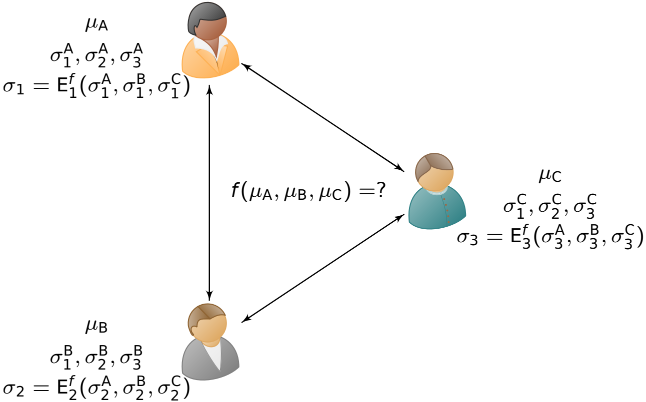
Motivation

- Secure & reliable storage
- Threshold Cryptography
- Multi-Party Computation



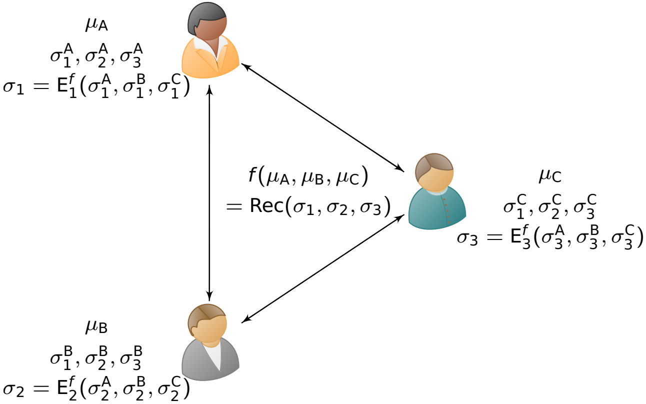
Motivation

- Secure & reliable storage
- Threshold Cryptography
- Multi-Party Computation



Motivation

- Secure & reliable storage
- Threshold Cryptography
- Multi-Party Computation



Formal definition

A t-out-of-n secret sharing scheme a tuple of algorithms $\Pi = (\mathsf{Share}, \mathsf{Rec})$:

Formal definition

A t-out-of-n secret sharing scheme a tuple of algorithms $\Pi = (\mathsf{Share}, \mathsf{Rec})$:

- Share takes as input a message $\mu \in \mathcal{M}$ and outputs n shares $\sigma_1, \ldots, \sigma_n$;

Formal definition

A t-out-of-n secret sharing scheme a tuple of algorithms $\Pi = (\mathsf{Share}, \mathsf{Rec})$:

- Share takes as input a message $\mu \in \mathcal{M}$ and outputs n shares $\sigma_1, \ldots, \sigma_n$;
- Rec takes as input a set of shares $(\sigma_i)_{i \in \mathcal{I}}$, where $|\mathcal{I}| \geq t$, and outputs a message μ'.

Formal definition

A t-out-of-n secret sharing scheme a tuple of algorithms $\Pi = (\mathsf{Share}, \mathsf{Rec})$:

- Share takes as input a message $\mu \in \mathcal{M}$ and outputs n shares $\sigma_1, \ldots, \sigma_n$;
- Rec takes as input a set of shares $(\sigma_i)_{i \in \mathcal{I}}$, where $|\mathcal{I}| \geq t$, and outputs a message μ'.

Main properties

- **Correctness:** For any message $\mu \in \mathcal{M}$ and any subset $\mathcal{I}$ of at least t indices, if $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu)$, then $\mathsf{Rec}((\sigma_i)_{i \in \mathcal{I}}) = \mu$ with probability 1 over the randomness of Share.

Formal definition

A t-out-of-n secret sharing scheme a tuple of algorithms $\Pi = (\mathsf{Share}, \mathsf{Rec})$:

- Share takes as input a message $\mu \in \mathcal{M}$ and outputs n shares $\sigma_1, \ldots, \sigma_n$;
- Rec takes as input a set of shares $(\sigma_i)_{i \in \mathcal{I}}$, where $|\mathcal{I}| \geq t$, and outputs a message μ'.

Main properties

- **Correctness:** For any message $\mu \in \mathcal{M}$ and any subset $\mathcal{I}$ of at least t indices, if $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu)$, then $\mathsf{Rec}((\sigma_i)_{i \in \mathcal{I}}) = \mu$ with probability 1 over the randomness of Share.
- **Privacy:** For any two messages $\mu_0, \mu_1 \in \mathcal{M}$ and any subset $\mathcal{U}$ of at most $t-1$ indices, let, for all $b \in \{0, 1\}$, $(\mathbf{\Sigma}_1^b, \ldots, \mathbf{\Sigma}_n^b) = \mathsf{Share}(\mu_b)$. Then, $(\mathbf{\Sigma}_i^0)_{i \in \mathcal{U}} \equiv (\mathbf{\Sigma}_i^1)_{i \in \mathcal{U}}$.

Formal definition

A t-out-of-n secret sharing scheme a tuple of algorithms $\Pi = (\mathsf{Share}, \mathsf{Rec})$:

- Share takes as input a message $\mu \in \mathcal{M}$ and outputs n shares $\sigma_1, \ldots, \sigma_n$;
- Rec takes as input a set of shares $(\sigma_i)_{i \in \mathcal{I}}$, where $|\mathcal{I}| \geq t$, and outputs a message μ'.

Main properties

- **Correctness:** For any message $\mu \in \mathcal{M}$ and any subset $\mathcal{I}$ of at least t indices, if $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu)$, then $\mathsf{Rec}((\sigma_i)_{i \in \mathcal{I}}) = \mu$ with probability 1 over the randomness of Share.
- **Perfect privacy:** For any two messages $\mu_0, \mu_1 \in \mathcal{M}$ and any subset $\mathcal{U}$ of at most $t-1$ indices, let, for all $b \in \{0, 1\}$, $(\mathbf{\Sigma}_1^b, \ldots, \mathbf{\Sigma}_n^b) = \mathsf{Share}(\mu_b)$. Then, $(\mathbf{\Sigma}_i^0)_{i \in \mathcal{U}} \equiv (\mathbf{\Sigma}_i^1)_{i \in \mathcal{U}}$.

Formal definition

A t-out-of-n secret sharing scheme a tuple of algorithms $\Pi = (\mathsf{Share}, \mathsf{Rec})$:

- Share takes as input a message $\mu \in \mathcal{M}$ and outputs n shares $\sigma_1, \ldots, \sigma_n$;
- Rec takes as input a set of shares $(\sigma_i)_{i \in \mathcal{I}}$, where $|\mathcal{I}| \geq t$, and outputs a message μ'.

Main properties

- **Correctness:** For any message $\mu \in \mathcal{M}$ and any subset $\mathcal{I}$ of at least t indices, if $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\!\$\ \mathsf{Share}(\mu)$, then $\mathsf{Rec}((\sigma_i)_{i \in \mathcal{I}}) = \mu$ with probability 1 over the randomness of Share.
- **Perfect privacy:** For any two messages $\mu_0, \mu_1 \in \mathcal{M}$ and any subset $\mathcal{U}$ of at most $t-1$ indices, let, for all $b \in \{0,1\}$, $(\mathbf{\Sigma}_1^b, \ldots, \mathbf{\Sigma}_n^b) = \mathsf{Share}(\mu_b)$. Then, $(\mathbf{\Sigma}_i^0)_{i \in \mathcal{U}} \equiv (\mathbf{\Sigma}_i^1)_{i \in \mathcal{U}}$.
- **Statistical privacy:** ... Then, $(\mathbf{\Sigma}_i^0)_{i \in \mathcal{U}} \approx_\varepsilon (\mathbf{\Sigma}_i^1)_{i \in \mathcal{U}}$.

Formal definition

A t-out-of-n secret sharing scheme a tuple of algorithms $\Pi = (\mathsf{Share}, \mathsf{Rec})$:

- Share takes as input a message $\mu \in \mathcal{M}$ and outputs n shares $\sigma_1, \ldots, \sigma_n$;
- Rec takes as input a set of shares $(\sigma_i)_{i \in \mathcal{I}}$, where $|\mathcal{I}| \geq t$, and outputs a message μ'.

Main properties

- **Correctness:** For any message $\mu \in \mathcal{M}$ and any subset $\mathcal{I}$ of at least t indices, if $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu)$, then $\mathsf{Rec}((\sigma_i)_{i \in \mathcal{I}}) = \mu$ with probability 1 over the randomness of Share.
- **Perfect privacy:** For any two messages $\mu_0, \mu_1 \in \mathcal{M}$ and any subset $\mathcal{U}$ of at most $t - 1$ indices, let, for all $b \in \{0, 1\}$, $(\mathbf{\Sigma}_1^b, \ldots, \mathbf{\Sigma}_n^b) = \mathsf{Share}(\mu_b)$. Then, $(\mathbf{\Sigma}_i^0)_{i \in \mathcal{U}} \equiv (\mathbf{\Sigma}_i^1)_{i \in \mathcal{U}}$.
- **Statistical privacy:** ... Then, $(\mathbf{\Sigma}_i^0)_{i \in \mathcal{U}} \approx_\varepsilon (\mathbf{\Sigma}_i^1)_{i \in \mathcal{U}}$.
- **Computational privacy:** ... Then, $(\mathbf{\Sigma}_i^0)_{i \in \mathcal{U}} \approx_{\mathrm{C}} (\mathbf{\Sigma}_i^1)_{i \in \mathcal{U}}$.

Formal definition

A t-out-of-n secret sharing scheme a tuple of algorithms $\Pi = (\mathsf{Share}, \mathsf{Rec})$:

- Share takes as input a message $\mu \in \mathcal{M}$ and outputs n shares $\sigma_1, \ldots, \sigma_n$;
- Rec takes as input a set of shares $(\sigma_i)_{i\in\mathcal{I}}$, where $|\mathcal{I}| \geq t$, and outputs a message μ'.

Main properties

- **Correctness:** For any message $\mu \in \mathcal{M}$ and any subset $\mathcal{I}$ of at least t indices, if $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu)$, then $\mathsf{Rec}((\sigma_i)_{i\in\mathcal{I}}) = \mu$ with probability 1 over the randomness of Share.
- **Perfect privacy:** For any two messages $\mu_0, \mu_1 \in \mathcal{M}$ and any subset $\mathcal{U}$ of at most $t-1$ indices, let, for all $b \in \{0,1\}$, $(\boldsymbol{\Sigma}_1^b, \ldots, \boldsymbol{\Sigma}_n^b) = \mathsf{Share}(\mu_b)$. Then, $(\boldsymbol{\Sigma}_i^0)_{i\in\mathcal{U}} \equiv (\boldsymbol{\Sigma}_i^1)_{i\in\mathcal{U}}$.
- **Statistical privacy:** ... Then, $(\boldsymbol{\Sigma}_i^0)_{i\in\mathcal{U}} \approx_\varepsilon (\boldsymbol{\Sigma}_i^1)_{i\in\mathcal{U}}$.
- **Computational privacy:** ... Then, $(\boldsymbol{\Sigma}_i^0)_{i\in\mathcal{U}} \approx_{\mathsf{C}} (\boldsymbol{\Sigma}_i^1)_{i\in\mathcal{U}}$.

Access structure

A monotone class $\mathcal{A}$ of subsets of $[n]$. Defines the *authorized* subsets $\mathcal{I} \in \mathcal{A}$ and the *unauthorized* subsets $\mathcal{U} \notin \mathcal{A}$.
The t-out-of-n *threshold* access structure is the access structure $\mathcal{A} = \{\mathcal{I} \;:\; |\mathcal{I}| \geq t\}$.
For simplicity, in the rest of these slides, all access structures will be threshold access structures unless stated otherwise.

Example: Shamir Secret Sharing

- Based on polynomial interpolation.

[Sha79] "How to Share a Secret", *Adi Shamir*, Communications of the ACM, Volume 22, 1979

Example: Shamir Secret Sharing

- Based on polynomial interpolation.
- **Message:** a value $\mu \in \mathbb{F}$ in a finite field $\mathbb{F}$.

[Sha79] "How to Share a Secret", *Adi Shamir*, Communications of the ACM, Volume 22, 1979

Example: Shamir Secret Sharing

- Based on polynomial interpolation.
- **Message:** a value $\mu \in \mathbb{F}$ in a finite field $\mathbb{F}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{F}$, randomly sample a $t-1$ degree polynomial $p \in \mathbb{F}[X]$ such that $p(0) = \mu$.

[Sha79] "How to Share a Secret", *Adi Shamir*, Communications of the ACM, Volume 22, 1979

Example: Shamir Secret Sharing

- Based on polynomial interpolation.
- **Message:** a value $\mu \in \mathbb{F}$ in a finite field $\mathbb{F}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{F}$, randomly sample a $t-1$ degree polynomial $p \in \mathbb{F}[X]$ such that $p(0) = \mu$.
  - For all $i \in [n]$, compute $\sigma_i = p(i)$.

[Sha79] "How to Share a Secret", *Adi Shamir*, Communications of the ACM, Volume 22, 1979

Example: Shamir Secret Sharing

- Based on polynomial interpolation.
- **Message:** a value $\mu \in \mathbb{F}$ in a finite field $\mathbb{F}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{F}$, randomly sample a $t-1$ degree polynomial $p \in \mathbb{F}[X]$ such that $p(0) = \mu$.
  - For all $i \in [n]$, compute $\sigma_i = p(i)$.
  - Output the shares $(\sigma_1, \ldots, \sigma_n)$.

[Sha79] "How to Share a Secret", *Adi Shamir*, Communications of the ACM, Volume 22, 1979

Example: Shamir Secret Sharing

- Based on polynomial interpolation.
- **Message:** a value $\mu \in \mathbb{F}$ in a finite field $\mathbb{F}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{F}$, randomly sample a $t-1$ degree polynomial $p \in \mathbb{F}[X]$ such that $p(0) = \mu$.
  - For all $i \in [n]$, compute $\sigma_i = p(i)$.
  - Output the shares $(\sigma_1, \ldots, \sigma_n)$.
- **Algorithm** Rec**:**
  - Upon input at least t shares $(\sigma_i)_{i \in \mathcal{I}}$, interpolate the shares in order to find the only $t-1$ degree polynomial p such that, for all $i \in \mathcal{I}$, $\sigma_i = p(i)$.

[Sha79] "How to Share a Secret", *Adi Shamir*, Communications of the ACM, Volume 22, 1979

Example: Shamir Secret Sharing

- Based on polynomial interpolation.
- **Message:** a value $\mu \in \mathbb{F}$ in a finite field $\mathbb{F}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{F}$, randomly sample a $t-1$ degree polynomial $p \in \mathbb{F}[X]$ such that $p(0) = \mu$.
  - For all $i \in [n]$, compute $\sigma_i = p(i)$.
  - Output the shares $(\sigma_1, \ldots, \sigma_n)$.
- **Algorithm** Rec**:**
  - Upon input at least t shares $(\sigma_i)_{i \in \mathcal{I}}$, interpolate the shares in order to find the only $t-1$ degree polynomial p such that, for all $i \in \mathcal{I}$, $\sigma_i = p(i)$.
  - Compute and output $\mu = p(0)$.

[Sha79] “How to Share a Secret”, *Adi Shamir*, Communications of the ACM, Volume 22, 1979

Example: Shamir Secret Sharing

- Based on polynomial interpolation.
- **Message:** a value $\mu \in \mathbb{F}$ in a finite field $\mathbb{F}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{F}$, randomly sample a $t-1$ degree polynomial $p \in \mathbb{F}[X]$ such that $p(0) = \mu$.
  - For all $i \in [n]$, compute $\sigma_i = p(i)$.
  - Output the shares $(\sigma_1, \ldots, \sigma_n)$.
- **Algorithm** Rec**:**
  - Upon input at least t shares $(\sigma_i)_{i \in \mathcal{I}}$, interpolate the shares in order to find the only $t-1$ degree polynomial p such that, for all $i \in \mathcal{I}$, $\sigma_i = p(i)$.
  - Compute and output $\mu = p(0)$.

- **Correctness:** follows immediately.

[Sha79] "How to Share a Secret", *Adi Shamir*, Communications of the ACM, Volume 22, 1979

Example: Shamir Secret Sharing

- Based on polynomial interpolation.
- **Message:** a value $\mu \in \mathbb{F}$ in a finite field $\mathbb{F}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{F}$, randomly sample a $t-1$ degree polynomial $p \in \mathbb{F}[X]$ such that $p(0) = \mu$.
  - For all $i \in [n]$, compute $\sigma_i = p(i)$.
  - Output the shares $(\sigma_1, \ldots, \sigma_n)$.
- **Algorithm** Rec**:**
  - Upon input at least t shares $(\sigma_i)_{i \in \mathcal{I}}$, interpolate the shares in order to find the only $t-1$ degree polynomial p such that, for all $i \in \mathcal{I}$, $\sigma_i = p(i)$.
  - Compute and output $\mu = p(0)$.

- **Correctness:** follows immediately.
- **Privacy:** follows from the following alternative sharing algorithm.
  - Given any two messages μ_0, μ_1 and any subset $\mathcal{U}$ of $t-1$ indices, randomly sample $\sigma_i \in \mathbb{F}$ for all $i \in \mathcal{U}$.
  - For a secret sharing of μ_0, obtain a polynomial p by interpolating the values $\mu_0, (\sigma_i)_{i \in \mathcal{U}}$.
  - For a secret sharing of μ_1, obtain a polynomial p by interpolating the values $\mu_1, (\sigma_i)_{i \in \mathcal{U}}$.
  - Obtain all the remaining shares by computing $\sigma_i = p(i)$ for all $i \in [n] \setminus \mathcal{U}$.

[Sha79] "How to Share a Secret", *Adi Shamir*, Communications of the ACM, Volume 22, 1979

Example: Shamir Secret Sharing

- Based on polynomial interpolation.
- **Message:** a value $\mu \in \mathbb{F}$ in a finite field $\mathbb{F}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{F}$, randomly sample a $t-1$ degree polynomial $p \in \mathbb{F}[X]$ such that $p(0) = \mu$.
  - For all $i \in [n]$, compute $\sigma_i = p(i)$.
  - Output the shares $(\sigma_1, \ldots, \sigma_n)$.
- **Algorithm** Rec**:**
  - Upon input at least t shares $(\sigma_i)_{i \in \mathcal{I}}$, interpolate the shares in order to find the only $t-1$ degree polynomial p such that, for all $i \in \mathcal{I}$, $\sigma_i = p(i)$.
  - Compute and output $\mu = p(0)$.

- **Correctness:** follows immediately.
- **Privacy:** follows from the following alternative sharing algorithm.
  - Given any two messages μ_0, μ_1 and any subset $\mathcal{U}$ of $t-1$ indices, randomly sample $\sigma_i \in \mathbb{F}$ for all $i \in \mathcal{U}$.
  - For a secret sharing of μ_0, obtain a polynomial p by interpolating the values $\mu_0, (\sigma_i)_{i \in \mathcal{U}}$.
  - For a secret sharing of μ_1, obtain a polynomial p by interpolating the values $\mu_1, (\sigma_i)_{i \in \mathcal{U}}$.
  - Obtain all the remaining shares by computing $\sigma_i = p(i)$ for all $i \in [n] \setminus \mathcal{U}$.
  - Since the distribution of all the shares in $\mathcal{U}$ is the same for both μ_0 and μ_1, the scheme achieves perfect privacy.

[Sha79] “How to Share a Secret”, *Adi Shamir*, Communications of the ACM, Volume 22, 1979

Example: Additive Secret Sharing

- Based on finite groups; only *n*-out-of-*n* access structure.

Example: Additive Secret Sharing

- Based on finite groups; only *n*-out-of-*n* access structure.
- **Message:** a value $\mu \in \mathbb{G}$ in a finite group $\mathbb{G}$.

Example: Additive Secret Sharing

- Based on finite groups; only *n*-out-of-*n* access structure.
- **Message:** a value $\mu \in \mathbb{G}$ in a finite group $\mathbb{G}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{G}$, randomly sample a $n - 1$ group elements $\sigma_1, \ldots, \sigma_{n-1} \in \mathbb{G}$.

Example: Additive Secret Sharing

- Based on finite groups; only n-out-of-n access structure.
- **Message:** a value $\mu \in \mathbb{G}$ in a finite group $\mathbb{G}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{G}$, randomly sample a $n-1$ group elements $\sigma_1, \ldots, \sigma_{n-1} \in \mathbb{G}$.
  - Let $\sigma_n = \mu - \sum_{i=1}^{n-1} \sigma_i$.

Example: Additive Secret Sharing

- Based on finite groups; only *n*-out-of-*n* access structure.
- **Message:** a value $\mu \in \mathbb{G}$ in a finite group $\mathbb{G}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{G}$, randomly sample a $n - 1$ group elements $\sigma_1, \ldots, \sigma_{n-1} \in \mathbb{G}$.
  - Let $\sigma_n = \mu - \sum_{i=1}^{n-1} \sigma_i$.
  - Output the shares $(\sigma_1, \ldots, \sigma_n)$.

Example: Additive Secret Sharing

- Based on finite groups; only *n*-out-of-*n* access structure.
- **Message:** a value $\mu \in \mathbb{G}$ in a finite group $\mathbb{G}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{G}$, randomly sample a $n-1$ group elements $\sigma_1, \ldots, \sigma_{n-1} \in \mathbb{G}$.
  - Let $\sigma_n = \mu - \sum_{i=1}^{n-1} \sigma_i$.
  - Output the shares $(\sigma_1, \ldots, \sigma_n)$.
- **Algorithm** Rec**:**
  - Upon input all the shares $(\sigma_1, \ldots, \sigma_n)$, compute and output $\mu = \sum_{i=1}^{n} \sigma_i$.

Example: Additive Secret Sharing

- Based on finite groups; only n-out-of-n access structure.
- **Message:** a value $\mu \in \mathbb{G}$ in a finite group $\mathbb{G}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{G}$, randomly sample a $n-1$ group elements $\sigma_1, \ldots, \sigma_{n-1} \in \mathbb{G}$.
  - Let $\sigma_n = \mu - \sum_{i=1}^{n-1} \sigma_i$.
  - Output the shares $(\sigma_1, \ldots, \sigma_n)$.
- **Algorithm** Rec**:**
  - Upon input all the shares $(\sigma_1, \ldots, \sigma_n)$, compute and output $\mu = \sum_{i=1}^{n} \sigma_i$.

- **Correctness:** follows immediately.

Example: Additive Secret Sharing

- Based on finite groups; only n-out-of-n access structure.
- **Message:** a value $\mu \in \mathbb{G}$ in a finite group $\mathbb{G}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{G}$, randomly sample a $n-1$ group elements $\sigma_1, \ldots, \sigma_{n-1} \in \mathbb{G}$.
  - Let $\sigma_n = \mu - \sum_{i=1}^{n-1} \sigma_i$.
  - Output the shares $(\sigma_1, \ldots, \sigma_n)$.
- **Algorithm** Rec**:**
  - Upon input all the shares $(\sigma_1, \ldots, \sigma_n)$, compute and output $\mu = \sum_{i=1}^{n} \sigma_i$.

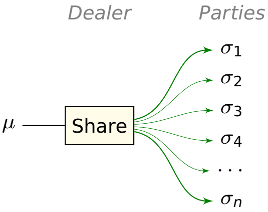
- **Correctness:** follows immediately.
- **Privacy:** follows from the following alternative sharing algorithm.
  - Given any two messages μ_0, μ_1 and any subset $\mathcal{U}$ of $n-1$ indices, randomly sample $\sigma_i \in \mathbb{G}$ for all $i \in \mathcal{U}$ and let $i^* \in [n] \setminus \mathcal{U}$.
  - For a secret sharing of μ_0, let $\sigma_{i^*} := \mu_0 - \sum_{i \in \mathcal{U}} \sigma_i$.
  - For a secret sharing of μ_1, let $\sigma_{i^*} := \mu_1 - \sum_{i \in \mathcal{U}} \sigma_i$.

Example: Additive Secret Sharing

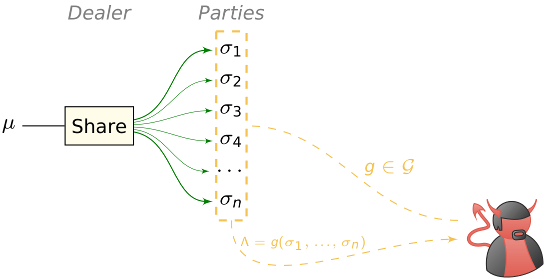
- Based on finite groups; only n-out-of-n access structure.
- **Message:** a value $\mu \in \mathbb{G}$ in a finite group $\mathbb{G}$.
- **Algorithm** Share**:**
  - Upon input a message $\mu \in \mathbb{G}$, randomly sample a $n-1$ group elements $\sigma_1, \ldots, \sigma_{n-1} \in \mathbb{G}$.
  - Let $\sigma_n = \mu - \sum_{i=1}^{n-1} \sigma_i$.
  - Output the shares $(\sigma_1, \ldots, \sigma_n)$.
- **Algorithm** Rec**:**
  - Upon input all the shares $(\sigma_1, \ldots, \sigma_n)$, compute and output $\mu = \sum_{i=1}^{n} \sigma_i$.

- **Correctness:** follows immediately.
- **Privacy:** follows from the following alternative sharing algorithm.
  - Given any two messages μ_0, μ_1 and any subset $\mathcal{U}$ of $n-1$ indices, randomly sample $\sigma_i \in \mathbb{G}$ for all $i \in \mathcal{U}$ and let $i^* \in [n] \setminus \mathcal{U}$.
  - For a secret sharing of μ_0, let $\sigma_{i*} := \mu_0 - \sum_{i \in \mathcal{U}} \sigma_i$.
  - For a secret sharing of μ_1, let $\sigma_{i*} := \mu_1 - \sum_{i \in \mathcal{U}} \sigma_i$.
  - Since the distribution of all the shares in $\mathcal{U}$ is the same for both μ_0 and μ_1, the scheme achieves perfect privacy.

Leakage Resilient Secret Sharing



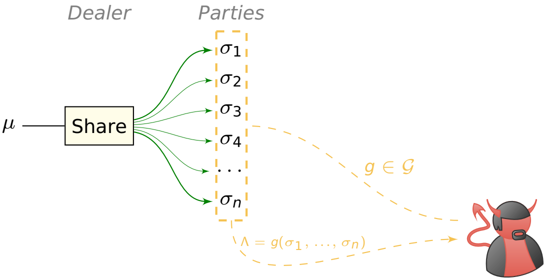
Leakage Resilient Secret Sharing



Side channel attacks: partial information from all the shares may reveal some information about the message!

SECURITY BREACH!

Leakage Resilient Secret Sharing

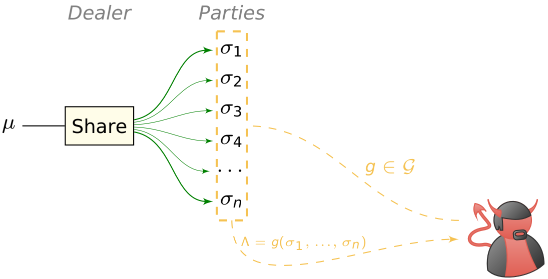


Side channel attacks: partial information from all the shares may reveal some information about the message!
SECURITY BREACH!

Leakage Resilient Secret Sharing [KMS18] **:** Λ reveals nothing about μ for a restricted family $\mathcal{G}$ of leakage functions.

[KMS18] "Leakage Resilient Secret Sharing", *Ashutosh Kumar, Raghu Meka, Amit Sahai*, IACR Cryptology ePrint Archive, Vol.2018/1138

Leakage Resilient Secret Sharing

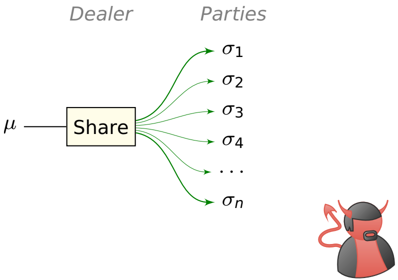


Side channel attacks: partial information from all the shares may reveal some information about the message!
SECURITY BREACH!

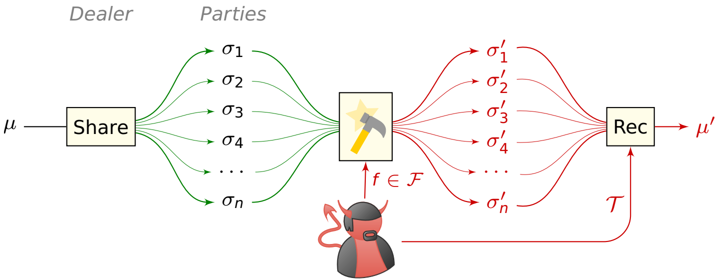
Leakage Resilient Secret Sharing [KMS18] **:** Λ reveals nothing about μ for a restricted family $\mathcal{G}$ of leakage functions.
Limitations: Impossible for arbitrary families $\mathcal{G}$ of leakage functions.

[KMS18] "Leakage Resilient Secret Sharing", *Ashutosh Kumar, Raghu Meka, Amit Sahai*, IACR Cryptology ePrint Archive, Vol.2018/1138

Non-malleable Secret Sharing



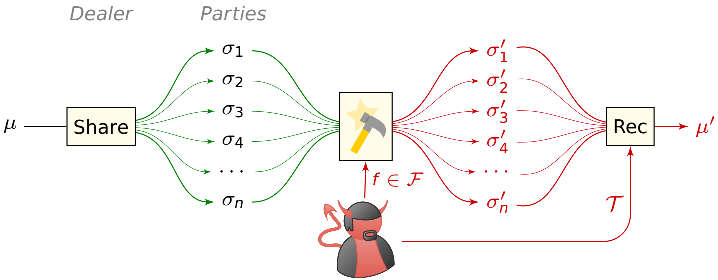
Non-malleable Secret Sharing



Tampering attacks: μ' may be related to μ!

SECURITY BREACH!

Non-malleable Secret Sharing



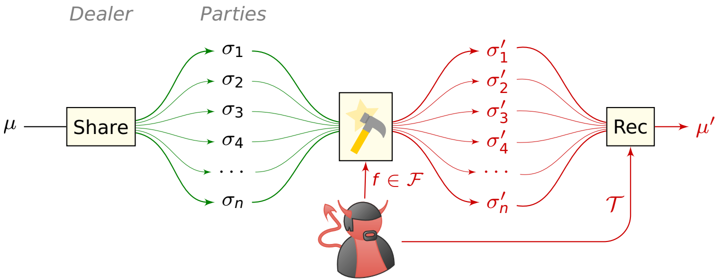
Tampering attacks: μ' may be related to μ!

SECURITY BREACH!

Non-Malleable Secret Sharing [GK18] **:** μ' is unrelated to μ for a restricted family $\mathcal{F}$ of tampering functions.

[GK18] "Non-Malleable Secret Sharing", *Vipul Goyal, Ashutosh Kumar*, 50th STOC 2018

Non-malleable Secret Sharing



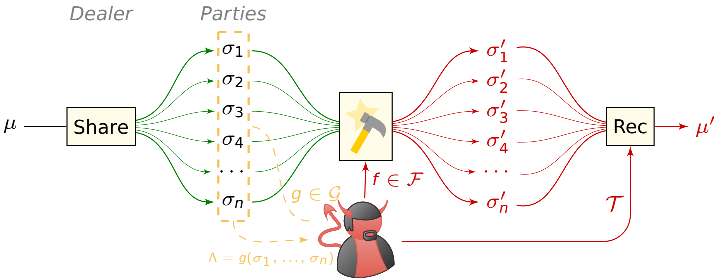
Tampering attacks: μ' may be related to μ!

SECURITY BREACH!

Non-Malleable Secret Sharing [GK18] **:** μ' is unrelated to μ for a restricted family $\mathcal{F}$ of tampering functions.
Limitations: Impossible for arbitrary families $\mathcal{F}$ of tampering functions.

[GK18] "Non-Malleable Secret Sharing", *Vipul Goyal, Ashutosh Kumar*, 50th STOC 2018

Non-malleable Secret Sharing



Tampering attacks: μ' may be related to μ!

SECURITY BREACH!

Non-Malleable Secret Sharing [GK18] **:** μ' is unrelated to μ for a restricted family $\mathcal{F}$ of tampering functions.
Limitations: Impossible for arbitrary families $\mathcal{F}$ of tampering functions.

Often, leakage resilience and non-malleability are considered together.

[GK18] “Non-Malleable Secret Sharing”, *Vipul Goyal, Ashutosh Kumar*, 50th STOC 2018

Limitations for leakage resilience and non-malleability

Why is it impossible to obtain leakage resilience or non malleability for arbitrary families of functions?

Limitations for leakage resilience and non-malleability

Why is it impossible to obtain leakage resilience or non malleability for arbitrary families of functions?

- Because, in particular, it is impossible to achieve security against functions taking as input t or more shares.

Limitations for leakage resilience and non-malleability

Why is it impossible to obtain leakage resilience or non malleability for arbitrary families of functions?

- Because, in particular, it is impossible to achieve security against functions taking as input t or more shares.
- **Leakage resilience:** the attacker may use the following leakage function.
    - Upon input the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$, compute $\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_t})$.
    - Output the first bit of μ.

Limitations for leakage resilience and non-malleability

Why is it impossible to obtain leakage resilience or non malleability for arbitrary families of functions?

- Because, in particular, it is impossible to achieve security against functions taking as input t or more shares.
- **Leakage resilience:** the attacker may use the following leakage function.
  - Upon input the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$, compute $\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_t})$.
  - Output the first bit of μ.
- **Non-malleability:** the attacker may apply the following tampering function.
  - Upon input the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$, compute $\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_t})$.
  - Flip all the bits of μ and call $\tilde{\mu}$ the resulting value.
  - Replace the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$ with a secret sharing of $\tilde{\mu}$.

Limitations for leakage resilience and non-malleability

Why is it impossible to obtain leakage resilience or non malleability for arbitrary families of functions?

- Because, in particular, it is impossible to achieve security against functions taking as input t or more shares.
- **Leakage resilience:** the attacker may use the following leakage function.
  - Upon input the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$, compute $\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_t})$.
  - Output the first bit of μ.
- **Non-malleability:** the attacker may apply the following tampering function.
  - Upon input the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$, compute $\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_t})$.
  - Flip all the bits of μ and call $\tilde{\mu}$ the resulting value.
  - Replace the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$ with a secret sharing of $\tilde{\mu}$.

- Usually, schemes achieve security against a certain class of leakage or tampering functions.

Limitations for leakage resilience and non-malleability

Why is it impossible to obtain leakage resilience or non malleability for arbitrary families of functions?

- Because, in particular, it is impossible to achieve security against functions taking as input t or more shares.
- **Leakage resilience:** the attacker may use the following leakage function.
  - Upon input the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$, compute $\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_t})$.
  - Output the first bit of μ.
- **Non-malleability:** the attacker may apply the following tampering function.
  - Upon input the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$, compute $\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_t})$.
  - Flip all the bits of μ and call $\tilde{\mu}$ the resulting value.
  - Replace the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$ with a secret sharing of $\tilde{\mu}$.

- Usually, schemes achieve security against a certain class of leakage or tampering functions.
- Leakage function have a further limitation on the amount of tolerated leakage (otherwise, they could leak all the shares and achieving security would be impossible).

Limitations for leakage resilience and non-malleability

Why is it impossible to obtain leakage resilience or non malleability for arbitrary families of functions?

- Because, in particular, it is impossible to achieve security against functions taking as input t or more shares.
- **Leakage resilience:** the attacker may use the following leakage function.
  - Upon input the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$, compute $\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_t})$.
  - Output the first bit of μ.
- **Non-malleability:** the attacker may apply the following tampering function.
  - Upon input the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$, compute $\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_t})$.
  - Flip all the bits of μ and call $\tilde{\mu}$ the resulting value.
  - Replace the shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$ with a secret sharing of $\tilde{\mu}$.

- Usually, schemes achieve security against a certain class of leakage or tampering functions.
- Leakage function have a further limitation on the amount of tolerated leakage (otherwise, they could leak all the shares and achieving security would be impossible).
- **Admissible adversaries:** the class of adversaries against which a scheme is leakage resilient and/or non-malleable.

Bounded leakage resilience — formal definition

$\sigma_1 \quad \sigma_2 \quad \sigma_3 \quad \sigma_4 \quad \sigma_5 \quad \sigma_6 \quad \sigma_7 \quad \sigma_8 \quad \sigma_9 \quad \cdots \quad \sigma_n$

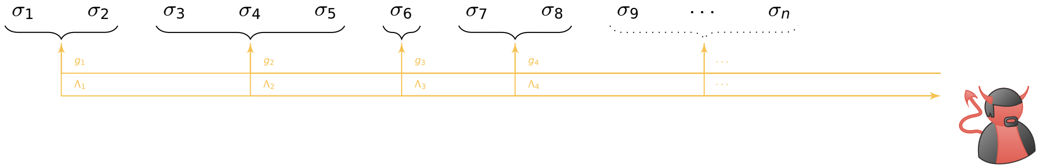
- The two most common kind of leakage resilient secret sharing schemes are against *independent* leakage attacks and *joint* leakage attacks.

Bounded leakage resilience — formal definition

$$\underbrace{\sigma_1 \quad \sigma_2} \quad \underbrace{\sigma_3 \quad \sigma_4 \quad \sigma_5} \quad \underbrace{\sigma_6} \quad \underbrace{\sigma_7 \quad \sigma_8} \quad \sigma_9 \quad \cdots \quad \sigma_n$$

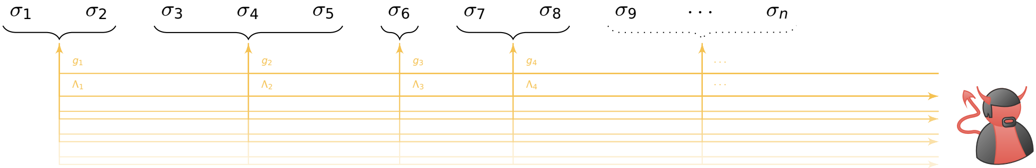
- The two most common kind of leakage resilient secret sharing schemes are against *independent* leakage attacks and *joint* leakage attacks.
- **Partitioning:** a *k-sized partition* of $[n]$ is a family $\mathcal{B}$ of disjoint subsets $(\mathcal{B}_1, \ldots, \mathcal{B}_m)$ of $[n]$ such that $\bigcup_{i=1}^{m} \mathcal{B}_i = [n]$ and, for all $i \in [m]$, $|\mathcal{B}_i| \leq k$.

Bounded leakage resilience — formal definition



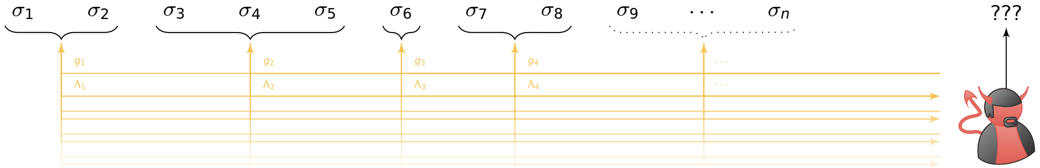
- The two most common kind of leakage resilient secret sharing schemes are against *independent* leakage attacks and *joint* leakage attacks.
- **Partitioning:** a *k-sized partition* of $[n]$ is a family $\mathcal{B}$ of disjoint subsets $(\mathcal{B}_1, \ldots, \mathcal{B}_m)$ of $[n]$ such that $\bigcup_{i=1}^{m} \mathcal{B}_i = [n]$ and, for all $i \in [m]$, $|\mathcal{B}_i| \leq k$.
- **Leakage queries:** a leakage query is a tuple of functions $(g_1, \ldots, g_m)$ such that, for all $i \in [m]$, g_i takes as input all the shares $(\sigma_j)_{j \in \mathcal{B}_i}$ and outputs a binary string $\Lambda \in \{0,1\}^{\ell_i}$.

Bounded leakage resilience — formal definition



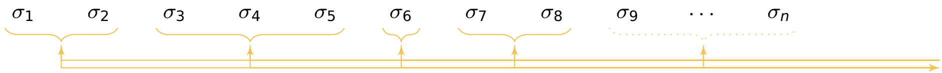
- The two most common kind of leakage resilient secret sharing schemes are against *independent* leakage attacks and *joint* leakage attacks.
- **Partitioning:** a *k-sized partition* of $[n]$ is a family $\mathcal{B}$ of disjoint subsets $(\mathcal{B}_1, \ldots, \mathcal{B}_m)$ of $[n]$ such that $\bigcup_{i=1}^{m} \mathcal{B}_i = [n]$ and, for all $i \in [m]$, $|\mathcal{B}_i| \leq k$.
- **Leakage queries:** a leakage query is a tuple of functions $(g_1, \ldots, g_m)$ such that, for all $i \in [m]$, g_i takes as input all the shares $(\sigma_j)_{j \in \mathcal{B}_i}$ and outputs a binary string $\Lambda \in \{0,1\}^{\ell_i}$.
- **Admissible adversaries:** an adversary A is (k, ℓ)-admissible if, fixed any k-partition, it is allowed to ask as many leakage queries he wants, chosen adaptively, as long as $|\Lambda^{(1)}| + \ldots + |\Lambda^{(q)}| \leq \ell$, where $\Lambda^{(i)}$ is the output of the i-th leakage query.

Bounded leakage resilience — formal definition



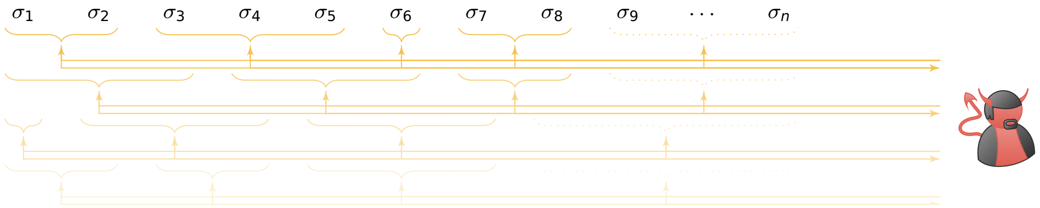
- The two most common kind of leakage resilient secret sharing schemes are against *independent* leakage attacks and *joint* leakage attacks.
- **Partitioning:** a *k-sized partition* of $[n]$ is a family $\mathcal{B}$ of disjoint subsets $(\mathcal{B}_1, \ldots, \mathcal{B}_m)$ of $[n]$ such that $\bigcup_{i=1}^{m} \mathcal{B}_i = [n]$ and, for all $i \in [m]$, $|\mathcal{B}_i| \leq k$.
- **Leakage queries:** a leakage query is a tuple of functions $(g_1, \ldots, g_m)$ such that, for all $i \in [m]$, g_i takes as input all the shares $(\sigma_j)_{j \in \mathcal{B}_i}$ and outputs a binary string $\Lambda \in \{0,1\}^{\ell_i}$.
- **Admissible adversaries:** an adversary A is (k, ℓ)-admissible if, fixed any k-partition, it is allowed to ask as many leakage queries he wants, chosen adaptively, as long as $|\Lambda^{(1)}| + \ldots + |\Lambda^{(q)}| \leq \ell$, where $\Lambda^{(i)}$ is the output of the i-th leakage query.
- **Leakage resilience:** a secret sharing scheme is k-joint ℓ-bounded ε-leakage-resilient against *selective partitioning* if, for all messages μ_0, μ_1, any (k, ℓ)-admissible adversary cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.

Bounded leakage resilience — adaptive partitioning



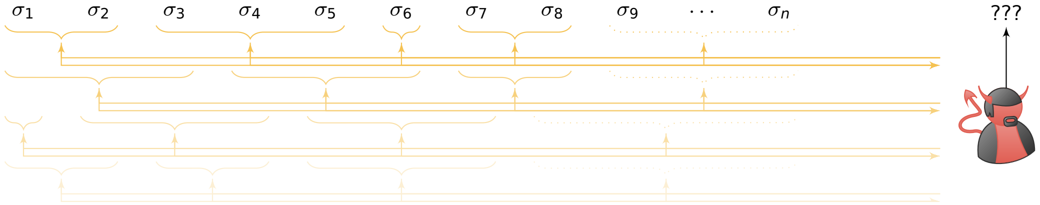
- **Leakage queries:** a leakage query is a pair $(\mathcal{B}, g)$, where $\mathcal{B}$ is a k-sized partition of $[n]$ and $g = (g_1, \ldots, g_m)$ is a tuple of functions such that, for all $i \in [m]$, g_i takes as input all the shares $(\sigma_j)_{j \in \mathcal{B}_i}$ and outputs a binary string $\Lambda \in \{0, 1\}^{\ell_i}$.

Bounded leakage resilience — adaptive partitioning



- **Leakage queries:** a leakage query is a pair $(\mathcal{B}, g)$, where $\mathcal{B}$ is a k-sized partition of $[n]$ and $g = (g_1, \ldots, g_m)$ is a tuple of functions such that, for all $i \in [m]$, g_i takes as input all the shares $(\sigma_j)_{j \in \mathcal{B}_i}$ and outputs a binary string $\Lambda \in \{0,1\}^{\ell_i}$.
- **Admissible adversaries:** an adversary A is (k, ℓ)-admissible if it is allowed to ask as many leakage queries he wants, chosen adaptively, as long as $|\Lambda^{(1)}| + \ldots + |\Lambda^{(q)}| \leq \ell$, where $\Lambda^{(i)}$ is the output of the i-th leakage query.

Bounded leakage resilience — adaptive partitioning



- **Leakage queries:** a leakage query is a pair $(\mathcal{B}, g)$, where $\mathcal{B}$ is a k-sized partition of $[n]$ and $g = (g_1, \ldots, g_m)$ is a tuple of functions such that, for all $i \in [m]$, g_i takes as input all the shares $(\sigma_j)_{j \in \mathcal{B}_i}$ and outputs a binary string $\Lambda \in \{0,1\}^{\ell_i}$.
- **Admissible adversaries:** an adversary A is (k, ℓ)-admissible if it is allowed to ask as many leakage queries he wants, chosen adaptively, as long as $|\Lambda^{(1)}| + \ldots + |\Lambda^{(q)}| \leq \ell$, where $\Lambda^{(i)}$ is the output of the i-th leakage query.
- **Leakage resilience:** a secret sharing scheme is k-joint ℓ-bounded ε-leakage-resilient against *adaptive partitioning* if, for all messages μ_0, μ_1, any (k, ℓ)-admissible adversary cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.

Noisy leakage resilience

- **Conditional average min-entropy [DORS08]:** defined as $\tilde{\mathbb{H}}_\infty(X|Y) := -\log\left(\mathbb{E}_{y\in\mathcal{Y}}\left[\max_{x\in\mathcal{X}} \mathbb{P}[X = x \mid Y = y]\right]\right)$.

[DORS08] "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data", *Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, Adam D. Smith*, SIAM Journal on Computing, Vol. 38, 2008

Noisy leakage resilience

- **Conditional average min-entropy [DORS08]:** defined as $\tilde{\mathbb{H}}_\infty(X|Y) := -\log\left(\mathbb{E}_{y\in\mathcal{Y}}\left[\max_{x\in\mathcal{X}} \mathbb{P}\left[X = x \mid Y = y\right]\right]\right)$.
- Informally, this value represents the best chanche to predict X by knowing Y, or, in a leakage perspective, the worst case for the min-entropy of X when an adversary knows Y.

[DORS08] “Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data”, *Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, Adam D. Smith*, SIAM Journal on Computing, Vol. 38, 2008

Noisy leakage resilience

- **Conditional average min-entropy [DORS08]:** defined as $\tilde{\mathbb{H}}_\infty(X|Y) := -\log\left(\mathbb{E}_{y\in\mathcal{Y}}\left[\max_{x\in\mathcal{X}} \mathbb{P}\left[X = x \mid Y = y\right]\right]\right)$.
- Informally, this value represents the best chanche to predict X by knowing Y, or, in a leakage perspective, the worst case for the min-entropy of X when an adversary knows Y.

- **Admissible adversaries:** an adversary A is (k, ℓ)-admissible, in the *noisy leakage* model, if it is allowed to ask as many leakage queries he wants, chosen adaptively, as long as, for all $i \in [m]$,

$$\tilde{\mathbb{H}}_\infty\left((\mathbf{\Sigma}_j)_{j\in\mathcal{B}_i} \mid \mathbf{\Lambda}_i\right) \geq \mathbb{H}_\infty\left((\mathbf{\Sigma}_j)_{j\in\mathcal{B}_i}\right) - \ell$$

[DORS08] "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data", *Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, Adam D. Smith*, SIAM Journal on Computing, Vol. 38, 2008

Noisy leakage resilience

- **Conditional average min-entropy [DORS08]:** defined as $\tilde{\mathbb{H}}_\infty(X|Y) := -\log\left(\mathbb{E}_{y\in\mathcal{Y}}\left[\max_{x\in\mathcal{X}} \mathbb{P}\left[X = x \mid Y = y\right]\right]\right)$.
- Informally, this value represents the best chanche to predict X by knowing Y, or, in a leakage perspective, the worst case for the min-entropy of X when an adversary knows Y.

- **Admissible adversaries:** an adversary A is (k,ℓ)-admissible, in the *noisy leakage* model, if it is allowed to ask as many leakage queries he wants, chosen adaptively, as long as, for all $i \in [m]$,

$$\tilde{\mathbb{H}}_\infty\left((\mathbf{\Sigma}_j)_{j\in\mathcal{B}_i} \mid \mathbf{\Lambda}_i\right) \geq \mathbb{H}_\infty\left((\mathbf{\Sigma}_j)_{j\in\mathcal{B}_i}\right) - \ell$$

- **Leakage resilience:** a secret sharing scheme is k-joint *ℓ-noisy ε-leakage-resilient against selective partitioning if, for all messages μ_0, μ_1, any (k,ℓ)-admissible adversary cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.*

[DORS08] "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data", *Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, Adam D. Smith*, SIAM Journal on Computing, Vol. 38, 2008

Noisy leakage resilience

- **Conditional average min-entropy [DORS08]:** defined as $\tilde{\mathbb{H}}_\infty(X|Y) := -\log\left(\mathbb{E}_{y\in\mathcal{Y}}\left[\max_{x\in\mathcal{X}} \mathbb{P}\left[X = x \mid Y = y\right]\right]\right)$.
- Informally, this value represents the best chanche to predict X by knowing Y, or, in a leakage perspective, the worst case for the min-entropy of X when an adversary knows Y.

- **Admissible adversaries:** an adversary A is (k, ℓ)-admissible, in the *noisy leakage* model, if it is allowed to ask as many leakage queries he wants, chosen adaptively, as long as, for all $i \in [m]$,

$$\tilde{\mathbb{H}}_\infty\left((\mathbf{\Sigma}_j)_{j\in\mathcal{B}_i} \mid \mathbf{\Lambda}_i\right) \geq \mathbb{H}_\infty\left((\mathbf{\Sigma}_j)_{j\in\mathcal{B}_i}\right) - \ell$$

- **Leakage resilience:** a secret sharing scheme is k-joint *ℓ-noisy ε-leakage-resilient against selective partitioning if, for all messages μ_0, μ_1, any (k, ℓ)-admissible adversary cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.*

Advantages of noisy leakage

- The output of the leakage functions can be longer.

[DORS08] "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data", *Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, Adam D. Smith*, SIAM Journal on Computing, Vol. 38, 2008

Noisy leakage resilience

- **Conditional average min-entropy [DORS08]:** defined as $\tilde{\mathbb{H}}_\infty(X|Y) := -\log\left(\mathbb{E}_{y\in\mathcal{Y}}\left[\max_{x\in\mathcal{X}}\mathbb{P}\left[X = x \mid Y = y\right]\right]\right)$.
- Informally, this value represents the best chanche to predict X by knowing Y, or, in a leakage perspective, the worst case for the min-entropy of X when an adversary knows Y.

- **Admissible adversaries:** an adversary A is (k,ℓ)-admissible, in the *noisy leakage* model, if it is allowed to ask as many leakage queries he wants, chosen adaptively, as long as, for all $i \in [m]$,

$$\tilde{\mathbb{H}}_\infty\left((\mathbf{\Sigma}_j)_{j\in\mathcal{B}_i} \mid \mathbf{\Lambda}_i\right) \geq \mathbb{H}_\infty\left((\mathbf{\Sigma}_j)_{j\in\mathcal{B}_i}\right) - \ell$$

- **Leakage resilience:** a secret sharing scheme is k-joint *ℓ-noisy ε-leakage-resilient against selective partitioning if, for all messages μ_0, μ_1, any (k,ℓ)-admissible adversary cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.*

Advantages of noisy leakage

- The output of the leakage functions can be longer.
- The maximum number of leakage queries is not bounded.

[DORS08] "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data", *Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, Adam D. Smith*, SIAM Journal on Computing, Vol. 38, 2008

Noisy leakage resilience

- **Conditional average min-entropy [DORS08]:** defined as $\tilde{\mathbb{H}}_\infty(X|Y) := -\log\left(\mathbb{E}_{y\in\mathcal{Y}}\left[\max_{x\in\mathcal{X}}\mathbb{P}\left[X=x \mid Y=y\right]\right]\right)$.
- Informally, this value represents the best chanche to predict X by knowing Y, or, in a leakage perspective, the worst case for the min-entropy of X when an adversary knows Y.

- **Admissible adversaries:** an adversary A is (k,ℓ)-admissible, in the *noisy leakage* model, if it is allowed to ask as many leakage queries he wants, chosen adaptively, as long as, for all $i \in [m]$,

$$\tilde{\mathbb{H}}_\infty\left((\mathbf{\Sigma}_j)_{j\in\mathcal{B}_i} \mid \mathbf{\Lambda}_i\right) \geq \mathbb{H}_\infty\left((\mathbf{\Sigma}_j)_{j\in\mathcal{B}_i}\right) - \ell$$

- **Leakage resilience:** a secret sharing scheme is k-joint *ℓ-noisy ε-leakage-resilient against selective partitioning if, for all messages μ_0, μ_1, any (k,ℓ)-admissible adversary cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.*

Advantages of noisy leakage

- The output of the leakage functions can be longer.
- The maximum number of leakage queries is not bounded.
- It is possible to ask the same leakage queries without wasting them.

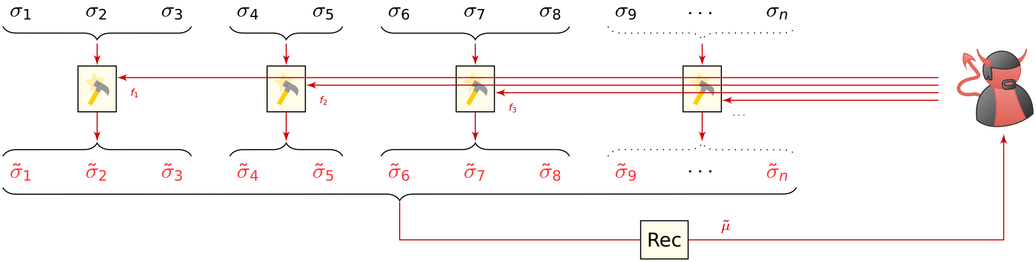
[DORS08] "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data", *Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, Adam D. Smith*, SIAM Journal on Computing, Vol. 38, 2008

Non-Malleability — formal definition

σ_1 σ_2 σ_3 σ_4 σ_5 σ_6 σ_7 σ_8 σ_9 $\cdots$ σ_n

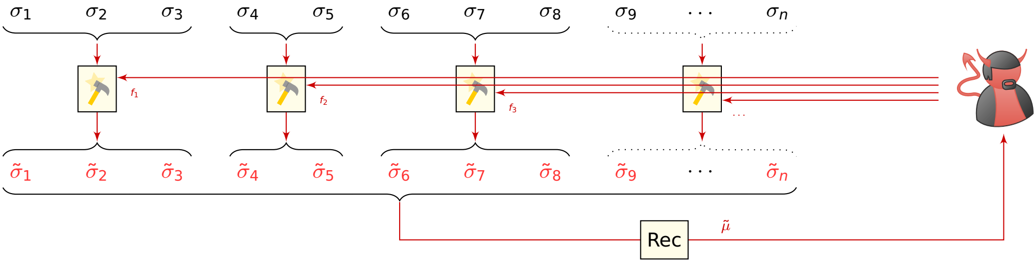
- The two most common kind of non-malleable secret sharing schemes are against *independent* tampering attacks and *joint* tampering attacks.

Non-Malleability — formal definition



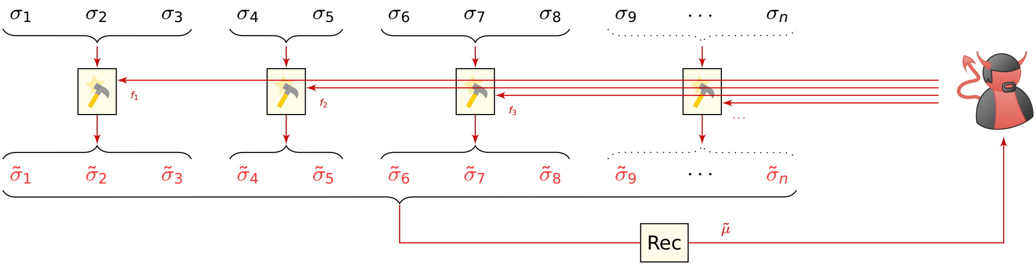
- The two most common kind of non-malleable secret sharing schemes are against *independent* tampering attacks and *joint* tampering attacks.
- **Tampering queries:** a tampering query is a pair $(\mathcal{T}, f)$, where $\mathcal{T} \subseteq [n], |\mathcal{T}| \geq t$ and $f = (f_1, \ldots, f_m)$ is a tuple of functions such that, for all $i \in [m]$, f_i takes as input all the shares $(\sigma_j)_{j \in \mathcal{B}_i}$ and outputs modified shares $(\tilde{\sigma}_j)_{j \in \mathcal{B}_i}$. The result of a tampering query is the reconstructed secret $\tilde{\mu} = \text{Rec}((\tilde{\sigma}_j)_{j \in \mathcal{T}})$.

Non-Malleability — formal definition



- The two most common kind of non-malleable secret sharing schemes are against *independent* tampering attacks and *joint* tampering attacks.
- **Tampering queries:** a tampering query is a pair $(\mathcal{T}, f)$, where $\mathcal{T} \subseteq [n], |\mathcal{T}| \geq t$ and $f = (f_1, \ldots, f_m)$ is a tuple of functions such that, for all $i \in [m]$, f_i takes as input all the shares $(\sigma_j)_{j\in\mathcal{B}_i}$ and outputs modified shares $(\tilde{\sigma}_j)_{j\in\mathcal{B}_i}$. The result of a tampering query is the reconstructed secret $\tilde{\mu} = \mathsf{Rec}((\tilde{\sigma}_j)_{j\in\mathcal{T}})$.
- **Non-malleability:** a secret sharing scheme is k-joint one-time ε-non-malleable if, for all messages μ_0, μ_1 and all k-sized partitions of $[n]$, any adversary performing one tampering query cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.

Non-Malleability — formal definition



- The two most common kind of non-malleable secret sharing schemes are against *independent* tampering attacks and *joint* tampering attacks.
- **Tampering queries:** a tampering query is a pair $(\mathcal{T}, f)$, where $\mathcal{T} \subseteq [n], |\mathcal{T}| \geq t$ and $f = (f_1, \dots, f_m)$ is a tuple of functions such that, for all $i \in [m]$, f_i takes as input all the shares $(\sigma_j)_{j \in \mathcal{B}_i}$ and outputs modified shares $(\tilde{\sigma}_j)_{j \in \mathcal{B}_i}$. The result of a tampering query is the reconstructed secret $\tilde{\mu} = \mathrm{Rec}((\tilde{\sigma}_j)_{j \in \mathcal{T}})$.
- **Non-malleability:** a secret sharing scheme is k-joint one-time ε-non-malleable if, for all messages μ_0, μ_1 and all k-sized partitions of $[n]$, any adversary performing one tampering query cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.
- To avoid trivial attacks, $\tilde{\mu}$ is set to a special symbol $\clubsuit$ whenever $\tilde{\mu} \in \{\mu_0, \mu_1\}$.

Non-malleability against multiple-queries

- p**-time non-malleability:** a secret sharing scheme is k-joint p-time ε-non-malleable if, for all messages μ_0, μ_1 and all k-sized partitions of $[n]$, any adversary performing up to p tampering queries cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.

Non-malleability against multiple-queries

- *p***-time non-malleability:** a secret sharing scheme is *k*-joint *p*-time ε-non-malleable if, for all messages μ_0, μ_1 and all *k*-sized partitions of $[n]$, any adversary performing up to *p* tampering queries cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.
- **Continuous non-malleability:** a secret sharing scheme is *k*-joint continuously non-malleable if, for all messages μ_0, μ_1 and all *k*-sized partitions of $[n]$, any *PPT* adversary performing any polynomial number of tampering queries cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with more than negligible advantage.

Non-malleability against multiple-queries

- ***p*-time non-malleability:** a secret sharing scheme is *k*-joint *p*-time ε-non-malleable if, for all messages μ_0, μ_1 and all *k*-sized partitions of $[n]$, any adversary performing up to *p* tampering queries cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.
- **Continuous non-malleability:** a secret sharing scheme is *k*-joint continuously non-malleable if, for all messages μ_0, μ_1 and all *k*-sized partitions of $[n]$, any *PPT* adversary performing any polynomial number of tampering queries cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with more than negligible advantage.

Limitations of continuous non-malleability

- **Mandatory self-destruct feature [FV19]:** once a tampering query yields an invalid value $\tilde{\mu} = \perp$, the tampering oracle self-destructs and no more tampering queries are allowed.

Non-malleability against multiple-queries

- ***p*-time non-malleability:** a secret sharing scheme is *k*-joint *p*-time ε-non-malleable if, for all messages μ_0, μ_1 and all *k*-sized partitions of $[n]$, any adversary performing up to *p* tampering queries cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.
- **Continuous non-malleability:** a secret sharing scheme is *k*-joint continuously non-malleable if, for all messages μ_0, μ_1 and all *k*-sized partitions of $[n]$, any *PPT* adversary performing any polynomial number of tampering queries cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with more than negligible advantage.

Limitations of continuous non-malleability

- **Mandatory self-destruct feature [FV19]:** once a tampering query yields an invalid value $\tilde{\mu} = \perp$, the tampering oracle self-destructs and no more tampering queries are allowed.
- Otherwise, it would be possible to indefinitively query the tampering oracle with functions that reads one bit at a time, thus recovering all the shares in clear.

Non-malleability against multiple-queries

- ***p*-time non-malleability:** a secret sharing scheme is *k*-joint *p*-time ε-non-malleable if, for all messages μ_0, μ_1 and all *k*-sized partitions of [*n*], any adversary performing up to *p* tampering queries cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.
- **Continuous non-malleability:** a secret sharing scheme is *k*-joint continuously non-malleable if, for all messages μ_0, μ_1 and all *k*-sized partitions of [*n*], any *PPT* adversary performing any polynomial number of tampering queries cannot distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with more than negligible advantage.

Limitations of continuous non-malleability

- **Mandatory self-destruct feature [FV19]:** once a tampering query yields an invalid value $\tilde{\mu} = \perp$, the tampering oracle self-destructs and no more tampering queries are allowed.
- Otherwise, it would be possible to indefinitively query the tampering oracle with functions that reads one bit at a time, thus recovering all the shares in clear.
- **Only achievable against computationally bounded adversaries [FV19]:** *next slide*.

[FV19] "Non-Malleable Secret Sharing in the Computational Setting: Adaptive Tampering, Noisy-Leakage Resilience, and Improved Rate", *Antonio Faonio, Daniele Venturi*, CRYPTO 2019

Shared-value uniqueness

Definition

A t-out-of-n secret sharing scheme satisfies *shared-value uniqueness* if, for all subsets $\{i_1, \ldots, i_t\} \subseteq [n]$ of indices, there exists j^* such that, for all shares $\sigma_{i_1}, \ldots, \sigma_{i_{j^*-1}}, \sigma_{i_{j^*+1}}, \ldots, \sigma_{i_t}$ and for all $\sigma_{i_{j^*}}, \sigma'_{i_{j^*}}$, either

$$\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_{j^*}}, \ldots, \sigma_{i_t}) = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma'_{i_{j^*}}, \ldots, \sigma_{i_t}) = \mu',$$

with $\mu, \mu' \in \mathcal{M}$, or at least one of μ, μ' is invalid.

Shared-value uniqueness

Definition

A t-out-of-n secret sharing scheme satisfies *shared-value uniqueness* if, for all subsets $\{i_1, \ldots, i_t\} \subseteq [n]$ of indices, there exists j^* such that, for all shares $\sigma_{i_1}, \ldots, \sigma_{i_{j^*-1}}, \sigma_{i_{j^*+1}}, \ldots, \sigma_{i_t}$ and for all $\sigma_{i_{j^*}}, \sigma'_{i_{j^*}}$, either

$$\mu = \mathrm{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_{j^*}}, \ldots, \sigma_{i_t}) = \mathrm{Rec}(\sigma_{i_1}, \ldots, \sigma'_{i_{j^*}}, \ldots, \sigma_{i_t}) = \mu',$$

with $\mu, \mu' \in \mathcal{M}$, or at least one of μ, μ' is invalid.

- **Theorem [BS19]:** Any *independent* (i.e. 1-joint) continuously non-malleable secret sharing scheme must satisfy shared-value uniqueness.

[BS19] “Revisiting Non-Malleable Secret Sharing”, *Saikrishna Badrinarayanan, Akshayaram Srinivasan*, EUROCRYPT 2019

Shared-value uniqueness

Definition

A t-out-of-n secret sharing scheme satisfies *shared-value uniqueness* if, for all subsets $\{i_1, \ldots, i_t\} \subseteq [n]$ of indices, there exists j^* such that, for all shares $\sigma_{i_1}, \ldots, \sigma_{i_{j^*-1}}, \sigma_{i_{j^*+1}}, \ldots, \sigma_{i_t}$ and for all $\sigma_{i_{j^*}}, \sigma'_{i_{j^*}}$, either

$$\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_{j^*}}, \ldots, \sigma_{i_t}) = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma'_{i_{j^*}}, \ldots, \sigma_{i_t}) = \mu',$$

with $\mu, \mu' \in \mathcal{M}$, or at least one of μ, μ' is invalid.

- **Theorem [BS19]:** Any *independent* (i.e. 1-joint) continuously non-malleable secret sharing scheme must satisfy shared-value uniqueness.
- *Proof idea.* If the above does not hold, then there exist a set of indices $\mathcal{I}$ such that, for all $j^* \in \mathcal{I}$, it is possible to find shares $\sigma_{i_1}, \ldots, \sigma_{i_{j^*-1}}, \sigma_{i_{j^*+1}}, \ldots, \sigma_{i_t}$ for which the reconstructed values μ and μ' are both valid and distinct. Therefore, any adversary may exploit this fact in order to learn, for all $j^* \in [t]$, the shares $\sigma^*_{i_{j^*}}$ of the target secret sharing.

[BS19] "Revisiting Non-Malleable Secret Sharing", *Saikrishna Badrinarayanan, Akshayaram Srinivasan*, EUROCRYPT 2019

Shared-value uniqueness

Definition

A t-out-of-n secret sharing scheme satisfies *shared-value uniqueness* if, for all subsets $\{i_1, \ldots, i_t\} \subseteq [n]$ of indices, there exists j^* such that, for all shares $\sigma_{i_1}, \ldots, \sigma_{i_{j^*-1}}, \sigma_{i_{j^*+1}}, \ldots, \sigma_{i_t}$ and for all $\sigma_{i_{j^*}}, \sigma'_{i_{j^*}}$, either

$$\mu = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma_{i_{j^*}}, \ldots, \sigma_{i_t}) = \mathsf{Rec}(\sigma_{i_1}, \ldots, \sigma'_{i_{j^*}}, \ldots, \sigma_{i_t}) = \mu',$$

with $\mu, \mu' \in \mathcal{M}$, or at least one of μ, μ' is invalid.

- **Theorem [BS19]:** Any *independent* (i.e. 1-joint) continuously non-malleable secret sharing scheme must satisfy shared-value uniqueness.
- *Proof idea.* If the above does not hold, then there exist a set of indices $\mathcal{I}$ such that, for all $j^* \in \mathcal{I}$, it is possible to find shares $\sigma_{i_1}, \ldots, \sigma_{i_{j^*-1}}, \sigma_{i_{j^*+1}}, \ldots, \sigma_{i_t}$ for which the reconstructed values μ and μ' are both valid and distinct. Therefore, any adversary may exploit this fact in order to learn, for all $j^* \in [t]$, the shares $\sigma^*_{i_{j^*}}$ of the target secret sharing.
- On the other side, any secret sharing scheme satisfying statistical privacy must violate the above property, otherwise there would exist a setting in which the distribution $(\mathbf{\Sigma}_{i_1}, \ldots, \mathbf{\Sigma}_{i_{j^*}}, \ldots, \mathbf{\Sigma}_{i_t})$ only assumes valid values for at most one single message μ.

[BS19] "Revisiting Non-Malleable Secret Sharing", *Saikrishna Badrinarayanan, Akshayaram Srinivasan*, EUROCRYPT 2019

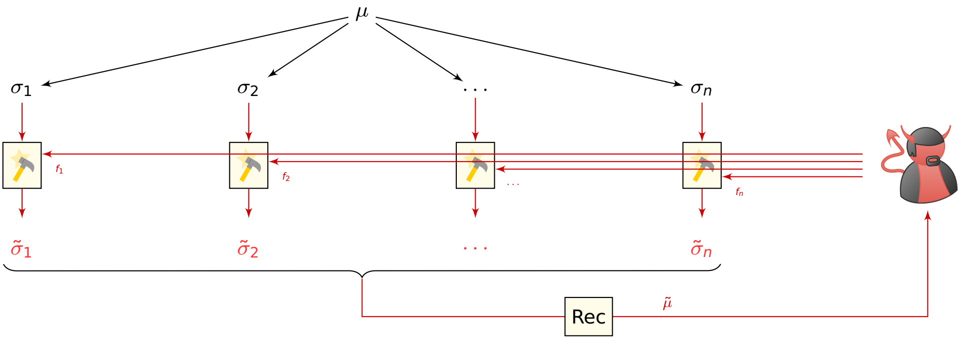
Split-state non-malleable codes

$$
\begin{tikzcd}
 & & \mu \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & \\
\sigma_1 & \sigma_2 & \ldots & \sigma_n
\end{tikzcd}
$$

- ***n*-split-state non-malleable code [DPW09]:** encodes a message μ into a codeword $(\sigma_1, \ldots, \sigma_n)$.

[DPW09] “Non-Malleable Codes”, *Stefan Dziembowski, Krzysztof Pietrzak, Daniel Wichs*, IACR Cryptology ePrint Archive, Vol.2009/608

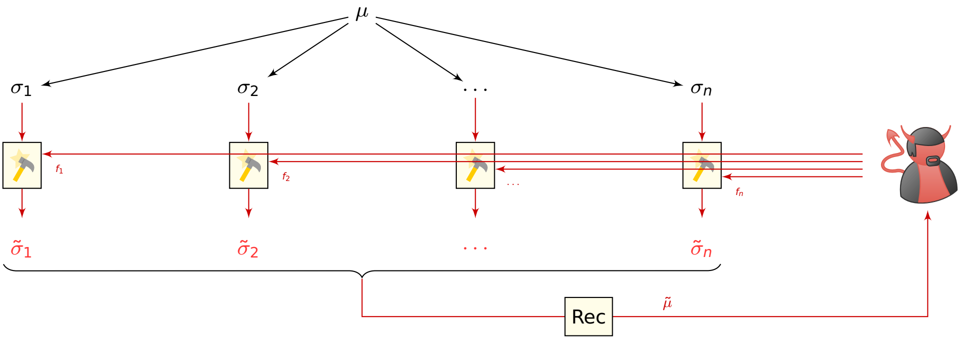
Split-state non-malleable codes



- ***n*-split-state non-malleable code [DPW09]:** encodes a message μ into a codeword $(\sigma_1, \ldots, \sigma_n)$.
- **Tampering:** a tampering query is a tuple of functions $(f_1, \ldots, f_n)$ such that, for each $i \in [n]$, $f_i(\sigma_i) = \tilde{\sigma}_i$. The result of a tampering query is the reconstructed message $\tilde{\mu} = \text{Rec}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n)$.

[DPW09] "Non-Malleable Codes", *Stefan Dziembowski, Krzysztof Pietrzak, Daniel Wichs*, IACR Cryptology ePrint Archive, Vol.2009/608

Split-state non-malleable codes



- ***n*-split-state non-malleable code [DPW09]:** encodes a message μ into a codeword $(\sigma_1, \ldots, \sigma_n)$.
- **Tampering:** a tampering query is a tuple of functions $(f_1, \ldots, f_n)$ such that, for each $i \in [n]$, $f_i(\sigma_i) = \tilde{\sigma}_i$. The result of a tampering query is the reconstructed message $\tilde{\mu} = \text{Rec}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n)$.
- **Non-malleability:** the result $\tilde{\mu}$ of the tampering either equals μ or it is completely unrelated.

[DPW09] "Non-Malleable Codes", *Stefan Dziembowski, Krzysztof Pietrzak, Daniel Wichs*, IACR Cryptology ePrint Archive, Vol.2009/608

Split-state non-malleable codes

- Often used to construct non-malleable secret sharing schemes.

Split-state non-malleable codes

- Often used to construct non-malleable secret sharing schemes.
- May have leakage-resilience as well.

Split-state non-malleable codes

- Often used to construct non-malleable secret sharing schemes.
- May have leakage-resilience as well.
- Commonly obtained from efficiently-invertible n-source non-malleable randomness extractors.

Split-state non-malleable codes

- Often used to construct non-malleable secret sharing schemes.
- May have leakage-resilience as well.
- Commonly obtained from efficiently-invertible *n*-source non-malleable randomness extractors.

- 2-split-state non-malleable codes are 2-out-of-2 non-malleable secret sharing schemes.

Split-state non-malleable codes

- Often used to construct non-malleable secret sharing schemes.
- May have leakage-resilience as well.
- Commonly obtained from efficiently-invertible n-source non-malleable randomness extractors.

- 2-split-state non-malleable codes are 2-out-of-2 non-malleable secret sharing schemes.
- *Proof.* Correctness and non-malleability follow immediately; privacy follows because, for a tampering query (f_1, f_2), the function f_1 may distinguish between an encoding of μ_0 and an encoding of μ_1 from the codeword portion σ_1 and may act as the identity function if the encoded message is μ_0 or replace σ_1 with a random string otherwise.

Split-state non-malleable codes

- Often used to construct non-malleable secret sharing schemes.
- May have leakage-resilience as well.
- Commonly obtained from efficiently-invertible n-source non-malleable randomness extractors.

- 2-split-state non-malleable codes are 2-out-of-2 non-malleable secret sharing schemes.
- *Proof.* Correctness and non-malleability follow immediately; privacy follows because, for a tampering query (f_1, f_2), the function f_1 may distinguish between an encoding of μ_0 and an encoding of μ_1 from the codeword portion σ_1 and may act as the identity function if the encoded message is μ_0 or replace σ_1 with a random string otherwise.
- n-split-state non-malleable codes for $n > 2$ are **not** n-out-of-n non-malleable secret sharing schemes.

Split-state non-malleable codes

- Often used to construct non-malleable secret sharing schemes.
- May have leakage-resilience as well.
- Commonly obtained from efficiently-invertible n-source non-malleable randomness extractors.

- 2-split-state non-malleable codes are 2-out-of-2 non-malleable secret sharing schemes.
- *Proof.* Correctness and non-malleability follow immediately; privacy follows because, for a tampering query (f_1, f_2), the function f_1 may distinguish between an encoding of μ_0 and an encoding of μ_1 from the codeword portion σ_1 and may act as the identity function if the encoded message is μ_0 or replace σ_1 with a random string otherwise.
- n-split-state non-malleable codes for $n > 2$ are **not** n-out-of-n non-malleable secret sharing schemes.
- *Proof.* Consider a 2-split-state non-malleable code $\mathsf{Share}(\mu) = (\sigma_1, \sigma_2)$ and construct the algorithm $\mathsf{Share}^*(\mu) = (\sigma_1^*, \sigma_2^*, \sigma_3^*)$, where $\sigma_1^* = \sigma_1, \sigma_2^* = \sigma_2$ and $\sigma_3^* = \mu$. Then, Share^* is a 3-split-state non-malleable code, but it is not a 3-out-of-3 non-malleable secret sharing scheme (privacy does not hold).

Augmented property

- Sometimes, it is useful that security holds even when an attacker is able to see in full all the shares in a subset $\mathcal{B}_i$ of the k-sized partition $\mathcal{B}$ after the leakage/tampering phase.

Augmented property

- Sometimes, it is useful that security holds even when an attacker is able to see in full all the shares in a subset $\mathcal{B}_i$ of the k-sized partition $\mathcal{B}$ after the leakage/tampering phase.
- When leakage is allowed after the last tampering query, this property comes at the cost of only 1 extra bit of leakage.

Augmented property

- Sometimes, it is useful that security holds even when an attacker is able to see in full all the shares in a subset $\mathcal{B}_i$ of the k-sized partition $\mathcal{B}$ after the leakage/tampering phase.
- When leakage is allowed after the last tampering query, this property comes at the cost of only 1 extra bit of leakage.
- *Proof.* By reduction to security of the same scheme without the augmented property.
  - Suppose that there exists an adversary $\mathsf{A}^+ = (\mathsf{A}_1^+, \mathsf{A}_2^+)$ that is able to distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.

Augmented property

- Sometimes, it is useful that security holds even when an attacker is able to see in full all the shares in a subset $\mathcal{B}_i$ of the k-sized partition $\mathcal{B}$ after the leakage/tampering phase.
- When leakage is allowed after the last tampering query, this property comes at the cost of only 1 extra bit of leakage.
- *Proof.* By reduction to security of the same scheme without the augmented property.
  - Suppose that there exists an adversary $\mathsf{A}^+ = (\mathsf{A}_1^+, \mathsf{A}_2^+)$ that is able to distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.
  - Consider the reduction A that forwards all the leakage (and tampering) queries of A_1^+, returning the corresponding answer to A_1^+.

Augmented property

- Sometimes, it is useful that security holds even when an attacker is able to see in full all the shares in a subset $\mathcal{B}_i$ of the k-sized partition $\mathcal{B}$ after the leakage/tampering phase.
- When leakage is allowed after the last tampering query, this property comes at the cost of only 1 extra bit of leakage.
- *Proof.* By reduction to security of the same scheme without the augmented property.
    - Suppose that there exists an adversary $\mathsf{A}^+ = (\mathsf{A}_1^+, \mathsf{A}_2^+)$ that is able to distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.
    - Consider the reduction A that forwards all the leakage (and tampering) queries of A_1^+, returning the corresponding answer to A_1^+.
    - When A_1^+ outputs the index i^* of the subset of shares he wants to see, construct the leakage query that takes as input all the shares in $\mathcal{B}_{i^*}$, feeds them into the algorithm A_2^+ and outputs the same distinguishing bit as A_2^+.

Augmented property

- Sometimes, it is useful that security holds even when an attacker is able to see in full all the shares in a subset $\mathcal{B}_i$ of the k-sized partition $\mathcal{B}$ after the leakage/tampering phase.
- When leakage is allowed after the last tampering query, this property comes at the cost of only 1 extra bit of leakage.
- *Proof.* By reduction to security of the same scheme without the augmented property.
    - Suppose that there exists an adversary $\mathsf{A}^+ = (\mathsf{A}_1^+, \mathsf{A}_2^+)$ that is able to distinguish between a secret sharing of μ_0 and a secret sharing of μ_1 with advantage more than ε.
    - Consider the reduction A that forwards all the leakage (and tampering) queries of A_1^+, returning the corresponding answer to A_1^+.
    - When A_1^+ outputs the index i^* of the subset of shares he wants to see, construct the leakage query that takes as input all the shares in $\mathcal{B}_{i^*}$, feeds them into the algorithm A_2^+ and outputs the same distinguishing bit as A_2^+.
    - Send the above leakage query to the oracle and, upon receiving an answer $b \in \{0, 1\}$, output the same b.