Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model

Gianluca Brian
Sapienza University of Rome
Rome, Italy

Antonio Faonio
IMDEA Software Institute
Madrid, Spain
(Now at EUROCOM)

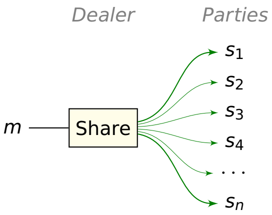
Maciej Obremski
National University of Singapore
Singapore, Singapore

Mark Simkin
Aarhus University
Aarhus, Denmark

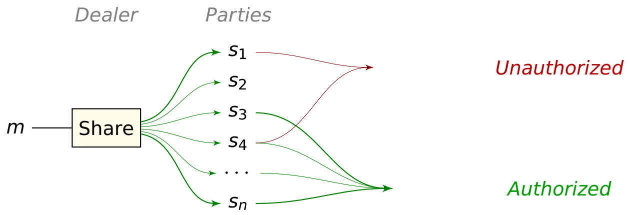
Daniele Venturi
Sapienza University of Rome
Rome, Italy

CRYPTO 2020
Online version

Secret Sharing

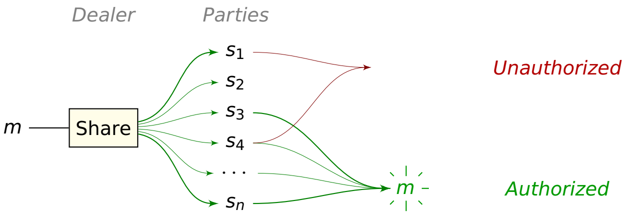


Secret Sharing

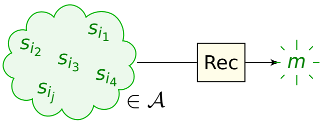


- **Access structure:** t-out-of-n

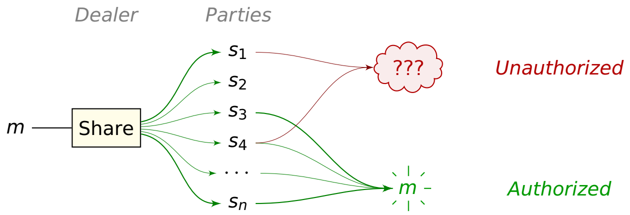
Secret Sharing



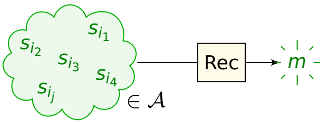
- **Access structure:** t-out-of-n
- **Correctness:** at least t parties are able to reconstruct the secret.

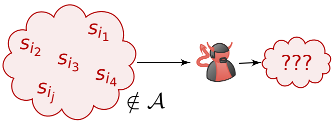


Secret Sharing

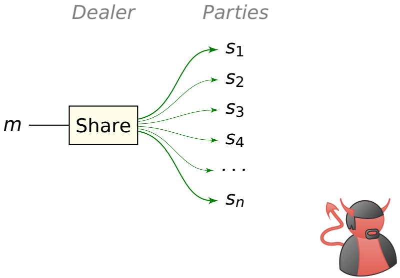


- **Access structure:** t-out-of-n
- **Correctness:** at least t parties are able to reconstruct the secret.
- **Privacy:** less than t parties should not be able to learn any information about the secret.

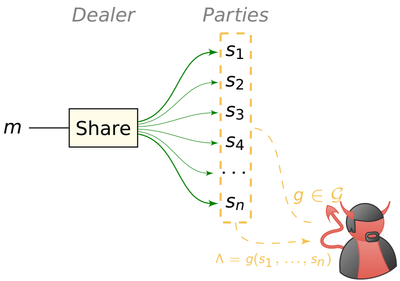




Leakage Resilient and Non-malleable Secret Sharing



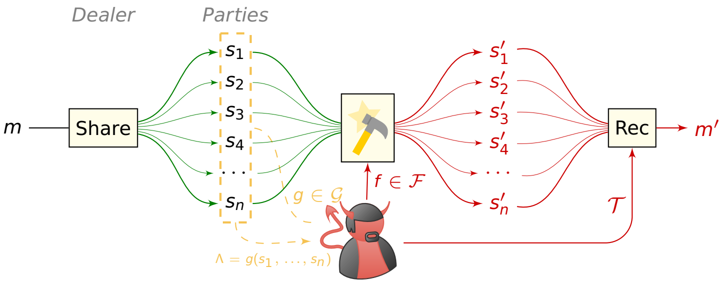
Leakage Resilient and Non-malleable Secret Sharing



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

SECURITY BREACH!

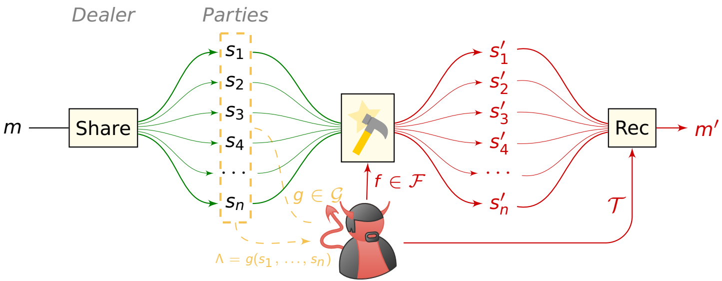
Leakage Resilient and Non-malleable Secret Sharing



Side channel attacks: partial information from all the shares may reveal some information about the message!
Tampering attacks: m' may be related to m!

SECURITY BREACH!!!

Leakage Resilient and Non-malleable Secret Sharing

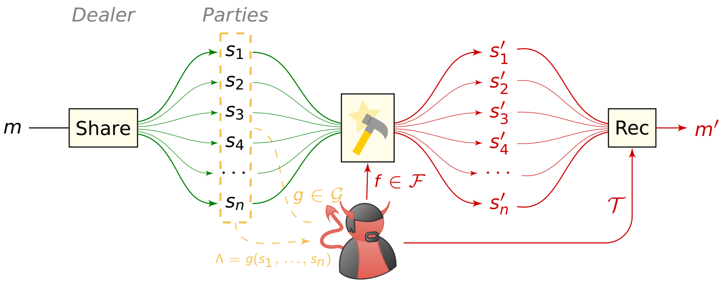


Side channel attacks: partial information from all the shares may reveal some information about the message!
Tampering attacks: m' may be related to m!

SECURITY BREACH!!!

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

Leakage Resilient and Non-malleable Secret Sharing

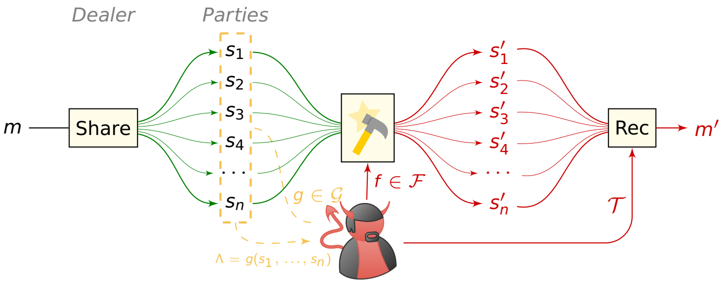


Side channel attacks: partial information from all the shares may reveal some information about the message!
Tampering attacks: m' may be related to m!

SECURITY BREACH!!!

Leakage Resilient Secret Sharing [KMS18] **:** Λ reveals nothing about m for a restricted family $\mathcal{G}$.
Non-Malleable Secret Sharing [GK18] **:** m' is unrelated to m for a restricted family $\mathcal{F}$.

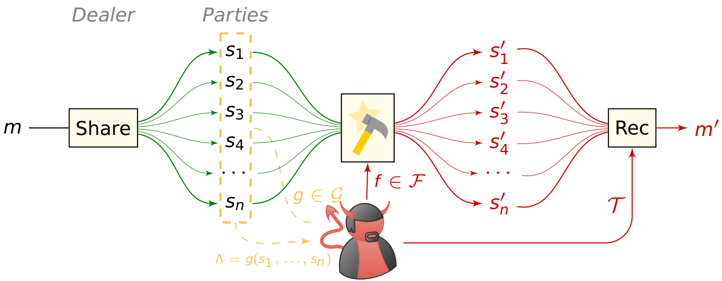
Leakage Resilient and Non-malleable Secret Sharing



Side channel attacks: partial information from all the shares may reveal some information about the message!
Tampering attacks: m' may be related to m!

SECURITY BREACH!!!

Leakage Resilient Secret Sharing [KMS18] **:** Λ reveals nothing about m for a restricted family $\mathcal{G}$.
Non-Malleable Secret Sharing [GK18] **:** m' is unrelated to m for a restricted family $\mathcal{F}$.
Leakage-resilient non-malleability: the best of both worlds.



Side channel attacks: partial information from all the shares may reveal some information about the message!
Tampering attacks: m' may be related to m!

SECURITY BREACH!!!

Leakage Resilient Secret Sharing [KMS18] **:** Λ reveals nothing about m for a restricted family $\mathcal{G}$.
Non-Malleable Secret Sharing [GK18] **:** m' is unrelated to m for a restricted family $\mathcal{F}$.
Leakage-resilient non-malleability: the best of both worlds.
Limitations: Impossible for arbitrary families $\mathcal{G}$ and $\mathcal{F}$.

Our contributions

Our model

- Joint leakage and tampering (selective partitioning, semi-adaptive partitioning).

Our contributions

Our model

- Joint leakage and tampering (selective partitioning, semi-adaptive partitioning).
- Bounded leakage: the total leakage amounts to at most ℓ bits.

Our contributions

Our model

- Joint leakage and tampering (selective partitioning, semi-adaptive partitioning).
- Bounded leakage: the total leakage amounts to at most ℓ bits.

Selective partitioning

- Any one-time statistically non-malleable secret sharing scheme is also leakage resilient.

Our contributions

Our model

- Joint leakage and tampering (selective partitioning, semi-adaptive partitioning).
- Bounded leakage: the total leakage amounts to at most ℓ bits.

Selective partitioning

- Any one-time statistically non-malleable secret sharing scheme is also leakage resilient.
- **Corollary:** lower bounds for the size of the shares of non-malleable secret sharing schemes using [NS20].

Our contributions

Our model

- Joint leakage and tampering (selective partitioning, semi-adaptive partitioning).
- Bounded leakage: the total leakage amounts to at most ℓ bits.

Selective partitioning

- Any one-time statistically non-malleable secret sharing scheme is also leakage resilient.
- **Corollary:** lower bounds for the size of the shares of non-malleable secret sharing schemes using [NS20].

Semi-adaptive partitioning

- We construct a one-time non-malleable secret-sharing scheme against joint leakage and tampering under semi-adaptive partitioning.

Our contributions

Our model

- Joint leakage and tampering (selective partitioning, semi-adaptive partitioning).
- Bounded leakage: the total leakage amounts to at most ℓ bits.

Selective partitioning

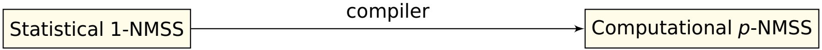
- Any one-time statistically non-malleable secret sharing scheme is also leakage resilient.
- **Corollary:** lower bounds for the size of the shares of non-malleable secret sharing schemes using [NS20].

Semi-adaptive partitioning

- We construct a one-time non-malleable secret-sharing scheme against joint leakage and tampering under semi-adaptive partitioning.

Both settings

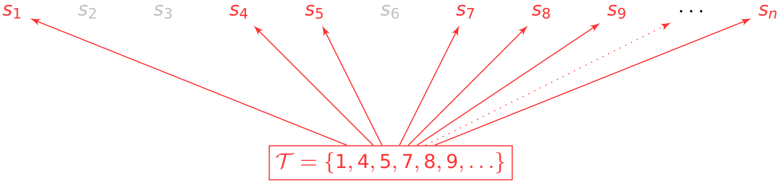
- **Corollary:** construction of a p-time non-malleable secret sharing scheme from known techniques [OPVV18, BFV19].



Security against selective partitioning

s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 $\cdots$ s_n

Security against selective partitioning

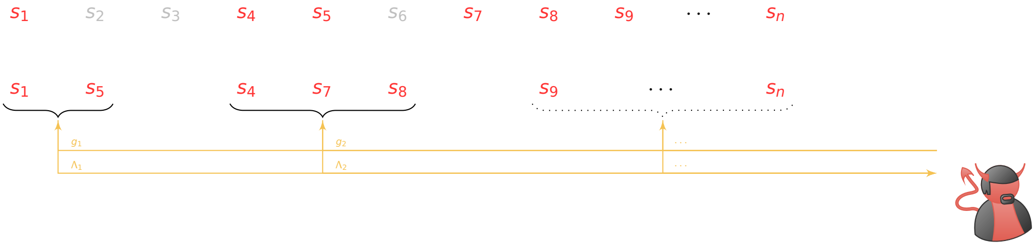


Security against selective partitioning

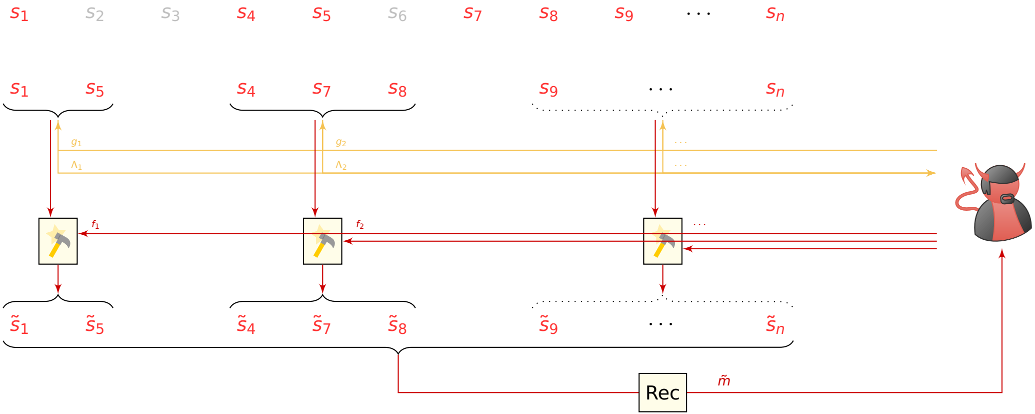
s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 $\cdots$ s_n

$\underbrace{s_1 \quad s_5}$ $\underbrace{s_4 \quad s_7 \quad s_8}$ $s_9 \quad \cdots \quad s_n$

Security against selective partitioning

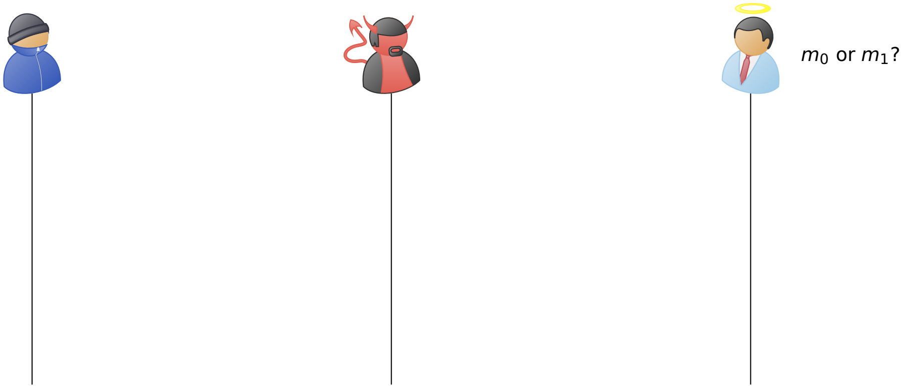


Security against selective partitioning



A non-malleable secret sharing is also leakage resilient

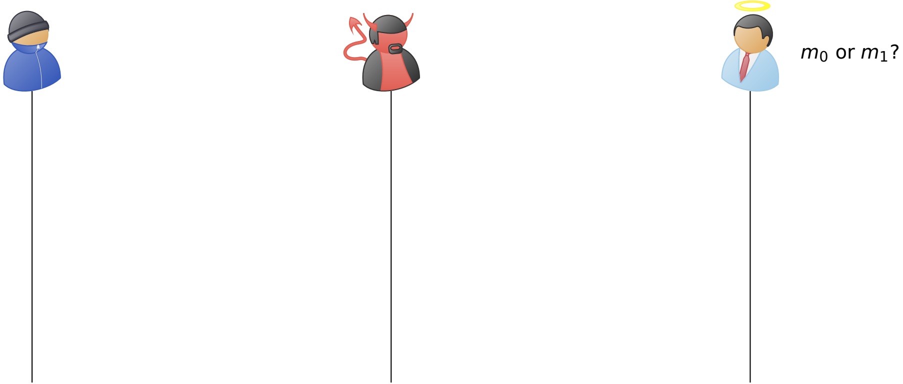
Any one-time $\epsilon/2^{\ell}$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.



A non-malleable secret sharing is also leakage resilient

Any one-time $\epsilon/2^{\ell}$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.

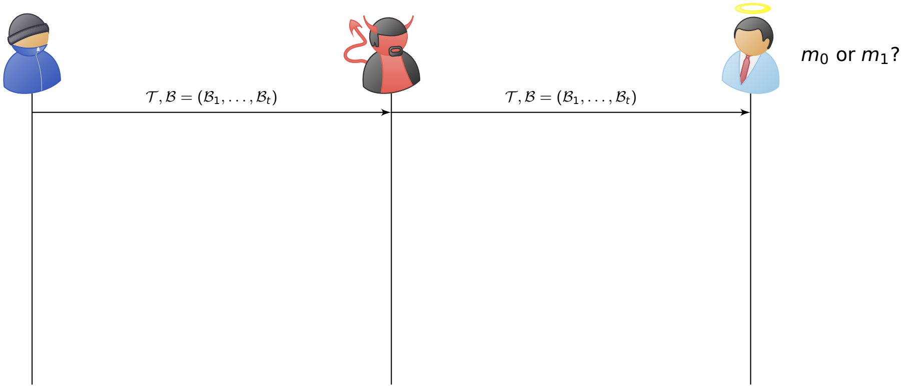
Proof strategy: complexity leveraging.



A non-malleable secret sharing is also leakage resilient

Any one-time $\epsilon/2^{\ell}$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.

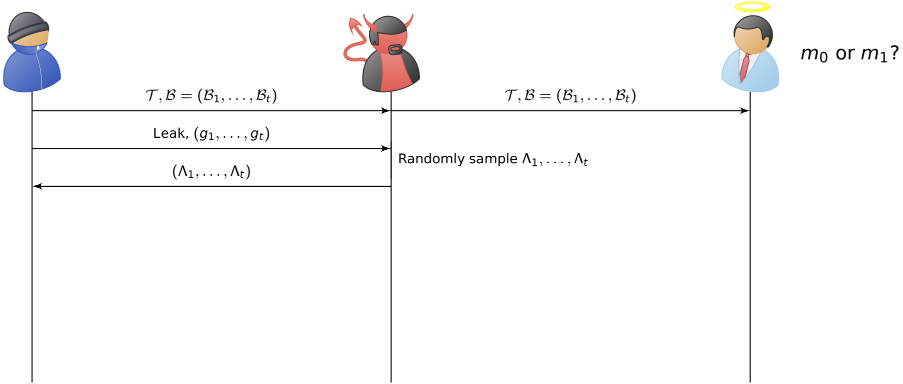
Proof strategy: complexity leveraging.



A non-malleable secret sharing is also leakage resilient

Any one-time $\epsilon/2^{\ell}$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.

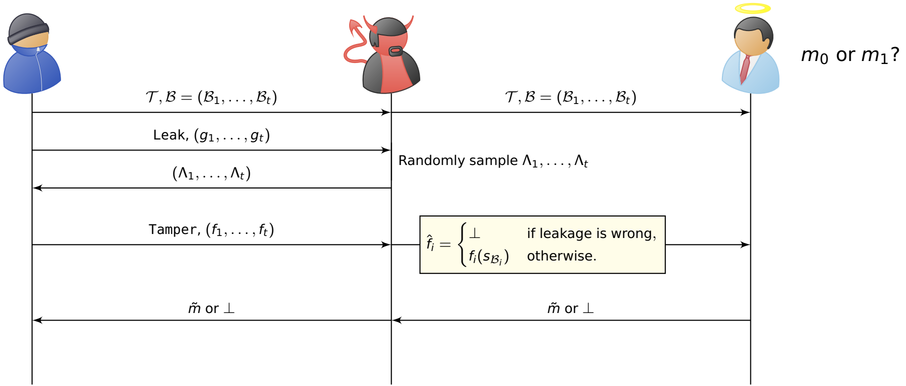
Proof strategy: complexity leveraging.



A non-malleable secret sharing is also leakage resilient

Any one-time $\epsilon/2^{\ell}$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.

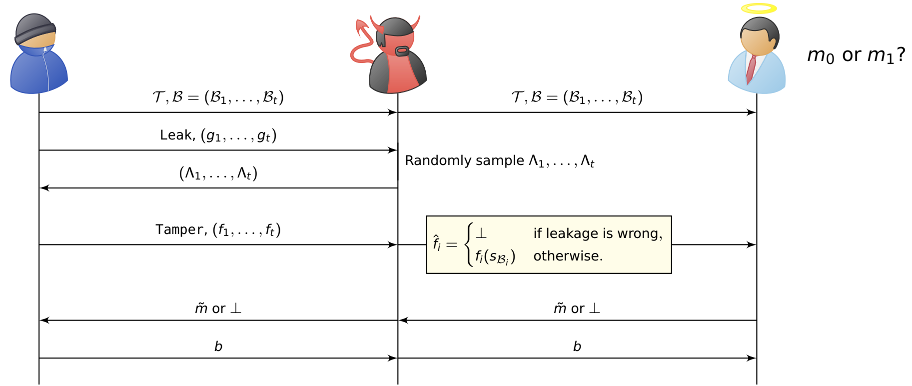
Proof strategy: complexity leveraging.



A non-malleable secret sharing is also leakage resilient

Any one-time $\epsilon/2^{\ell}$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.

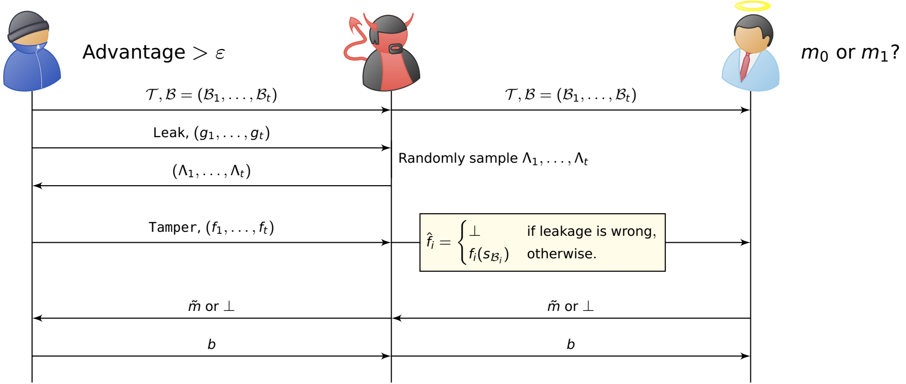
Proof strategy: complexity leveraging.



A non-malleable secret sharing is also leakage resilient

Any one-time $\epsilon/2^{\ell}$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.

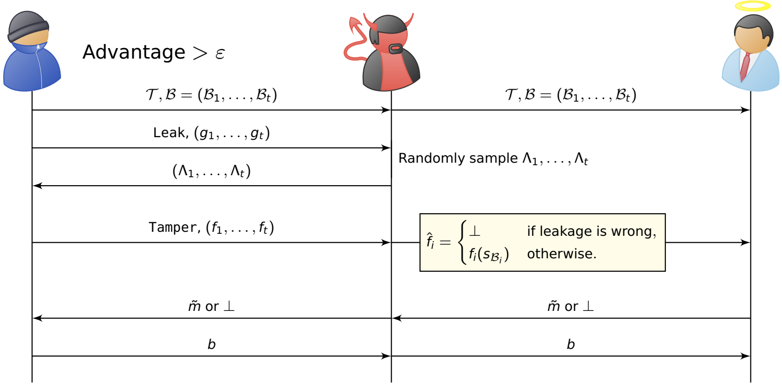
Proof strategy: complexity leveraging.



A non-malleable secret sharing is also leakage resilient

Any one-time $\epsilon/2^{\ell}$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.

Proof strategy: complexity leveraging.



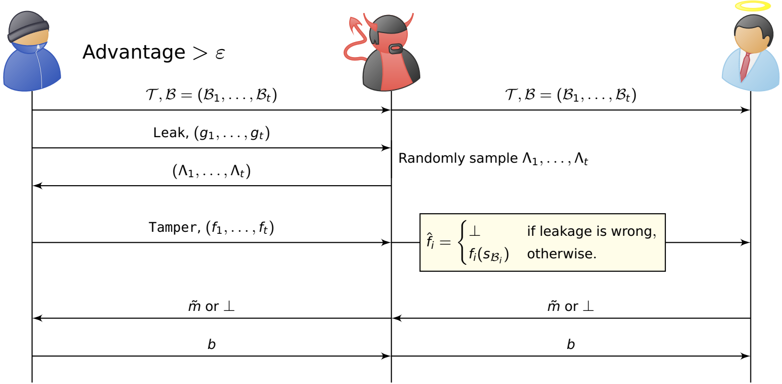
Guess $\Longrightarrow$ perfect sim.

Leakage and tampering answers are correct

A non-malleable secret sharing is also leakage resilient

Any one-time $\epsilon/2^\ell$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.

Proof strategy: complexity leveraging.



m_0 or m_1?

Guess $\Longrightarrow$ perfect sim.

Leakage and tampering answers are correct

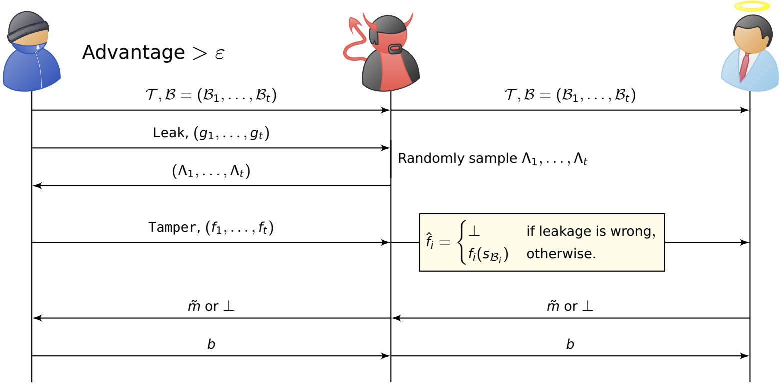
$\neg$**Guess** $\Longrightarrow$ no advantage

The view of the adversary is independent of m_0, m_1

A non-malleable secret sharing is also leakage resilient

Any one-time $\epsilon/2^{\ell}$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.

Proof strategy: complexity leveraging.



m_0 or m_1?

Guess $\implies$ perfect sim.

Leakage and tampering answers are correct

$\neg$**Guess** $\implies$ no advantage

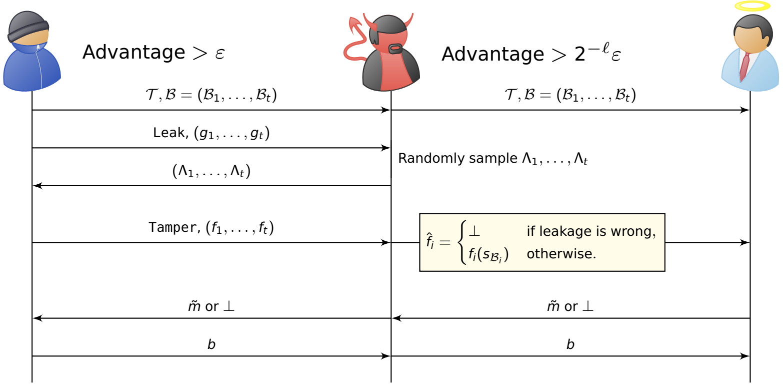
The view of the adversary is independent of m_0, m_1

$\mathbb{P}\,[\textbf{Guess}] = 2^{-\ell}$

A non-malleable secret sharing is also leakage resilient

Any one-time $\epsilon/2^{\ell}$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.

Proof strategy: complexity leveraging.



Guess $\Longrightarrow$ perfect sim.

Leakage and tampering answers are correct

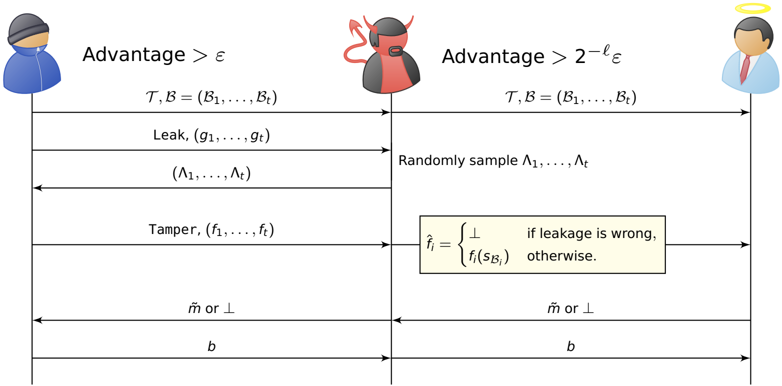
$\neg$**Guess** $\Longrightarrow$ no advantage

The view of the adversary is independent of m_0, m_1

$\mathbb{P}\,[\textbf{Guess}] = 2^{-\ell}$

Any one-time $\epsilon/2^{\ell}$-non-malleable secret sharing scheme is also a ℓ-bounded leakage resilient one-time ϵ-non-malleable secret sharing scheme.

Proof strategy: complexity leveraging.



m_0 or m_1?

Guess $\implies$ perfect sim.

Leakage and tampering answers are correct

$\neg$**Guess** $\implies$ no advantage

The view of the adversary is independent of m_0, m_1

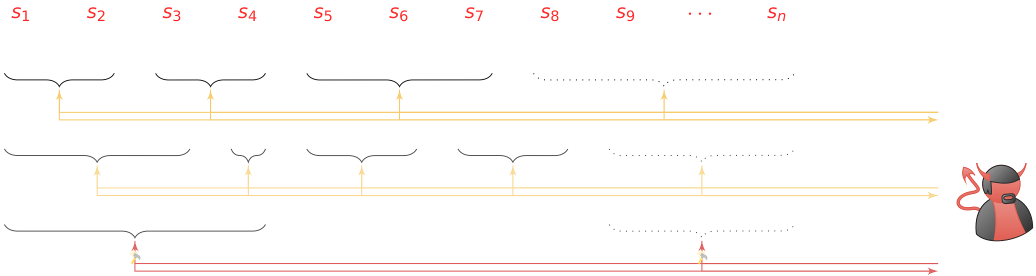
$\mathbb{P}$ [**Guess**] $= 2^{-\ell}$

Q.E.D.

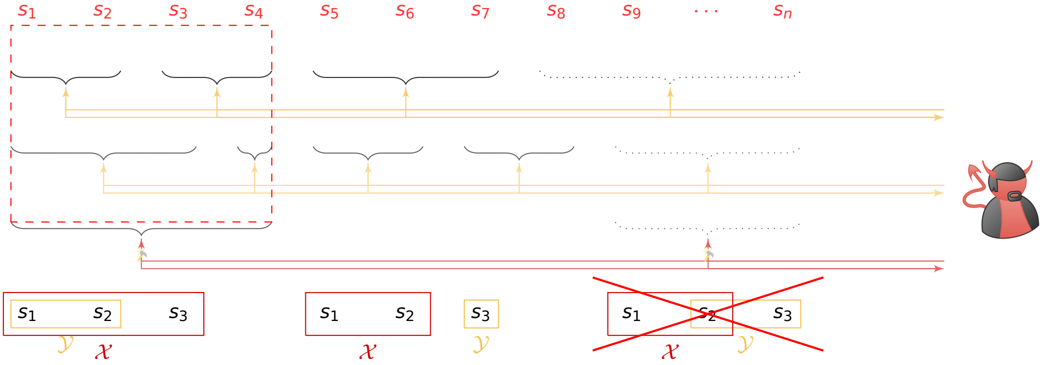
Security against **semi-**adaptive partitioning

s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 $\cdots$ s_n

Security against **semi-**adaptive partitioning



Security against **semi-**adaptive partitioning



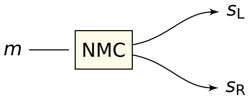
- The attacker only tampers within partitions whose subsets do not **partially** overlap with subsets belonging to leakage partitions.
- Much easier to achieve.

Our t-out-of-n semi-adaptive leakage-resilient non-malleable secret sharing

Construction inspired by [GK18]

Our t-out-of-n semi-adaptive leakage-resilient non-malleable secret sharing

Construction inspired by [GK18]

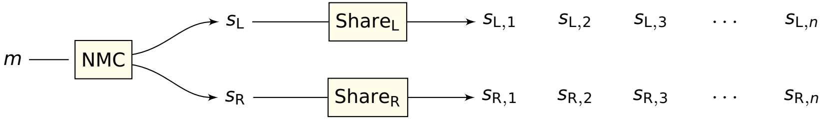


Building blocks:

- NMC: a 2-out-of-2 one-time non-malleable secret sharing scheme (i.e. a non malleable code);

Our t-out-of-n semi-adaptive leakage-resilient non-malleable secret sharing

Construction inspired by [GK18]

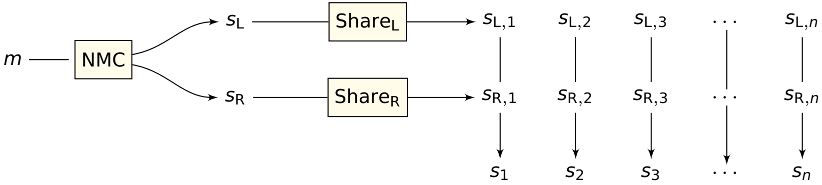


Building blocks:

- NMC: a 2-out-of-2 one-time non-malleable secret sharing scheme (i.e. a non malleable code);
- Share_L: a joint-leakage resilient t-out-of-n secret sharing scheme;
- Share_R: a joint-leakage resilient k'-out-of-n secret sharing scheme, where $k' \approx \sqrt{t}$.

Our t-out-of-n semi-adaptive leakage-resilient non-malleable secret sharing

Construction inspired by [GK18]

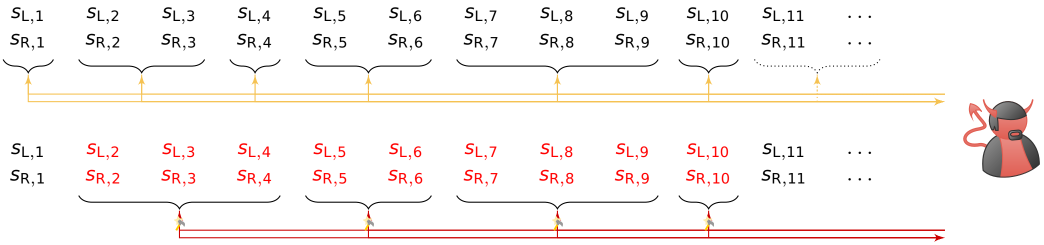


Building blocks:

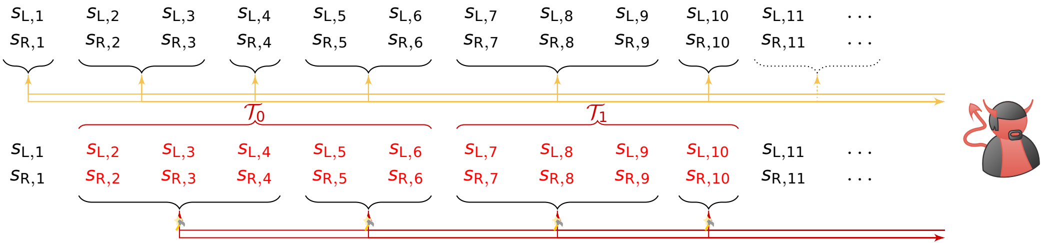
- NMC: a 2-out-of-2 one-time non-malleable secret sharing scheme (i.e. a non malleable code);
- Share_L: a joint-leakage resilient t-out-of-n secret sharing scheme;
- Share_R: a joint-leakage resilient k'-out-of-n secret sharing scheme, where $k' \approx \sqrt{t}$.

- Security proof inspired by [KMS18]
- We extend their result obtaining security against joint tampering with $k' - 1$ shares (instead of independent tampering).

Our semi-adaptive leakage-resilient non-malleable secret sharing – Proof strategy

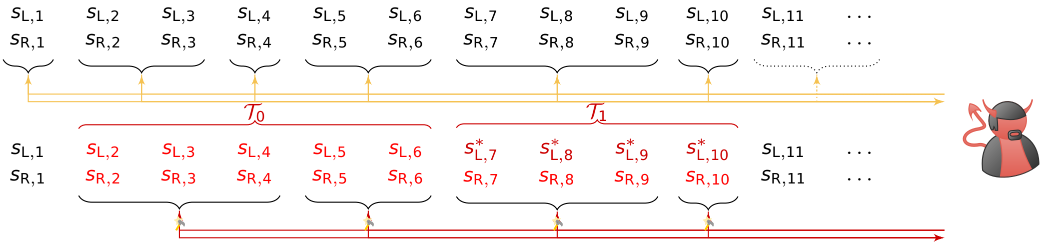


Our semi-adaptive leakage-resilient non-malleable secret sharing – Proof strategy



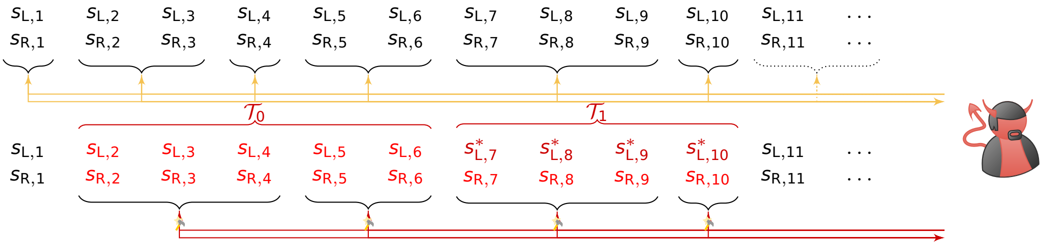
- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of $\mathsf{Share}_\mathsf{R}$.

Our semi-adaptive leakage-resilient non-malleable secret sharing – Proof strategy



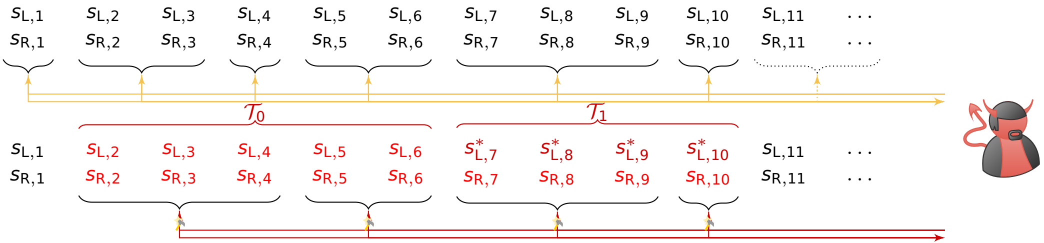
- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of Share_R.
- **Hybrid 1:** before tampering, replace the left shares within $\mathcal{T}_1$ with valid and consistent shares of the same secret.

Our semi-adaptive leakage-resilient non-malleable secret sharing – Proof strategy



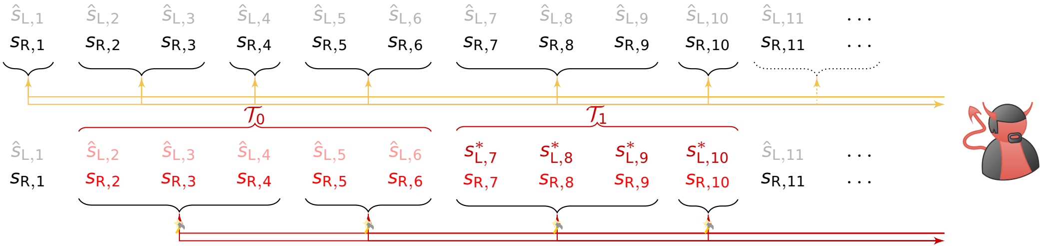
- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of $\mathsf{Share}_\mathsf{R}$.
- **Hybrid 1:** before tampering, replace the left shares within $\mathcal{T}_1$ with valid and consistent shares of the same secret.
  - Since we put the limitation of the **semi**-adaptive partitioning, the two subsets of shares $\mathcal{T}_0$ and $\mathcal{T}_1$ are unrelated each other even conditioning on the leakage.

Our semi-adaptive leakage-resilient non-malleable secret sharing – Proof strategy



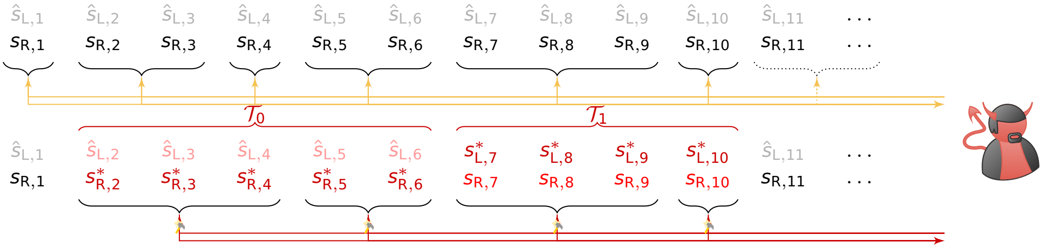
- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of Share_R.
- **Hybrid 1:** before tampering, replace the left shares within $\mathcal{T}_1$ with valid and consistent shares of the same secret.
  - Since we put the limitation of the **semi**-adaptive partitioning, the two subsets of shares $\mathcal{T}_0$ and $\mathcal{T}_1$ are unrelated each other even conditioning on the leakage.
  - This is because of each subset of each leakage partition containing only shares that are within at most one subset of the tampering partition.

Our semi-adaptive leakage-resilient non-malleable secret sharing – Proof strategy



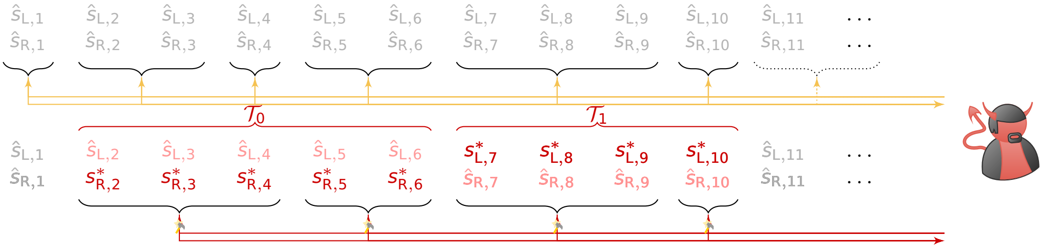
- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of Share_R.
- **Hybrid 1:** before tampering, replace the left shares within $\mathcal{T}_1$ with valid and consistent shares of the same secret.
  - Since we put the limitation of the **semi**-adaptive partitioning, the two subsets of shares $\mathcal{T}_0$ and $\mathcal{T}_1$ are unrelated each other even conditioning on the leakage.
  - This is because of each subset of each leakage partition containing only shares that are within at most one subset of the tampering partition.
- **Hybrid 2:** replace all the left shares with shares of an unrelated value $\hat{s}_L$.

Our semi-adaptive leakage-resilient non-malleable secret sharing – Proof strategy



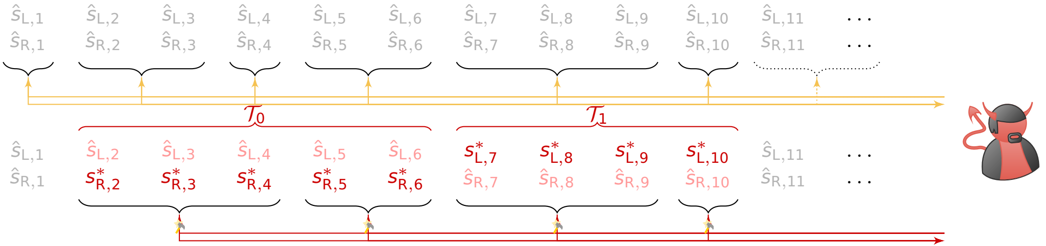
- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of Share_R.
- **Hybrid 1:** before tampering, replace the left shares within $\mathcal{T}_1$ with valid and consistent shares of the same secret.
  - Since we put the limitation of the **semi**-adaptive partitioning, the two subsets of shares $\mathcal{T}_0$ and $\mathcal{T}_1$ are unrelated each other even conditioning on the leakage.
  - This is because of each subset of each leakage partition containing only shares that are within at most one subset of the tampering partition.
- **Hybrid 2:** replace all the left shares with shares of an unrelated value $\hat{s}_L$.
- **Hybrid 3-4:** the same as in Hybrid 1-2, but on the right shares.

Our semi-adaptive leakage-resilient non-malleable secret sharing – Proof strategy



- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of Share_R.
- **Hybrid 1:** before tampering, replace the left shares within $\mathcal{T}_1$ with valid and consistent shares of the same secret.
  - Since we put the limitation of the **semi**-adaptive partitioning, the two subsets of shares $\mathcal{T}_0$ and $\mathcal{T}_1$ are unrelated each other even conditioning on the leakage.
  - This is because of each subset of each leakage partition containing only shares that are within at most one subset of the tampering partition.
- **Hybrid 2:** replace all the left shares with shares of an unrelated value $\hat{s}_L$.
- **Hybrid 3-4:** the same as in Hybrid 1-2, but on the right shares.

Our semi-adaptive leakage-resilient non-malleable secret sharing – Proof strategy



- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of Share_R.
- **Hybrid 1:** before tampering, replace the left shares within $\mathcal{T}_1$ with valid and consistent shares of the same secret.
  - Since we put the limitation of the **semi**-adaptive partitioning, the two subsets of shares $\mathcal{T}_0$ and $\mathcal{T}_1$ are unrelated each other even conditioning on the leakage.
  - This is because of each subset of each leakage partition containing only shares that are within at most one subset of the tampering partition.
- **Hybrid 2:** replace all the left shares with shares of an unrelated value $\hat{s}_L$.
- **Hybrid 3-4:** the same as in Hybrid 1-2, but on the right shares.
- Now we can safely reduce to non-malleability of the non-malleable code.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\mathsf{Share}^*(m)$**:**

- Sample random coins r for Com.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\mathsf{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\mathtt{com} \leftarrow \mathsf{Com}(m; r)$.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm Share$^*(m)$**:**

- Sample random coins r for Com.
- Compute $\texttt{com} \leftarrow \text{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\text{Share}(m||r)$.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\text{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\texttt{com} \leftarrow \text{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\text{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\texttt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\mathsf{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\mathtt{com} \leftarrow \mathsf{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\mathsf{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\mathtt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Algorithm $\mathsf{Rec}^*((s_i^*)_{i \in \mathcal{I}})$**:**

- Parse each $s_i^* = (\mathtt{com}_i, s_i)$.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\text{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\texttt{com} \leftarrow \text{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\text{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\texttt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Algorithm $\text{Rec}^*((s_i^*)_{i \in \mathcal{I}})$**:**

- Parse each $s_i^* = (\texttt{com}_i, s_i)$.
- Check if all the com are all the same.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\text{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\texttt{com} \leftarrow \text{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\text{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\texttt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Algorithm $\text{Rec}^*((s_i^*)_{i \in \mathcal{I}})$**:**

- Parse each $s_i^* = (\texttt{com}_i, s_i)$.
- Check if all the com are all the same.
- Reconstruct $m||r \leftarrow \text{Rec}((s_i)_{i \in \mathcal{I}})$.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\mathsf{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\mathtt{com} \leftarrow \mathsf{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\mathsf{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\mathtt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Algorithm $\mathsf{Rec}^*((s_i^*)_{i \in \mathcal{I}})$**:**

- Parse each $s_i^* = (\mathtt{com}_i, s_i)$.
- Check if all the com are all the same.
- Reconstruct $m||r \leftarrow \mathsf{Rec}((s_i)_{i \in \mathcal{I}})$.
- Check that (m, r) is a valid opening for com.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\text{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\texttt{com} \leftarrow \text{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\text{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\texttt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Algorithm $\text{Rec}^*((s_i^*)_{i \in \mathcal{I}})$**:**

- Parse each $s_i^* = (\texttt{com}_i, s_i)$.
- Check if all the com are all the same.
- Reconstruct $m||r \leftarrow \text{Rec}((s_i)_{i \in \mathcal{I}})$.
- Check that (m, r) is a valid opening for com.
- If everything is OK, output m; otherwise, output $\perp$.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\mathsf{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\mathtt{com} \leftarrow \mathsf{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\mathsf{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\mathtt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Algorithm $\mathsf{Rec}^*((s_i^*)_{i \in \mathcal{I}})$**:**

- Parse each $s_i^* = (\mathtt{com}_i, s_i)$.
- Check if all the com are all the same.
- Reconstruct $m||r \leftarrow \mathsf{Rec}((s_i)_{i \in \mathcal{I}})$.
- Check that (m, r) is a valid opening for com.
- If everything is OK, output m; otherwise, output $\perp$.

Key ideas (very similar to [BFV19])

- By induction over the number of queries.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\mathsf{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\mathtt{com} \leftarrow \mathsf{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\mathsf{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\mathtt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Algorithm $\mathsf{Rec}^*((s_i^*)_{i \in \mathcal{I}})$**:**

- Parse each $s_i^* = (\mathtt{com}_i, s_i)$.
- Check if all the com are all the same.
- Reconstruct $m||r \leftarrow \mathsf{Rec}((s_i)_{i \in \mathcal{I}})$.
- Check that (m, r) is a valid opening for com.
- If everything is OK, output m; otherwise, output $\perp$.

Key ideas (very similar to [BFV19])

- By induction over the number of queries.
- Simulate tampering with leakage: obtain the mauled commitment and then extract the respective secret message.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\mathsf{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\mathtt{com} \leftarrow \mathsf{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\mathsf{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\mathtt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Algorithm $\mathsf{Rec}^*((s_i^*)_{i \in \mathcal{I}})$**:**

- Parse each $s_i^* = (\mathtt{com}_i, s_i)$.
- Check if all the com are all the same.
- Reconstruct $m||r \leftarrow \mathsf{Rec}((s_i)_{i \in \mathcal{I}})$.
- Check that (m, r) is a valid opening for com.
- If everything is OK, output m; otherwise, output $\perp$.

Key ideas (very similar to [BFV19])

- By induction over the number of queries.
- Simulate tampering with leakage: obtain the mauled commitment and then extract the respective secret message.
- Check if everything is correct with the last tampering query.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\mathsf{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\mathtt{com} \leftarrow \mathsf{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\mathsf{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\mathtt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Algorithm $\mathsf{Rec}^*((s_i^*)_{i\in\mathcal{I}})$**:**

- Parse each $s_i^* = (\mathtt{com}_i, s_i)$.
- Check if all the com are all the same.
- Reconstruct $m||r \leftarrow \mathsf{Rec}((s_i)_{i\in\mathcal{I}})$.
- Check that (m, r) is a valid opening for com.
- If everything is OK, output m; otherwise, output $\perp$.

Key ideas (very similar to [BFV19])

- By induction over the number of queries.
- Simulate tampering with leakage: obtain the mauled commitment and then extract the respective secret message.
- Check if everything is correct with the last tampering query.
- Commitment scheme $\Longrightarrow$ computational setting.

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\mathsf{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\mathtt{com} \leftarrow \mathsf{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\mathsf{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\mathtt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Algorithm $\mathsf{Rec}^*((s_i^*)_{i \in \mathcal{I}})$**:**

- Parse each $s_i^* = (\mathtt{com}_i, s_i)$.
- Check if all the com are all the same.
- Reconstruct $m||r \leftarrow \mathsf{Rec}((s_i)_{i \in \mathcal{I}})$.
- Check that (m, r) is a valid opening for com.
- If everything is OK, output m; otherwise, output $\perp$.

Key ideas (very similar to [BFV19])

- By induction over the number of queries.
- Simulate tampering with leakage: obtain the mauled commitment and then extract the respective secret message.
- Check if everything is correct with the last tampering query.
- Commitment scheme $\Longrightarrow$ computational setting.
- Bounded leakage $\Longrightarrow$ p-time non-malleability (instead of continuously).

Corollary: p-time non-malleability

Known techniques [OPVV18, BFV19], building blocks: Share, Com.

Algorithm $\mathsf{Share}^*(m)$**:**

- Sample random coins r for Com.
- Compute $\mathtt{com} \leftarrow \mathsf{Com}(m; r)$.
- Share $(s_1, \ldots, s_n) \leftarrow \$\mathsf{Share}(m||r)$.
- Let, for all $i \in [n]$, $s_i^* = (\mathtt{com}, s_i)$.
- Output $(s_1^*, \ldots, s_n^*)$.

Algorithm $\mathsf{Rec}^*((s_i^*)_{i\in\mathcal{I}})$**:**

- Parse each $s_i^* = (\mathtt{com}_i, s_i)$.
- Check if all the com are all the same.
- Reconstruct $m||r \leftarrow \mathsf{Rec}((s_i)_{i\in\mathcal{I}})$.
- Check that (m, r) is a valid opening for com.
- If everything is OK, output m; otherwise, output $\perp$.

Key ideas (very similar to [BFV19])

- By induction over the number of queries.
- Simulate tampering with leakage: obtain the mauled commitment and then extract the respective secret message.
- Check if everything is correct with the last tampering query.
- Commitment scheme $\Longrightarrow$ computational setting.
- Bounded leakage $\Longrightarrow$ p-time non-malleability (instead of continuously).
- Security against joint tampering.

Our results

- We prove that a non-malleable secret sharing scheme is also leakage resilient.

Our results

- We prove that a non-malleable secret sharing scheme is also leakage resilient.
- We give a construction of a leakage-resilient non-malleable secret sharing scheme against **semi-**adaptive partitioning.

Our results

- We prove that a non-malleable secret sharing scheme is also leakage resilient.
- We give a construction of a leakage-resilient non-malleable secret sharing scheme against **semi-**adaptive partitioning.
- **Corollary:** lower bounds on the size of the shares of a non-malleable secret sharing scheme.

Our results

- We prove that a non-malleable secret sharing scheme is also leakage resilient.
- We give a construction of a leakage-resilient non-malleable secret sharing scheme against **semi-**adaptive partitioning.
- **Corollary:** lower bounds on the size of the shares of a non-malleable secret sharing scheme.
- **Corollary:** construction of a p-time non-malleable secret sharing scheme.

Our results

- We prove that a non-malleable secret sharing scheme is also leakage resilient.
- We give a construction of a leakage-resilient non-malleable secret sharing scheme against **semi-**adaptive partitioning.
- **Corollary:** lower bounds on the size of the shares of a non-malleable secret sharing scheme.
- **Corollary:** construction of a p-time non-malleable secret sharing scheme.

Open problems / Work in Progress

- *Actually, we already have some preliminary work in progress...*
- Continuous non-mallebility against joint selective/(semi-)adaptive in the plain model.

Our results

- We prove that a non-malleable secret sharing scheme is also leakage resilient.
- We give a construction of a leakage-resilient non-malleable secret sharing scheme against **semi-**adaptive partitioning.
- **Corollary:** lower bounds on the size of the shares of a non-malleable secret sharing scheme.
- **Corollary:** construction of a p-time non-malleable secret sharing scheme.

Open problems / Work in Progress

- *Actually, we already have some preliminary work in progress...*
- Continuous non-mallebility against joint selective/(semi-)adaptive in the plain model.
- Optimal rate, i.e. $\frac{\text{size of message}}{\text{size of share}}$.

Conclusion and open problems

Our results

- We prove that a non-malleable secret sharing scheme is also leakage resilient.
- We give a construction of a leakage-resilient non-malleable secret sharing scheme against **semi-**adaptive partitioning.
- **Corollary:** lower bounds on the size of the shares of a non-malleable secret sharing scheme.
- **Corollary:** construction of a p-time non-malleable secret sharing scheme.

Open problems / Work in Progress

- *Actually, we already have some preliminary work in progress...*
- Continuous non-mallebility against joint selective/(semi-)adaptive in the plain model.
- Optimal rate, i.e. $\frac{\text{size of message}}{\text{size of share}}$.

Thank You!