Leakage Resilient Non-Malleable Secret Sharing

Gianluca Brian
Sapienza University of Rome
Rome, Italy

12-13 october 2020
Part 2

State of the art

Tampering model		Leakage model	Reference	Notes
1-time	independent tampering	/	[GK18]	
		/	[SV18]	
		Bounded-leakage	[KMS18]	
1-time	joint tampering	/	[GK18]	$\mathcal{B}$ partition of $\mathcal{T}$
1-time	cover-free tampering	/	[GSZ20]	
p-time	independent tampering	/	[BS18]	NAT
		/	[ADN+20]	NAT, NACR
p-time	joint tampering	Bounded-leakage	[BFOSV20]	$\mathcal{B}$ partition of $\mathcal{T}$
		/	[BFOSV20]	Semi-adaptive partitioning
continuous	independent tampering	Noisy-leakage*	[FV19]	Non-standard leakage model, ramp
		Noisy-leakage*	[BFV19]	Non-standard leakage model
continuous	joint tampering	Bounded-leakage	[BFV19]	CRS model
/	/	Bounded-leakage	[KMZ20]	$O(t/\log(t))$-sized partitioning
/	/	Bounded-leakage	[CGGL20]	$(0.99n)$-sized partitioning, n-out-of-n

A common technique: NMC, then Share [GK18]

Building blocks

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.
- A t-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^t, \mathsf{Rec}_n^t)$ taking as input values in $\mathcal{L}$.
- A k-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^k, \mathsf{Rec}_n^k)$ taking as input values in $\mathcal{R}$, where $k = 1 + \lfloor t/2 \rfloor$.

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.
- A t-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^t, \mathsf{Rec}_n^t)$ taking as input values in $\mathcal{L}$.
- A k-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^k, \mathsf{Rec}_n^k)$ taking as input values in $\mathcal{R}$, where $k = 1 + \lfloor t/2 \rfloor$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.
- A t-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^t, \mathsf{Rec}_n^t)$ taking as input values in $\mathcal{L}$.
- A k-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^k, \mathsf{Rec}_n^k)$ taking as input values in $\mathcal{R}$, where $k = 1 + \lfloor t/2 \rfloor$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_{\mathsf{L}}, \sigma_{\mathsf{R}}) \leftarrow\!\!\$\ \mathsf{NMEnc}(\mu)$;

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.
- A t-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^t, \mathsf{Rec}_n^t)$ taking as input values in $\mathcal{L}$.
- A k-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^k, \mathsf{Rec}_n^k)$ taking as input values in $\mathcal{R}$, where $k = 1 + \lfloor t/2 \rfloor$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_{\mathsf{L}}, \sigma_{\mathsf{R}}) \leftarrow\!\!\$\ \mathsf{NMEnc}(\mu)$;
  - compute $(\sigma_{\mathsf{L},1}, \ldots, \sigma_{\mathsf{L},n}) \leftarrow\!\!\$\ \mathsf{Share}_n^t(\sigma_{\mathsf{L}})$ and $(\sigma_{\mathsf{R},1}, \ldots, \sigma_{\mathsf{R},n}) \leftarrow\!\!\$\ \mathsf{Share}_n^k(\sigma_{\mathsf{R}})$;

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.
- A t-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^t, \mathsf{Rec}_n^t)$ taking as input values in $\mathcal{L}$.
- A k-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^k, \mathsf{Rec}_n^k)$ taking as input values in $\mathcal{R}$, where $k = 1 + \lfloor t/2 \rfloor$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_{\mathsf{L}}, \sigma_{\mathsf{R}}) \leftarrow\$ \mathsf{NMEnc}(\mu)$;
  - compute $(\sigma_{\mathsf{L},1}, \ldots, \sigma_{\mathsf{L},n}) \leftarrow\$ \mathsf{Share}_n^t(\sigma_{\mathsf{L}})$ and $(\sigma_{\mathsf{R},1}, \ldots, \sigma_{\mathsf{R},n}) \leftarrow\$ \mathsf{Share}_n^k(\sigma_{\mathsf{R}})$;
  - output the shares $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$.

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.
- A t-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^t, \mathsf{Rec}_n^t)$ taking as input values in $\mathcal{L}$.
- A k-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^k, \mathsf{Rec}_n^k)$ taking as input values in $\mathcal{R}$, where $k = 1 + \lfloor t/2 \rfloor$.

- **Sharing algorithm** $\mathsf{NMShare}$**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_\mathsf{L}, \sigma_\mathsf{R}) \leftarrow\$ \mathsf{NMEnc}(\mu)$;
  - compute $(\sigma_{\mathsf{L},1}, \ldots, \sigma_{\mathsf{L},n}) \leftarrow\$ \mathsf{Share}_n^t(\sigma_\mathsf{L})$ and $(\sigma_{\mathsf{R},1}, \ldots, \sigma_{\mathsf{R},n}) \leftarrow\$ \mathsf{Share}_n^k(\sigma_\mathsf{R})$;
  - output the shares $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.
- A t-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^t, \mathsf{Rec}_n^t)$ taking as input values in $\mathcal{L}$.
- A k-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^k, \mathsf{Rec}_n^k)$ taking as input values in $\mathcal{R}$, where $k = 1 + \lfloor t/2 \rfloor$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_{\mathsf{L}}, \sigma_{\mathsf{R}}) \leftarrow\$ \mathsf{NMEnc}(\mu)$;
  - compute $(\sigma_{\mathsf{L},1}, \ldots, \sigma_{\mathsf{L},n}) \leftarrow\$ \mathsf{Share}_n^t(\sigma_{\mathsf{L}})$ and $(\sigma_{\mathsf{R},1}, \ldots, \sigma_{\mathsf{R},n}) \leftarrow\$ \mathsf{Share}_n^k(\sigma_{\mathsf{R}})$;
  - output the shares $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$;

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.
- A t-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^t, \mathsf{Rec}_n^t)$ taking as input values in $\mathcal{L}$.
- A k-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^k, \mathsf{Rec}_n^k)$ taking as input values in $\mathcal{R}$, where $k = 1 + \lfloor t/2 \rfloor$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_\mathsf{L}, \sigma_\mathsf{R}) \leftarrow\$ \mathsf{NMEnc}(\mu)$;
  - compute $(\sigma_{\mathsf{L},1}, \ldots, \sigma_{\mathsf{L},n}) \leftarrow\$ \mathsf{Share}_n^t(\sigma_\mathsf{L})$ and $(\sigma_{\mathsf{R},1}, \ldots, \sigma_{\mathsf{R},n}) \leftarrow\$ \mathsf{Share}_n^k(\sigma_\mathsf{R})$;
  - output the shares $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma_i^*)_{i\in\mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$;
  - verify if all the shares $(\sigma_{\mathsf{R},i})_{i\in\mathcal{I}}$ are consistent under k-out-of-n Shamir Secret Sharing, and output $\perp$ if not;

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.
- A t-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^t, \mathsf{Rec}_n^t)$ taking as input values in $\mathcal{L}$.
- A k-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^k, \mathsf{Rec}_n^k)$ taking as input values in $\mathcal{R}$, where $k = 1 + \lfloor t/2 \rfloor$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_\mathsf{L}, \sigma_\mathsf{R}) \leftarrow\$ \mathsf{NMEnc}(\mu)$;
  - compute $(\sigma_{\mathsf{L},1}, \ldots, \sigma_{\mathsf{L},n}) \leftarrow\$ \mathsf{Share}_n^t(\sigma_\mathsf{L})$ and $(\sigma_{\mathsf{R},1}, \ldots, \sigma_{\mathsf{R},n}) \leftarrow\$ \mathsf{Share}_n^k(\sigma_\mathsf{R})$;
  - output the shares $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$;
  - verify if all the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{I}}$ are consistent under k-out-of-n Shamir Secret Sharing, and output $\perp$ if not;
  - reconstruct $\sigma_\mathsf{L} = \mathsf{Rec}_n^t((\sigma_{\mathsf{L},i})_{i \in \mathcal{I}})$ and $\sigma_\mathsf{R} = \mathsf{Rec}_n^k((\sigma_{\mathsf{R},i})_{i \in \mathcal{I}})$;

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.
- A t-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^t, \mathsf{Rec}_n^t)$ taking as input values in $\mathcal{L}$.
- A k-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^k, \mathsf{Rec}_n^k)$ taking as input values in $\mathcal{R}$, where $k = 1 + \lfloor t/2 \rfloor$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_\mathsf{L}, \sigma_\mathsf{R}) \leftarrow\$ \mathsf{NMEnc}(\mu)$;
  - compute $(\sigma_{\mathsf{L},1}, \ldots, \sigma_{\mathsf{L},n}) \leftarrow\$ \mathsf{Share}_n^t(\sigma_\mathsf{L})$ and $(\sigma_{\mathsf{R},1}, \ldots, \sigma_{\mathsf{R},n}) \leftarrow\$ \mathsf{Share}_n^k(\sigma_\mathsf{R})$;
  - output the shares $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$;
  - verify if all the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{I}}$ are consistent under k-out-of-n Shamir Secret Sharing, and output $\perp$ if not;
  - reconstruct $\sigma_\mathsf{L} = \mathsf{Rec}_n^t((\sigma_{\mathsf{L},i})_{i \in \mathcal{I}})$ and $\sigma_\mathsf{R} = \mathsf{Rec}_n^k((\sigma_{\mathsf{R},i})_{i \in \mathcal{I}})$;
  - decode $\mu = \mathsf{NMDec}(\sigma_\mathsf{L}, \sigma_\mathsf{R})$ and output μ.

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

A common technique: NMC, then Share [GK18]

Building blocks

- A one-time ε-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$ and is also $(\log(|\mathcal{L}|) + \log(1/\varepsilon))$-leakage-resilient on the right share.
- A t-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^t, \mathsf{Rec}_n^t)$ taking as input values in $\mathcal{L}$.
- A k-out-of-n Shamir Secret Sharing scheme $(\mathsf{Share}_n^k, \mathsf{Rec}_n^k)$ taking as input values in $\mathcal{R}$, where $k = 1 + \lfloor t/2 \rfloor$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_\mathsf{L}, \sigma_\mathsf{R}) \leftarrow\$ \mathsf{NMEnc}(\mu)$;
  - compute $(\sigma_{\mathsf{L},1}, \ldots, \sigma_{\mathsf{L},n}) \leftarrow\$ \mathsf{Share}_n^t(\sigma_\mathsf{L})$ and $(\sigma_{\mathsf{R},1}, \ldots, \sigma_{\mathsf{R},n}) \leftarrow\$ \mathsf{Share}_n^k(\sigma_\mathsf{R})$;
  - output the shares $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$;
  - verify if all the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{I}}$ are consistent under k-out-of-n Shamir Secret Sharing, and output $\perp$ if not;
  - reconstruct $\sigma_\mathsf{L} = \mathsf{Rec}_n^t((\sigma_{\mathsf{L},i})_{i \in \mathcal{I}})$ and $\sigma_\mathsf{R} = \mathsf{Rec}_n^k((\sigma_{\mathsf{R},i})_{i \in \mathcal{I}})$;
  - decode $\mu = \mathsf{NMDec}(\sigma_\mathsf{L}, \sigma_\mathsf{R})$ and output μ.

The above scheme is a $(t-1)$-joint* t-out-of-n one-time 2ε-non-malleable secret sharing scheme.

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

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction $\hat{\mathsf{A}}$ samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$.

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction $\hat{\mathsf{A}}$ samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$.
- **Leakage from σ_{R}:** using σ_{R}, randomness ρ_1 and the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$, obtain the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_1}$; then,

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction $\hat{\mathrm{A}}$ samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$.
- **Leakage from σ_{R}:** using σ_{R}, randomness ρ_1 and the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$, obtain the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_1}$; then,
    - apply the tampering function f_1;

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction $\hat{\mathsf{A}}$ samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$.
- **Leakage from σ_{R}:** using σ_{R}, randomness ρ_1 and the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$, obtain the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_1}$; then,
  - apply the tampering function f_1;
  - compute a partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ of the left tampered share;

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction $\hat{\mathsf{A}}$ samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$.
- **Leakage from σ_{R}:** using σ_{R}, randomness ρ_1 and the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$, obtain the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_1}$; then,
  - apply the tampering function f_1;
  - compute a partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ of the left tampered share;
  - compute an auxiliary information α that depends on the randomness ρ and the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_2}$ obtained by interpolating the values $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_1}$ (if they are consistent);

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction Â samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$.
- **Leakage from σ_{R}:** using σ_{R}, randomness ρ_1 and the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$, obtain the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_1}$; then,
  - apply the tampering function f_1;
  - compute a partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ of the left tampered share;
  - compute an auxiliary information α that depends on the randomness ρ and the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_2}$ obtained by interpolating the values $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_1}$ (if they are consistent);
  - output $(\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}, \alpha)$.

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction $\hat{\mathsf{A}}$ samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$.
- **Leakage from σ_R:** using σ_R, randomness ρ_1 and the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$, obtain the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_1}$; then,
  - apply the tampering function f_1;
  - compute a partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ of the left tampered share;
  - compute an auxiliary information α that depends on the randomness ρ and the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_2}$ obtained by interpolating the values $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_1}$ (if they are consistent);
  - output $(\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}, \alpha)$.
- **Tampering with σ_L:** using σ_L, randomness ρ_2 and the shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$, obtain the shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_2}$; then,

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction $\hat{\mathsf{A}}$ samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$.
- **Leakage from σ_{R}:** using σ_{R}, randomness ρ_1 and the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$, obtain the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_1}$; then,
    - apply the tampering function f_1;
    - compute a partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ of the left tampered share;
    - compute an auxiliary information α that depends on the randomness ρ and the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_2}$ obtained by interpolating the values $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_1}$ (if they are consistent);
    - output $(\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}, \alpha)$.
- **Tampering with σ_{L}:** using σ_{L}, randomness ρ_2 and the shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$, obtain the shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_2}$; then,
    - apply the tampering function f_2;

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction $\hat{\mathsf{A}}$ samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$.
- **Leakage from σ_{R}:** using σ_{R}, randomness ρ_1 and the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$, obtain the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_1}$; then,
  - apply the tampering function f_1;
  - compute a partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ of the left tampered share;
  - compute an auxiliary information α that depends on the randomness ρ and the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_2}$ obtained by interpolating the values $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_1}$ (if they are consistent);
  - output $(\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}, \alpha)$.
- **Tampering with σ_{L}:** using σ_{L}, randomness ρ_2 and the shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$, obtain the shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_2}$; then,
  - apply the tampering function f_2;
  - use randomness ρ and the auxiliary information α to check that the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_2}$ are consistent with the ones computed during the leakage phase;

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction $\hat{\mathsf{A}}$ samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$.
- **Leakage from σ_R:** using σ_R, randomness ρ_1 and the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$, obtain the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_1}$; then,
  - apply the tampering function f_1;
  - compute a partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ of the left tampered share;
  - compute an auxiliary information α that depends on the randomness ρ and the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_2}$ obtained by interpolating the values $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_1}$ (if they are consistent);
  - output $(\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}, \alpha)$.
- **Tampering with σ_L:** using σ_L, randomness ρ_2 and the shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$, obtain the shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_2}$; then,
  - apply the tampering function f_2;
  - use randomness ρ and the auxiliary information α to check that the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_2}$ are consistent with the ones computed during the leakage phase;
  - if everything is consistent, use the partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ and the tampered shares $(\tilde{\sigma}_{\mathsf{L},j})_{j \in \mathcal{B}_2}$ to obtain the tampered left share $\tilde{\sigma}_\mathsf{L}$; otherwise, output $\perp$;

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction $\hat{\mathsf{A}}$ samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i\in\mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i\in\mathcal{B}_2}$.
- **Leakage from σ_R:** using σ_R, randomness ρ_1 and the shares $(\sigma_{\mathsf{R},i})_{i\in\mathcal{B}_2}$, obtain the shares $(\sigma_{\mathsf{R},i})_{i\in\mathcal{B}_1}$; then,
  - apply the tampering function f_1;
  - compute a partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ of the left tampered share;
  - compute an auxiliary information α that depends on the randomness ρ and the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j\in\mathcal{B}_2}$ obtained by interpolating the values $(\tilde{\sigma}_{\mathsf{R},j})_{j\in\mathcal{B}_1}$ (if they are consistent);
  - output $(\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}, \alpha)$.
- **Tampering with σ_L:** using σ_L, randomness ρ_2 and the shares $(\sigma_{\mathsf{L},i})_{i\in\mathcal{B}_1}$, obtain the shares $(\sigma_{\mathsf{L},i})_{i\in\mathcal{B}_2}$; then,
  - apply the tampering function f_2;
  - use randomness ρ and the auxiliary information α to check that the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j\in\mathcal{B}_2}$ are consistent with the ones computed during the leakage phase;
  - if everything is consistent, use the partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ and the tampered shares $(\tilde{\sigma}_{\mathsf{L},j})_{j\in\mathcal{B}_2}$ to obtain the tampered left share $\tilde{\sigma}_\mathsf{L}$; otherwise, output $\perp$;
  - output $\tilde{\sigma}_\mathsf{L}$.

A common technique: NMC, then Share [GK18] — Proof

- By reduction to the underlyng leakage-resilient non-malleable code. Fix any set $\mathcal{T} \subset [n]$ such that $|\mathcal{T}| = t$ and any partition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{T}$ such that $|\mathcal{B}_1| \geq k > |\mathcal{B}_2|$ (since $k = 1 + \lfloor t/2 \rfloor$). Let (f_1, f_2) be the tampering query.
- **Setup:** the reduction $\hat{\mathsf{A}}$ samples random strings ρ, ρ_1, ρ_2 and random shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$ and $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$.
- **Leakage from σ_R:** using σ_R, randomness ρ_1 and the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_2}$, obtain the shares $(\sigma_{\mathsf{R},i})_{i \in \mathcal{B}_1}$; then,
  - apply the tampering function f_1;
  - compute a partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ of the left tampered share;
  - compute an auxiliary information α that depends on the randomness ρ and the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_2}$ obtained by interpolating the values $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_1}$ (if they are consistent);
  - output $(\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}, \alpha)$.
- **Tampering with σ_L:** using σ_L, randomness ρ_2 and the shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_1}$, obtain the shares $(\sigma_{\mathsf{L},i})_{i \in \mathcal{B}_2}$; then,
  - apply the tampering function f_2;
  - use randomness ρ and the auxiliary information α to check that the tampered shares $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_2}$ are consistent with the ones computed during the leakage phase;
  - if everything is consistent, use the partial reconstruction $\tilde{\sigma}_{\mathsf{L},\mathcal{B}_1}$ and the tampered shares $(\tilde{\sigma}_{\mathsf{L},j})_{j \in \mathcal{B}_2}$ to obtain the tampered left share $\tilde{\sigma}_\mathsf{L}$; otherwise, output $\perp$;
  - output $\tilde{\sigma}_\mathsf{L}$.
- **Tampering with σ_R:** perform the same steps as in the leakage phase, but output the value $\tilde{\sigma}_\mathsf{R}$ if the shares $(\tilde{\sigma}_{\mathsf{R},j})_{j \in \mathcal{B}_1}$ are consistent and $\perp$ otherwise.

Achieving leakage resilience [BFOSV20]

Building blocks

[BFOSV20] “Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model”, *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving leakage resilience [BFOSV20]

Building blocks

- A one-time ε_2-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$.

[BFOSV20] “Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model”, *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving leakage resilience [BFOSV20]

Building blocks

- A one-time ε_2-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$.
- A t-out-of-n k-joint ℓ_L-bounded ε_L-leakage-resilient secret sharing scheme $(\mathsf{Share}_\mathsf{L}, \mathsf{Rec}_\mathsf{L})$.
- A t_R-out-of-n $(t_\mathsf{R} - 1)$-joint ℓ_R-bounded ε_R-leakage-resilient secret sharing scheme $(\mathsf{Share}_\mathsf{R}, \mathsf{Rec}_\mathsf{R})$.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving leakage resilience [BFOSV20]

Building blocks

- A one-time ε_2-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$.
- A t-out-of-n k-joint ℓ_{L}-bounded ε_{L}-leakage-resilient secret sharing scheme $(\mathsf{Share}_{\mathsf{L}}, \mathsf{Rec}_{\mathsf{L}})$.
- A t_{R}-out-of-n $(t_{\mathsf{R}} - 1)$-joint ℓ_{R}-bounded ε_{R}-leakage-resilient secret sharing scheme $(\mathsf{Share}_{\mathsf{R}}, \mathsf{Rec}_{\mathsf{R}})$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_{\mathsf{L}}, \sigma_{\mathsf{R}}) \leftarrow\!\$\ \mathsf{NMEnc}(\mu)$;
  - compute $(\sigma_{\mathsf{L},1}, \ldots, \sigma_{\mathsf{L},n}) \leftarrow\!\$\ \mathsf{Share}_{\mathsf{L}}(\sigma_{\mathsf{L}})$ and $(\sigma_{\mathsf{R},1}, \ldots, \sigma_{\mathsf{R},n}) \leftarrow\!\$\ \mathsf{Share}_{\mathsf{R}}(\sigma_{\mathsf{R}})$;
  - output the shares $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving leakage resilience [BFOSV20]

Building blocks

- A one-time ε_2-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$.
- A t-out-of-n k-joint ℓ_L-bounded ε_L-leakage-resilient secret sharing scheme $(\mathsf{Share}_\mathsf{L}, \mathsf{Rec}_\mathsf{L})$.
- A t_R-out-of-n $(t_\mathsf{R} - 1)$-joint ℓ_R-bounded ε_R-leakage-resilient secret sharing scheme $(\mathsf{Share}_\mathsf{R}, \mathsf{Rec}_\mathsf{R})$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_\mathsf{L}, \sigma_\mathsf{R}) \leftarrow\$ \mathsf{NMEnc}(\mu)$;
  - compute $(\sigma_{\mathsf{L},1}, \ldots, \sigma_{\mathsf{L},n}) \leftarrow\$ \mathsf{Share}_\mathsf{L}(\sigma_\mathsf{L})$ and $(\sigma_{\mathsf{R},1}, \ldots, \sigma_{\mathsf{R},n}) \leftarrow\$ \mathsf{Share}_\mathsf{R}(\sigma_\mathsf{R})$;
  - output the shares $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$;
  - reconstruct $\sigma_\mathsf{L} = \mathsf{Rec}_\mathsf{L}((\sigma_{\mathsf{L},i})_{i \in \mathcal{I}})$ and $\sigma_\mathsf{R} = \mathsf{Rec}_\mathsf{R}((\sigma_{\mathsf{R},i})_{i \in \mathcal{I}_{t_\mathsf{R}}})$;
  - decode $\mu = \mathsf{NMDec}(\sigma_\mathsf{L}, \sigma_\mathsf{R})$ and output μ.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving leakage resilience [BFOSV20]

Building blocks

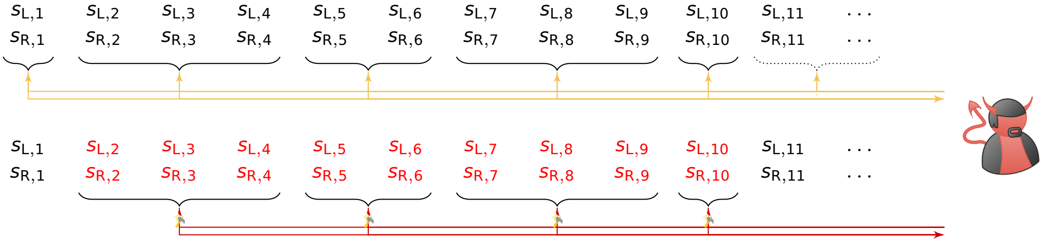
- A one-time ε_2-non-malleable code $(\mathsf{NMEnc}, \mathsf{NMDec})$ that encodes a message $\mu \in \mathcal{M}$ in two shares in $\mathcal{L} \times \mathcal{R}$.
- A t-out-of-n k-joint ℓ_L-bounded ε_L-leakage-resilient secret sharing scheme $(\mathsf{Share}_\mathsf{L}, \mathsf{Rec}_\mathsf{L})$.
- A t_R-out-of-n $(t_\mathsf{R} - 1)$-joint ℓ_R-bounded ε_R-leakage-resilient secret sharing scheme $(\mathsf{Share}_\mathsf{R}, \mathsf{Rec}_\mathsf{R})$.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_\mathsf{L}, \sigma_\mathsf{R}) \leftarrow\$ \mathsf{NMEnc}(\mu)$;
  - compute $(\sigma_{\mathsf{L},1}, \ldots, \sigma_{\mathsf{L},n}) \leftarrow\$ \mathsf{Share}_\mathsf{L}(\sigma_\mathsf{L})$ and $(\sigma_{\mathsf{R},1}, \ldots, \sigma_{\mathsf{R},n}) \leftarrow\$ \mathsf{Share}_\mathsf{R}(\sigma_\mathsf{R})$;
  - output the shares $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$;
  - reconstruct $\sigma_\mathsf{L} = \mathsf{Rec}_\mathsf{L}((\sigma_{\mathsf{L},i})_{i \in \mathcal{I}})$ and $\sigma_\mathsf{R} = \mathsf{Rec}_\mathsf{R}((\sigma_{\mathsf{R},i})_{i \in \mathcal{I}_{t_\mathsf{R}}})$;
  - decode $\mu = \mathsf{NMDec}(\sigma_\mathsf{L}, \sigma_\mathsf{R})$ and output μ.

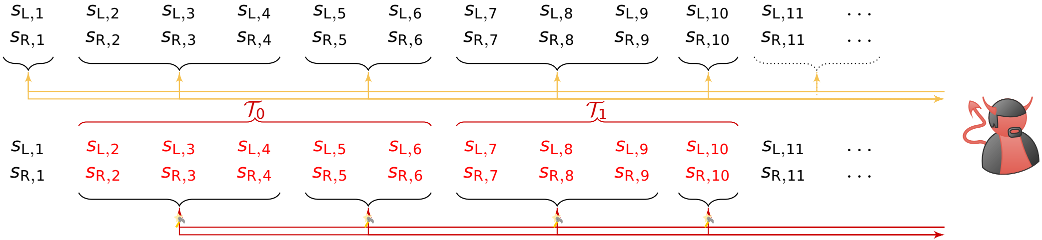
The above scheme is a $(t_\mathsf{R} - 1)$-joint* ℓ-bounded leakage resilient one-time non-malleable secret sharing scheme with security $2(\varepsilon_\mathsf{L} + \varepsilon_\mathsf{R}) + \varepsilon_2$ so long as $t_\mathsf{R} = \sqrt{k}$, $\ell_\mathsf{L} = \ell + 1$ and $\ell_\mathsf{R} = \ell + n \cdot \log|\mathcal{S}_{\mathsf{L},i}|$ for all $i \in [n]$.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving leakage resilience [BFOSV20] — Proof strategy

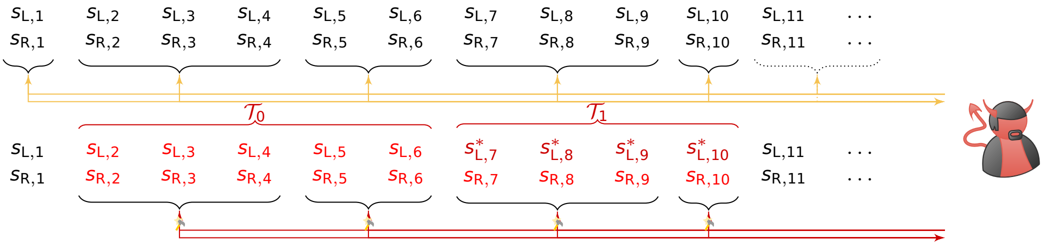


Achieving leakage resilience [BFOSV20] — Proof strategy



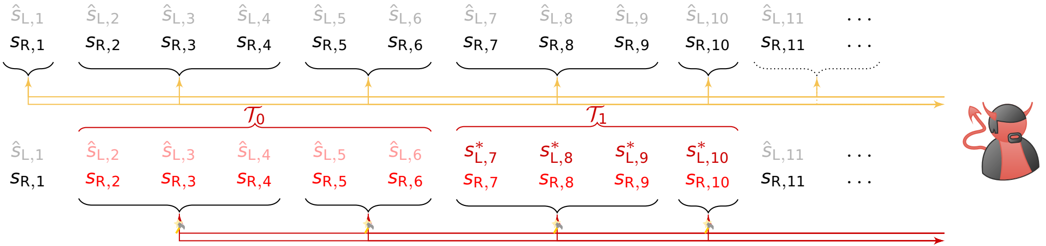
- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of $\mathsf{Share}_{\mathsf{R}}$.

Achieving leakage resilience [BFOSV20] — Proof strategy



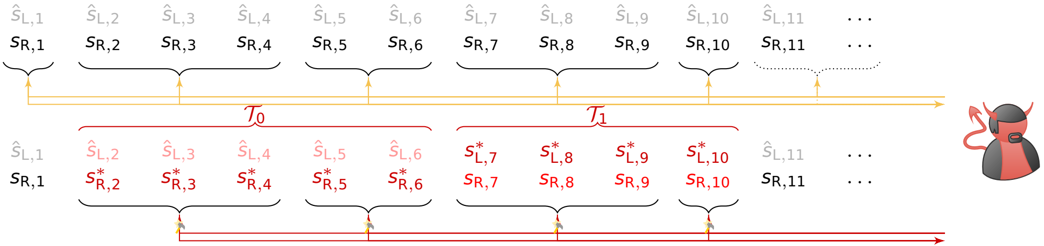
- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of $\mathsf{Share}_\mathsf{R}$.
- **Hybrid 1:** before tampering, replace the left shares within $\mathcal{T}_1$ with valid and consistent shares of the same secret.

Achieving leakage resilience [BFOSV20] — Proof strategy



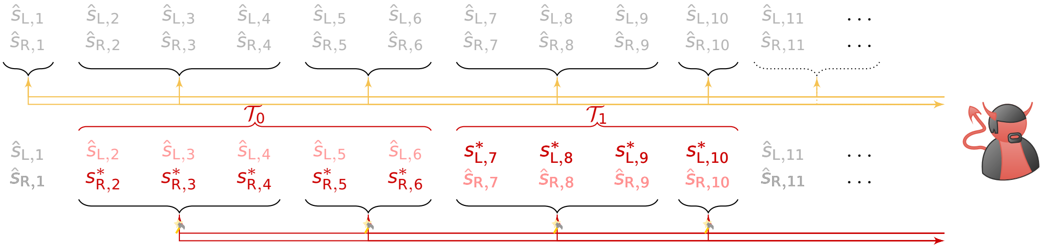
- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of $\mathsf{Share}_{\mathsf{R}}$.
- **Hybrid 1:** before tampering, replace the left shares within $\mathcal{T}_1$ with valid and consistent shares of the same secret.
- **Hybrid 2:** replace all the left shares with shares of an unrelated value $\hat{s}_{\mathsf{L}}$.

Achieving leakage resilience [BFOSV20] — Proof strategy



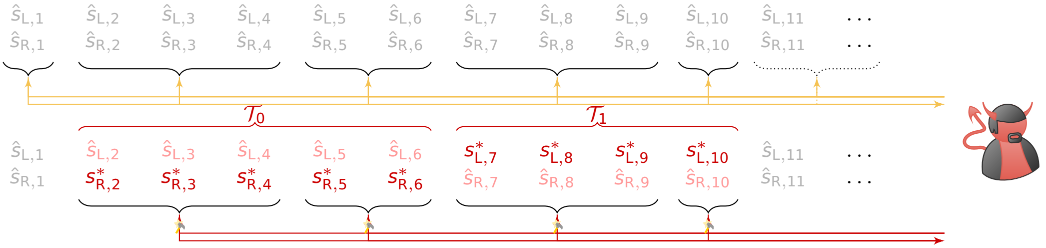
- Split the tampering set into two subsets $\mathcal{T}_0$ and $\mathcal{T}_1$ such that $|\mathcal{T}_0| \geq$ threshold of Share_R.
- **Hybrid 1:** before tampering, replace the left shares within $\mathcal{T}_1$ with valid and consistent shares of the same secret.
- **Hybrid 2:** replace all the left 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.

Achieving leakage resilience [BFOSV20] — 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.
- **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.

Achieving leakage resilience [BFOSV20] — 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.
- **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.

On adaptive partitioning...

- The previous construction is secure in a model that is stronger that selective partitioning, called *semi-adaptive partitioning*.

On adaptive partitioning...

- The previous construction is secure in a model that is stronger that selective partitioning, called *semi-adaptive partitioning*.
  - Intuitively, this model combines the adaptive partitioning for the leakage queries with the selective partitioning for the tampering query.

On adaptive partitioning...

- The previous construction is secure in a model that is stronger that selective partitioning, called *semi-adaptive partitioning*.
  - Intuitively, this model combines the adaptive partitioning for the leakage queries with the selective partitioning for the tampering query.
  - In particular, the admissible adversary is defined so that, at the end of the experiment, the leakage performed inside the reconstruction set $\mathcal{T}$ of the tampering query is leakage under selective partitioning, while the leakage performed outside $\mathcal{T}$ is leakage under adaptive partitioning.

On adaptive partitioning...

- The previous construction is secure in a model that is stronger that selective partitioning, called *semi-adaptive partitioning*.
  - Intuitively, this model combines the adaptive partitioning for the leakage queries with the selective partitioning for the tampering query.
  - In particular, the admissible adversary is defined so that, at the end of the experiment, the leakage performed inside the reconstruction set $\mathcal{T}$ of the tampering query is leakage under selective partitioning, while the leakage performed outside $\mathcal{T}$ is leakage under adaptive partitioning.
- Actually, non-malleability against adaptive partitioning is very hard to achieve. Even constructing a 3-out-of-3 secret sharing scheme that is non-malleable against adversaries who perform joint leakage from each of the three subsets $\{1,2\},\{1,3\},\{2,3\}$ and then independent tampering appears to be a challenging task [KMS18].

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

On adaptive partitioning...

- The previous construction is secure in a model that is stronger that selective partitioning, called *semi-adaptive partitioning*.
  - Intuitively, this model combines the adaptive partitioning for the leakage queries with the selective partitioning for the tampering query.
  - In particular, the admissible adversary is defined so that, at the end of the experiment, the leakage performed inside the reconstruction set $\mathcal{T}$ of the tampering query is leakage under selective partitioning, while the leakage performed outside $\mathcal{T}$ is leakage under adaptive partitioning.
- Actually, non-malleability against adaptive partitioning is very hard to achieve. Even constructing a 3-out-of-3 secret sharing scheme that is non-malleable against adversaries who perform joint leakage from each of the three subsets $\{1, 2\}, \{1, 3\}, \{2, 3\}$ and then independent tampering appears to be a challenging task [KMS18].
- This is because joint leakage leads to loss of independence among the shares, therefore the subsequent tampering queries are not independent anymore.

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

On adaptive partitioning...

- The previous construction is secure in a model that is stronger that selective partitioning, called *semi-adaptive partitioning*.
  - Intuitively, this model combines the adaptive partitioning for the leakage queries with the selective partitioning for the tampering query.
  - In particular, the admissible adversary is defined so that, at the end of the experiment, the leakage performed inside the reconstruction set $\mathcal{T}$ of the tampering query is leakage under selective partitioning, while the leakage performed outside $\mathcal{T}$ is leakage under adaptive partitioning.
- Actually, non-malleability against adaptive partitioning is very hard to achieve. Even constructing a 3-out-of-3 secret sharing scheme that is non-malleable against adversaries who perform joint leakage from each of the three subsets $\{1,2\},\{1,3\},\{2,3\}$ and then independent tampering appears to be a challenging task [KMS18].
- This is because joint leakage leads to loss of independence among the shares, therefore the subsequent tampering queries are not independent anymore.
- **Cover-free tampering [GSZ20]:** let $\mathcal{T}_1,\ldots,\mathcal{T}_n \subseteq [n]$. $(\mathcal{T}_1,\ldots,\mathcal{T}_n)$ is a *k-cover-free* family of subsets if, for all $i \in [n]$, the union of all $\mathcal{T}_j \ni i$ has at most $k-1$ elements.

[KMS18] "Leakage Resilient Secret Sharing", *Ashutosh Kumar, Raghu Meka, Amit Sahai*, IACR Cryptology ePrint Archive, Vol.2018/1138
[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

[GSZ20] “Multi-Source Non-Malleable Extractors and Applications”, *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.

[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.
- Two t-out-of-n Shamir Secret Sharing schemes $(\mathsf{Share}, \mathsf{Rec})$ with different input sizes.

[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.
- Two t-out-of-n Shamir Secret Sharing schemes $(\mathsf{Share}, \mathsf{Rec})$ with different input sizes.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,

[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.
- Two t-out-of-n Shamir Secret Sharing schemes $(\mathsf{Share}, \mathsf{Rec})$ with different input sizes.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_1, \ldots, \sigma_n) \overset{\$}{\longleftarrow} \mathsf{Share}(\mu)$;

[GSZ20] “Multi-Source Non-Malleable Extractors and Applications”, *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.
- Two t-out-of-n Shamir Secret Sharing schemes $(\mathsf{Share}, \mathsf{Rec})$ with different input sizes.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu)$;
  - for all $i \in [n]$, sample a random string ρ_i of $2|\mu|$ bits, and compute $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\$ \mathsf{2SLRNMExt}^{-1}(\sigma_i||\rho_i)$;

[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.
- Two t-out-of-n Shamir Secret Sharing schemes $(\mathsf{Share}, \mathsf{Rec})$ with different input sizes.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\!\$\ \mathsf{Share}(\mu)$;
  - for all $i \in [n]$, sample a random string ρ_i of $2|\mu|$ bits, and compute $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\!\!\$\ \mathsf{2SLRNMExt}^{-1}(\sigma_i || \rho_i)$;
  - for all $i \in [n]$, compute $(\sigma_{\mathsf{R},i}^{(1)}, \ldots, \sigma_{\mathsf{R},i}^{(n)}) \leftarrow\!\!\$\ \mathsf{Share}(\sigma_{\mathsf{R},i})$;

[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.
- Two t-out-of-n Shamir Secret Sharing schemes $(\mathsf{Share}, \mathsf{Rec})$ with different input sizes.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\$\ \mathsf{Share}(\mu)$;
  - for all $i \in [n]$, sample a random string ρ_i of $2|\mu|$ bits, and compute $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\!\$\ \mathsf{2SLRNMExt}^{-1}(\sigma_i || \rho_i)$;
  - for all $i \in [n]$, compute $(\sigma_{\mathsf{R},i}^{(1)}, \ldots, \sigma_{\mathsf{R},i}^{(n)}) \leftarrow\!\$\ \mathsf{Share}(\sigma_{\mathsf{R},i})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},1}^{(i)}, \ldots, \sigma_{\mathsf{R},n}^{(i)})$.

[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.
- Two t-out-of-n Shamir Secret Sharing schemes $(\mathsf{Share}, \mathsf{Rec})$ with different input sizes.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_1, \ldots, \sigma_n) \xleftarrow{\$} \mathsf{Share}(\mu)$;
  - for all $i \in [n]$, sample a random string ρ_i of $2|\mu|$ bits, and compute $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \xleftarrow{\$} \mathsf{2SLRNMExt}^{-1}(\sigma_i || \rho_i)$;
  - for all $i \in [n]$, compute $(\sigma^{(1)}_{\mathsf{R},i}, \ldots, \sigma^{(n)}_{\mathsf{R},i}) \xleftarrow{\$} \mathsf{Share}(\sigma_{\mathsf{R},i})$;
  - output $(\sigma^*_1, \ldots, \sigma^*_n)$, where, for each $i \in [n]$, $\sigma^*_i = (\sigma_{\mathsf{L},i}, \sigma^{(i)}_{\mathsf{R},1}, \ldots, \sigma^{(i)}_{\mathsf{R},n})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma^*_i)_{i \in \mathcal{I}}$,

[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.
- Two t-out-of-n Shamir Secret Sharing schemes $(\mathsf{Share}, \mathsf{Rec})$ with different input sizes.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu)$;
  - for all $i \in [n]$, sample a random string ρ_i of $2|\mu|$ bits, and compute $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\$ \mathsf{2SLRNMExt}^{-1}(\sigma_i||\rho_i)$;
  - for all $i \in [n]$, compute $(\sigma^{(1)}_{\mathsf{R},i}, \ldots, \sigma^{(n)}_{\mathsf{R},i}) \leftarrow\$ \mathsf{Share}(\sigma_{\mathsf{R},i})$;
  - output $(\sigma^*_1, \ldots, \sigma^*_n)$, where, for each $i \in [n]$, $\sigma^*_i = (\sigma_{\mathsf{L},i}, \sigma^{(i)}_{\mathsf{R},1}, \ldots, \sigma^{(i)}_{\mathsf{R},n})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma^*_i)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma^*_i = (\sigma_{\mathsf{L},i}, \sigma^{(i)}_{\mathsf{R},1}, \ldots, \sigma^{(i)}_{\mathsf{R},n})$;

[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.
- Two t-out-of-n Shamir Secret Sharing schemes $(\mathsf{Share}, \mathsf{Rec})$ with different input sizes.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\!\$\ \mathsf{Share}(\mu)$;
  - for all $i \in [n]$, sample a random string ρ_i of $2|\mu|$ bits, and compute $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\!\!\$\ \mathsf{2SLRNMExt}^{-1}(\sigma_i || \rho_i)$;
  - for all $i \in [n]$, compute $(\sigma^{(1)}_{\mathsf{R},i}, \ldots, \sigma^{(n)}_{\mathsf{R},i}) \leftarrow\!\!\$\ \mathsf{Share}(\sigma_{\mathsf{R},i})$;
  - output $(\sigma^*_1, \ldots, \sigma^*_n)$, where, for each $i \in [n]$, $\sigma^*_i = (\sigma_{\mathsf{L},i}, \sigma^{(i)}_{\mathsf{R},1}, \ldots, \sigma^{(i)}_{\mathsf{R},n})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma^*_i)_{i\in\mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma^*_i = (\sigma_{\mathsf{L},i}, \sigma^{(i)}_{\mathsf{R},1}, \ldots, \sigma^{(i)}_{\mathsf{R},n})$;
  - for all $i \in \mathcal{I}$, reconstruct $\sigma_{\mathsf{R},i} = \mathsf{Rec}((\sigma^{(j)}_{\mathsf{R},i})_{j\in\mathcal{I}})$;

[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.
- Two t-out-of-n Shamir Secret Sharing schemes $(\mathsf{Share}, \mathsf{Rec})$ with different input sizes.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu)$;
  - for all $i \in [n]$, sample a random string ρ_i of $2|\mu|$ bits, and compute $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\$ \mathsf{2SLRNMExt}^{-1}(\sigma_i||\rho_i)$;
  - for all $i \in [n]$, compute $(\sigma_{\mathsf{R},i}^{(1)}, \ldots, \sigma_{\mathsf{R},i}^{(n)}) \leftarrow\$ \mathsf{Share}(\sigma_{\mathsf{R},i})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},1}^{(i)}, \ldots, \sigma_{\mathsf{R},n}^{(i)})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},1}^{(i)}, \ldots, \sigma_{\mathsf{R},n}^{(i)})$;
  - for all $i \in \mathcal{I}$, reconstruct $\sigma_{\mathsf{R},i} = \mathsf{Rec}((\sigma_{\mathsf{R},i}^{(j)})_{j \in \mathcal{I}})$;
  - for all $i \in \mathcal{I}$, reconstruct $\sigma_i||\rho_i = \mathsf{2SLRNMExt}(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$;

[GSZ20] “Multi-Source Non-Malleable Extractors and Applications”, *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Example: an actual scheme achieving non-malleability against cover-free tampering

Building blocks

- A strong leakage-resilient t-times 2-source non-malleable extractor 2SLRNMExt.
- Two t-out-of-n Shamir Secret Sharing schemes $(\mathsf{Share}, \mathsf{Rec})$ with different input sizes.

- **Sharing algorithm** NMShare**:** upon input a message $\mu \in \mathcal{M}$,
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu)$;
  - for all $i \in [n]$, sample a random string ρ_i of $2|\mu|$ bits, and compute $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\$ \mathsf{2SLRNMExt}^{-1}(\sigma_i||\rho_i)$;
  - for all $i \in [n]$, compute $(\sigma_{\mathsf{R},i}^{(1)}, \ldots, \sigma_{\mathsf{R},i}^{(n)}) \leftarrow\$ \mathsf{Share}(\sigma_{\mathsf{R},i})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},1}^{(i)}, \ldots, \sigma_{\mathsf{R},n}^{(i)})$.
- **Reconstruction algorithm** NMRec**:** upon input a set of t shares $(\sigma_i^*)_{i\in\mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},1}^{(i)}, \ldots, \sigma_{\mathsf{R},n}^{(i)})$;
  - for all $i \in \mathcal{I}$, reconstruct $\sigma_{\mathsf{R},i} = \mathsf{Rec}((\sigma_{\mathsf{R},i}^{(j)})_{j\in\mathcal{I}})$;
  - for all $i \in \mathcal{I}$, reconstruct $\sigma_i||\rho_i = \mathsf{2SLRNMExt}(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$;
  - reconstruct $\mu = \mathsf{Rec}((\sigma_i)_{i\in\mathcal{I}})$ and output μ.

[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

Achieving multiple tampering queries

Building blocks

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme $(\mathsf{Commit}, \mathsf{Open})$.

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme $(\mathsf{Commit}, \mathsf{Open})$.
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme $(\mathsf{Commit}, \mathsf{Open})$.
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

- **Sharing algorithm** Share^***:** upon input a message $\mu \in \mathcal{M}$,

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme $(\mathsf{Commit}, \mathsf{Open})$.
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

- **Sharing algorithm** Share^***:** upon input a message $\mu \in \mathcal{M}$,
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme $(\mathsf{Commit}, \mathsf{Open})$.
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

- **Sharing algorithm** Share^***:** upon input a message $\mu \in \mathcal{M}$,
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\!\$\ \mathsf{Share}(\mu || \rho)$;

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme (Commit, Open).
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

- **Sharing algorithm** Share***:** upon input a message $\mu \in \mathcal{M}$,
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme $(\mathsf{Commit}, \mathsf{Open})$.
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

- **Sharing algorithm** Share^***:** upon input a message $\mu \in \mathcal{M}$,
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\!\$\ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Reconstruction algorithm** Rec^***:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme $(\mathsf{Commit}, \mathsf{Open})$.
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

- **Sharing algorithm** Share^***:** upon input a message $\mu \in \mathcal{M}$,
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\!\$\ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Reconstruction algorithm** Rec^***:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\gamma_i, \sigma_i)$;

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme $(\mathsf{Commit}, \mathsf{Open})$.
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

- **Sharing algorithm** Share^***:** upon input a message $\mu \in \mathcal{M}$,
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Reconstruction algorithm** Rec^***:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\gamma_i, \sigma_i)$;
  - if all the commitments are the same, let $\gamma = \gamma_i$, otherwise output $\perp$;

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme (Commit, Open).
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

- **Sharing algorithm** Share^***:** upon input a message $\mu \in \mathcal{M}$,
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\$\ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Reconstruction algorithm** Rec^***:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\gamma_i, \sigma_i)$;
  - if all the commitments are the same, let $\gamma = \gamma_i$, otherwise output $\perp$;
  - reconstruct $\mu || \rho = \mathsf{Rec}((\sigma_i)_{i \in \mathcal{I}})$;

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme $(\mathsf{Commit}, \mathsf{Open})$.
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

- **Sharing algorithm** Share^***:** upon input a message $\mu \in \mathcal{M}$,
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu||\rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Reconstruction algorithm** Rec^***:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\gamma_i, \sigma_i)$;
  - if all the commitments are the same, let $\gamma = \gamma_i$, otherwise output $\perp$;
  - reconstruct $\mu||\rho = \mathsf{Rec}((\sigma_i)_{i \in \mathcal{I}})$;
  - if $\gamma = \mathsf{Commit}(\mu; \rho)$, output μ, otherwise output $\perp$.

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme $(\mathsf{Commit}, \mathsf{Open})$.
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

- **Sharing algorithm** Share^***:** upon input a message $\mu \in \mathcal{M}$,
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Reconstruction algorithm** Rec^***:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\gamma_i, \sigma_i)$;
  - if all the commitments are the same, let $\gamma = \gamma_i$, otherwise output $\perp$;
  - reconstruct $\mu || \rho = \mathsf{Rec}((\sigma_i)_{i \in \mathcal{I}})$;
  - if $\gamma = \mathsf{Commit}(\mu; \rho)$, output μ, otherwise output $\perp$.

- If Π is ℓ-bounded leakage-resilient against selective/semi-adaptive partitioning, then the above scheme is p-time non-malleable against selective/semi-adaptive partitioning as long as $\ell = p \cdot (|\gamma| + n) + 1$.

Achieving multiple tampering queries

Building blocks

- A perfectly binding/computationally hiding non-interactive commitment scheme (Commit, Open).
- A t-out-of-n k-joint leakage-resilient one-time non-malleable secret sharing scheme $\Pi = (\mathsf{Share}, \mathsf{Rec})$ with information-theoretic security.

- **Sharing algorithm** Share^***:** upon input a message $\mu \in \mathcal{M}$,
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\$\ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Reconstruction algorithm** Rec^***:** upon input a set of t shares $(\sigma_i^*)_{i \in \mathcal{I}}$,
  - parse, for all $i \in \mathcal{I}$, $\sigma_i^* = (\gamma_i, \sigma_i)$;
  - if all the commitments are the same, let $\gamma = \gamma_i$, otherwise output $\perp$;
  - reconstruct $\mu || \rho = \mathsf{Rec}((\sigma_i)_{i \in \mathcal{I}})$;
  - if $\gamma = \mathsf{Commit}(\mu; \rho)$, output μ, otherwise output $\perp$.

- If Π is ℓ-bounded leakage-resilient against selective/semi-adaptive partitioning, then the above scheme is p-time non-malleable against selective/semi-adaptive partitioning as long as $\ell = p \cdot (|\gamma| + n) + 1$.
- If Π is ℓ-noisy* leakage-resilient against independent leakage and tampering, then the above scheme is ℓ'-noisy* leakage-resilient continuously non-malleable as long as $\ell = \ell' + |\gamma| + 1 + O(\log(\lambda))$.

Achieving multiple tampering queries [BFOSV20] — Proof

Hybrid argument

- **Original game:**
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\!\$\ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.

[BFOSV20] “Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model”, *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Hybrid argument

- **Original game:**
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Hybrid game:**
  - sample a random message $\hat{\mu}$ and random coins $\rho, \hat{\rho}$ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\hat{\mu} || \hat{\rho})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Hybrid argument

- **Original game:**
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Hybrid game:**
  - sample a random message $\hat{\mu}$ and random coins $\rho, \hat{\rho}$ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\hat{\mu} || \hat{\rho})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- The proof that the above games are *statistically* close proceeds by induction over the number p^* of tampering queries performed by the adversary A.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Hybrid argument

- **Original game:**
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Hybrid game:**
  - sample a random message $\hat{\mu}$ and random coins $\rho, \hat{\rho}$ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\hat{\mu} || \hat{\rho})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- The proof that the above games are *statistically* close proceeds by induction over the number p^* of tampering queries performed by the adversary A.
- **Basis of the induction:** by reduction to statistical leakage-resilience one-time non-malleability.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Hybrid argument

- **Original game:**
  - sample random coins ρ and compute $\gamma = \text{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \text{Share}(\mu||\rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Hybrid game:**
  - sample a random message $\hat{\mu}$ and random coins $\rho, \hat{\rho}$ and compute $\gamma = \text{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \text{Share}(\hat{\mu}||\hat{\rho})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- The proof that the above games are *statistically* close proceeds by induction over the number p^* of tampering queries performed by the adversary A.
- **Basis of the induction:** by reduction to statistical leakage-resilience one-time non-malleability.
  - Upon receiving the tampering query $(\mathcal{T}, f)$, the reduction uses a leakage query in order to obtain the result $\tilde{\gamma}_{i^*}$ of the tampering on one commitment γ_{i^*} such that $i^* \in \mathcal{T}$.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Hybrid argument

- **Original game:**
  - sample random coins ρ and compute $\gamma = \text{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \text{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Hybrid game:**
  - sample a random message $\hat{\mu}$ and random coins $\rho, \hat{\rho}$ and compute $\gamma = \text{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \text{Share}(\hat{\mu} || \hat{\rho})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- The proof that the above games are *statistically* close proceeds by induction over the number p^* of tampering queries performed by the adversary A.
- **Basis of the induction:** by reduction to statistical leakage-resilience one-time non-malleability.
  - Upon receiving the tampering query $(\mathcal{T}, f)$, the reduction uses a leakage query in order to obtain the result $\tilde{\gamma}_{i^*}$ of the tampering on one commitment γ_{i^*} such that $i^* \in \mathcal{T}$.
  - Using another leakage query, the reduction obtains a bit for each share in $\mathcal{T}$ telling if the corresponding commitment equals $\tilde{\gamma}_{i^*}$ or not; in the latter case, return $\perp$.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Hybrid argument

- **Original game:**
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu || \rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Hybrid game:**
  - sample a random message $\hat{\mu}$ and random coins $\rho, \hat{\rho}$ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\hat{\mu} || \hat{\rho})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- The proof that the above games are *statistically* close proceeds by induction over the number p^* of tampering queries performed by the adversary A.
- **Basis of the induction:** by reduction to statistical leakage-resilience one-time non-malleability.
  - Upon receiving the tampering query $(\mathcal{T}, f)$, the reduction uses a leakage query in order to obtain the result $\tilde{\gamma}_{i^*}$ of the tampering on one commitment γ_{i^*} such that $i^* \in \mathcal{T}$.
  - Using another leakage query, the reduction obtains a bit for each share in $\mathcal{T}$ telling if the corresponding commitment equals $\tilde{\gamma}_{i^*}$ or not; in the latter case, return $\perp$.
  - Then, the reduction forwards the tampering query to the oracle;

[BFOSV20] “Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model”, *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Hybrid argument

- **Original game:**
  - sample random coins ρ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\mu||\rho)$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- **Hybrid game:**
  - sample a random message $\hat{\mu}$ and random coins $\rho, \hat{\rho}$ and compute $\gamma = \mathsf{Commit}(\mu; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\$ \mathsf{Share}(\hat{\mu}||\hat{\rho})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- The proof that the above games are *statistically* close proceeds by induction over the number p^* of tampering queries performed by the adversary A.
- **Basis of the induction:** by reduction to statistical leakage-resilience one-time non-malleability.
  - Upon receiving the tampering query $(\mathcal{T}, f)$, the reduction uses a leakage query in order to obtain the result $\tilde{\gamma}_{i*}$ of the tampering on one commitment γ_{i*} such that $i^* \in \mathcal{T}$.
  - Using another leakage query, the reduction obtains a bit for each share in $\mathcal{T}$ telling if the corresponding commitment equals $\tilde{\gamma}_{i*}$ or not; in the latter case, return $\perp$.
  - Then, the reduction forwards the tampering query to the oracle;
  - Finally, the reduction checks that $\tilde{\gamma}_{i*}$ is a valid commitment for the outcome of the tampering query and returns either the result of the tampering or $\perp$.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.

[BFOSV20] “Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model”, *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain the result $\tilde{\gamma}_{i*}^{(q)}$ of the tampering on one commitment γ_{i*} such that $i^* \in \mathcal{T}$;

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain the result $\tilde{\gamma}_{i*}^{(q)}$ of the tampering on one commitment γ_{i*} such that $i^* \in \mathcal{T}$;
  - use another leakage query to check if all the tampered commitments correspond, and return $\perp$ if not;

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain the result $\tilde{\gamma}_{i*}^{(q)}$ of the tampering on one commitment γ_{i*} such that $i^* \in \mathcal{T}$;
  - use another leakage query to check if all the tampered commitments correspond, and return $\perp$ if not;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain the result $\tilde{\gamma}_{i*}^{(q)}$ of the tampering on one commitment γ_{i*} such that $i^* \in \mathcal{T}$;
  - use another leakage query to check if all the tampered commitments correspond, and return $\perp$ if not;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is "good", return $\mu^{(q)}$ to the adversary.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain the result $\tilde{\gamma}_{i*}^{(q)}$ of the tampering on one commitment γ_{i*} such that $i^* \in \mathcal{T}$;
  - use another leakage query to check if all the tampered commitments correspond, and return $\perp$ if not;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is "good", return $\mu^{(q)}$ to the adversary.
- Upon input the last tampering query $(\mathcal{T}^{(p+1)}, f^{(p+1)})$:

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain the result $\tilde{\gamma}_{i*}^{(q)}$ of the tampering on one commitment γ_{i*} such that $i^* \in \mathcal{T}$;
  - use another leakage query to check if all the tampered commitments correspond, and return $\perp$ if not;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is "good", return $\mu^{(q)}$ to the adversary.
- Upon input the last tampering query $(\mathcal{T}^{(p+1)}, f^{(p+1)})$:
  - construct a special leakage query checking that the simulation did not cause any inconsistency so far and outputs a bit b_{ok};

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain the result $\tilde{\gamma}_{i^*}^{(q)}$ of the tampering on one commitment γ_{i^*} such that $i^* \in \mathcal{T}$;
  - use another leakage query to check if all the tampered commitments correspond, and return $\perp$ if not;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i^*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is "good", return $\mu^{(q)}$ to the adversary.
- Upon input the last tampering query $(\mathcal{T}^{(p+1)}, f^{(p+1)})$:
  - construct a special leakage query checking that the simulation did not cause any inconsistency so far and outputs a bit b_{ok};
  - obtain the tampered commitment $\tilde{\gamma}_{i^*}^{(p+1)}$ as in the previous queries;

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain the result $\tilde{\gamma}_{i*}^{(q)}$ of the tampering on one commitment γ_{i*} such that $i^* \in \mathcal{T}$;
  - use another leakage query to check if all the tampered commitments correspond, and return $\perp$ if not;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is "good", return $\mu^{(q)}$ to the adversary.
- Upon input the last tampering query $(\mathcal{T}^{(p+1)}, f^{(p+1)})$:
  - construct a special leakage query checking that the simulation did not cause any inconsistency so far and outputs a bit b_{ok};
  - obtain the tampered commitment $\tilde{\gamma}_{i*}^{(p+1)}$ as in the previous queries;
  - forward the tampering query to the oracle and check that $\tilde{\gamma}_{i*}^{(p+1)}$ is a valid commitment for the answer of the query.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain the result $\tilde{\gamma}_{i*}^{(q)}$ of the tampering on one commitment γ_{i*} such that $i^* \in \mathcal{T}$;
  - use another leakage query to check if all the tampered commitments correspond, and return $\perp$ if not;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is "good", return $\mu^{(q)}$ to the adversary.
- Upon input the last tampering query $(\mathcal{T}^{(p+1)}, f^{(p+1)})$:
  - construct a special leakage query checking that the simulation did not cause any inconsistency so far and outputs a bit b_{ok};
  - obtain the tampered commitment $\tilde{\gamma}_{i*}^{(p+1)}$ as in the previous queries;
  - forward the tampering query to the oracle and check that $\tilde{\gamma}_{i*}^{(p+1)}$ is a valid commitment for the answer of the query.
- Output the same distinguishing bit as the adversary if $b_{\mathsf{ok}} = \mathsf{ok}$ and 0 if $b_{\mathsf{ok}} = \mathsf{error}$.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain the result $\tilde{\gamma}_{i*}^{(q)}$ of the tampering on one commitment γ_{i*} such that $i^* \in \mathcal{T}$;
  - use another leakage query to check if all the tampered commitments correspond, and return $\perp$ if not;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is "good", return $\mu^{(q)}$ to the adversary.
- Upon input the last tampering query $(\mathcal{T}^{(p+1)}, f^{(p+1)})$:
  - construct a special leakage query checking that the simulation did not cause any inconsistency so far and outputs a bit b_{ok};
  - obtain the tampered commitment $\tilde{\gamma}_{i*}^{(p+1)}$ as in the previous queries;
  - forward the tampering query to the oracle and check that $\tilde{\gamma}_{i*}^{(p+1)}$ is a valid commitment for the answer of the query.
- Output the same distinguishing bit as the adversary if $b_{\text{ok}} = \text{ok}$ and 0 if $b_{\text{ok}} = \text{error}$.

Leakage analysis

The total leakage performed by the reduction amounts to $(p+1) \cdot (|\gamma| + n) + 1$.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Final step

- We proved that the original game and the hybrid game are statistically close.

[BFOSV20] “Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model”, *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Final step

- We proved that the original game and the hybrid game are statistically close.
- **Hybrid game:**
  - sample a random message $\hat{\mu}$ and random coins $\rho, \hat{\rho}$ and compute $\gamma = \mathsf{Commit}(\mu_b; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\!\$\ \mathsf{Share}(\hat{\mu}||\hat{\rho})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Final step

- We proved that the original game and the hybrid game are statistically close.
- **Hybrid game:**
  - sample a random message $\hat{\mu}$ and random coins $\rho, \hat{\rho}$ and compute $\gamma = \text{Commit}(\mu_b; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \xleftarrow{\$} \text{Share}(\hat{\mu}||\hat{\rho})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- For any two messages μ_0, μ_1, the above game with $b = 0$ and with $b = 1$ are computationally close.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Achieving multiple tampering queries [BFOSV20] — Proof

Final step

- We proved that the original game and the hybrid game are statistically close.
- **Hybrid game:**
  - sample a random message $\hat{\mu}$ and random coins $\rho, \hat{\rho}$ and compute $\gamma = \text{Commit}(\mu_b; \rho)$;
  - compute $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\$\ \text{Share}(\hat{\mu} || \hat{\rho})$;
  - output $(\sigma_1^*, \ldots, \sigma_n^*)$, where, for each $i \in [n]$, $\sigma_i^* = (\gamma, \sigma_i)$.
- For any two messages μ_0, μ_1, the above game with $b = 0$ and with $b = 1$ are computationally close.
- *Proof:* by reduction to the computational hiding property of the commitment scheme.

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

Digression on the non-standard noisy-leakage notion

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

$$\tilde{\mathbb{H}}_\infty\left(\boldsymbol{\Sigma}_i \mid \boldsymbol{\Lambda}_i\right) \geq \mathbb{H}_\infty\left(\boldsymbol{\Sigma}_i\right) - \ell$$

Digression on the non-standard noisy-leakage notion

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

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

- **Admissible adversaries, non-standard version:** an adversary A is ℓ-admissible if it is allowed to ask as many leakage queries he wants, chosen adaptively, as long as, for all $i \in [n]$,

$$\tilde{\mathbb{H}}_\infty\left(\mathbf{\Sigma}_i \mid (\mathbf{\Sigma}_j)_{j\neq i}, \mathbf{\Lambda}_i\right) \geq \tilde{\mathbb{H}}_\infty\left(\mathbf{\Sigma}_i \mid (\mathbf{\Sigma}_j)_{j\neq i}\right) - \ell$$

Digression on the non-standard noisy-leakage notion

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

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

- **Admissible adversaries, non-standard version:** an adversary A is ℓ-admissible if it is allowed to ask as many leakage queries he wants, chosen adaptively, as long as, for all $i \in [n]$,

$$\tilde{\mathbb{H}}_\infty\left(\mathbf{\Sigma}_i \mid (\mathbf{\Sigma}_j)_{j \neq i}, \mathbf{\Lambda}_i\right) \geq \tilde{\mathbb{H}}_\infty\left(\mathbf{\Sigma}_i \mid (\mathbf{\Sigma}_j)_{j \neq i}\right) - \ell$$

Note: the non-standard version is tricky and “dangerous”, since there are many more leakage queries performing 0 bits of noisy leakage, and some of them could even break non-malleability.

Achieving multiple tampering queries [BFV19] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.

[BFV19] "Continuously Non-Malleable Secret Sharing for General Access Structures", *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- Forward all the leakage queries to the leakage oracle.

[BFV19] "Continuously Non-Malleable Secret Sharing for General Access Structures", *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- Forward all the leakage queries to the leakage oracle.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:

[BFV19] “Continuously Non-Malleable Secret Sharing for General Access Structures”, *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- Forward all the leakage queries to the leakage oracle.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain all the mauled commitments $(\tilde{\gamma}_i^{(q)})_{i \in \mathcal{T}^{(q)}}$ by hard-wiring the tampering functions;

[BFV19] “Continuously Non-Malleable Secret Sharing for General Access Structures”, *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- Forward all the leakage queries to the leakage oracle.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain all the mauled commitments $(\tilde{\gamma}_i^{(q)})_{i \in \mathcal{T}^{(q)}}$ by hard-wiring the tampering functions;
  - check that the leaked commitments are all the same and, if not, return $\perp$;

[BFV19] "Continuously Non-Malleable Secret Sharing for General Access Structures", *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- Forward all the leakage queries to the leakage oracle.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain all the mauled commitments $(\tilde{\gamma}_i^{(q)})_{i \in \mathcal{T}^{(q)}}$ by hard-wiring the tampering functions;
  - check that the leaked commitments are all the same and, if not, return $\perp$;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;

[BFV19] "Continuously Non-Malleable Secret Sharing for General Access Structures", *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- Forward all the leakage queries to the leakage oracle.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain all the mauled commitments $(\tilde{\gamma}_i^{(q)})_{i \in \mathcal{T}^{(q)}}$ by hard-wiring the tampering functions;
  - check that the leaked commitments are all the same and, if not, return $\perp$;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is "good", return $\mu^{(q)}$ to the adversary.

[BFV19] "Continuously Non-Malleable Secret Sharing for General Access Structures", *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- Forward all the leakage queries to the leakage oracle.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain all the mauled commitments $(\tilde{\gamma}_i^{(q)})_{i \in \mathcal{T}^{(q)}}$ by hard-wiring the tampering functions;
  - check that the leaked commitments are all the same and, if not, return $\perp$;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is "good", return $\mu^{(q)}$ to the adversary.
- Upon input the last tampering query $(\mathcal{T}^{(p+1)}, f^{(p+1)})$:
  - construct a special leakage query checking that the simulation did not cause any inconsistency so far and outputs a bit b_{ok};

[BFV19] "Continuously Non-Malleable Secret Sharing for General Access Structures", *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- Forward all the leakage queries to the leakage oracle.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain all the mauled commitments $(\tilde{\gamma}_i^{(q)})_{i \in \mathcal{T}^{(q)}}$ by hard-wiring the tampering functions;
  - check that the leaked commitments are all the same and, if not, return $\perp$;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is "good", return $\mu^{(q)}$ to the adversary.
- Upon input the last tampering query $(\mathcal{T}^{(p+1)}, f^{(p+1)})$:
  - construct a special leakage query checking that the simulation did not cause any inconsistency so far and outputs a bit b_{ok};
  - obtain and check all the mauled commitments $(\tilde{\gamma}_i^{(p+1)})_{i \in \mathcal{T}^{(p+1)}}$ as in the previous queries;

[BFV19] "Continuously Non-Malleable Secret Sharing for General Access Structures", *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- Forward all the leakage queries to the leakage oracle.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain all the mauled commitments $(\tilde{\gamma}_i^{(q)})_{i \in \mathcal{T}^{(q)}}$ by hard-wiring the tampering functions;
  - check that the leaked commitments are all the same and, if not, return $\perp$;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is "good", return $\mu^{(q)}$ to the adversary.
- Upon input the last tampering query $(\mathcal{T}^{(p+1)}, f^{(p+1)})$:
  - construct a special leakage query checking that the simulation did not cause any inconsistency so far and outputs a bit b_{ok};
  - obtain and check all the mauled commitments $(\tilde{\gamma}_i^{(p+1)})_{i \in \mathcal{T}^{(p+1)}}$ as in the previous queries;
  - forward the tampering query to the oracle and check that $\tilde{\gamma}_{i*}^{(p+1)}$ is a valid commitment for the answer of the query.

[BFV19] "Continuously Non-Malleable Secret Sharing for General Access Structures", *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Proof

Inductive step

- By reduction to statistical leakage-resilience one-time non-malleability.
- Forward all the leakage queries to the leakage oracle.
- For each tampering query $(\mathcal{T}^{(q)}, f^{(q)})$:
  - use a leakage query in order to obtain all the mauled commitments $(\tilde{\gamma}_i^{(q)})_{i \in \mathcal{T}^{(q)}}$ by hard-wiring the tampering functions;
  - check that the leaked commitments are all the same and, if not, return $\perp$;
  - find by brute force the opening $(\mu^{(q)}, \rho^{(q)})$ of $\tilde{\gamma}_{i*}$, and return $\perp$ if no such value is found;
  - after checking that $\mu^{(q)}$ is “good”, return $\mu^{(q)}$ to the adversary.
- Upon input the last tampering query $(\mathcal{T}^{(p+1)}, f^{(p+1)})$:
  - construct a special leakage query checking that the simulation did not cause any inconsistency so far and outputs a bit b_{ok};
  - obtain and check all the mauled commitments $(\tilde{\gamma}_i^{(p+1)})_{i \in \mathcal{T}^{(p+1)}}$ as in the previous queries;
  - forward the tampering query to the oracle and check that $\tilde{\gamma}_{i*}^{(p+1)}$ is a valid commitment for the answer of the query.
- Output the same distinguishing bit as the adversary if $b_{\mathrm{ok}} = \mathrm{ok}$ and 0 if $b_{\mathrm{ok}} = \mathrm{error}$.

[BFV19] “Continuously Non-Malleable Secret Sharing for General Access Structures”, *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Leakage analysis

For all $i \in [n]$,

$$\tilde{\mathbb{H}}\left(\mathbf{\Sigma}_i \mid (\mathbf{\Sigma}_j)_{j \neq i}, \mathbf{\Lambda}_i\right)$$

[BFV19] “Continuously Non-Malleable Secret Sharing for General Access Structures”, *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Leakage analysis

For all $i \in [n]$,

$$\tilde{\mathbb{H}}\left(\mathbf{\Sigma}_i \mid (\mathbf{\Sigma}_j)_{j \neq i}, \mathbf{\Lambda}_i\right) \geq \tilde{\mathbb{H}}\left(\mathbf{\Sigma}_i \mid (\mathbf{\Sigma}_j)_{j \neq i}, \tilde{\gamma}_i^{(1)}, \ldots, \tilde{\gamma}_i^{(\mathbf{q}_{\mathsf{sd}})}, \tilde{\gamma}_i^{(p+1)}\right) - \ell' - 1$$

[BFV19] “Continuously Non-Malleable Secret Sharing for General Access Structures”, *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Leakage analysis

For all $i \in [n]$,

$$\tilde{\mathbb{H}}\left(\boldsymbol{\Sigma}_i \mid (\boldsymbol{\Sigma}_j)_{j\neq i}, \boldsymbol{\Lambda}_i\right) \geq \tilde{\mathbb{H}}\left(\boldsymbol{\Sigma}_i \mid (\boldsymbol{\Sigma}_j)_{j\neq i}, \tilde{\gamma}_i^{(1)}, \ldots, \tilde{\gamma}_i^{(\mathbf{q}_{\mathsf{sd}})}, \tilde{\gamma}_i^{(p+1)}\right) - \ell' - 1$$
$$\geq \tilde{\mathbb{H}}\left(\boldsymbol{\Sigma}_i \mid (\boldsymbol{\Sigma}_j)_{j\neq i}, \mathbf{q}_{\mathbf{sd}}, \tilde{\gamma}_i^{(\mathbf{q}_{\mathsf{sd}})}, \tilde{\gamma}_i^{(p+1)}\right) - \ell' - 1$$

[BFV19] “Continuously Non-Malleable Secret Sharing for General Access Structures”, *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Leakage analysis

For all $i \in [n]$,

$$\begin{aligned}\tilde{\mathbb{H}}\left(\mathbf{\Sigma}_i \mid (\mathbf{\Sigma}_j)_{j\neq i}, \mathbf{\Lambda}_i\right) &\geq \tilde{\mathbb{H}}\left(\mathbf{\Sigma}_i \mid (\mathbf{\Sigma}_j)_{j\neq i}, \tilde{\gamma}_i^{(1)}, \ldots, \tilde{\gamma}_i^{(\mathbf{q}_{\mathsf{sd}})}, \tilde{\gamma}_i^{(p+1)}\right) - \ell' - 1 \\ &\geq \tilde{\mathbb{H}}\left(\mathbf{\Sigma}_i \mid (\mathbf{\Sigma}_j)_{j\neq i}, \mathbf{q}_{\mathsf{sd}}, \tilde{\gamma}_i^{(\mathbf{q}_{\mathsf{sd}})}, \tilde{\gamma}_i^{(p+1)}\right) - \ell' - 1 \\ &\geq \tilde{\mathbb{H}}\left(\mathbf{\Sigma}_i \mid (\mathbf{\Sigma}_j)_{j\neq i}\right) - \ell' - 1 - |\gamma| - O(\log(\lambda)).\end{aligned}$$

[BFV19] "Continuously Non-Malleable Secret Sharing for General Access Structures", *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Achieving multiple tampering queries [BFV19] — Leakage analysis

For all $i \in [n]$,

$$\begin{aligned}
\tilde{\mathbb{H}}\left(\boldsymbol{\Sigma}_i \mid (\boldsymbol{\Sigma}_j)_{j\neq i}, \boldsymbol{\Lambda}_i\right) &\geq \tilde{\mathbb{H}}\left(\boldsymbol{\Sigma}_i \mid (\boldsymbol{\Sigma}_j)_{j\neq i}, \tilde{\gamma}_i^{(1)}, \ldots, \tilde{\gamma}_i^{(\mathbf{q}_{\mathsf{sd}})}, \tilde{\gamma}_i^{(p+1)}\right) - \ell' - 1 \\
&\geq \tilde{\mathbb{H}}\left(\boldsymbol{\Sigma}_i \mid (\boldsymbol{\Sigma}_j)_{j\neq i}, \mathbf{q}_{\mathsf{sd}}, \tilde{\gamma}_i^{(\mathbf{q}_{\mathsf{sd}})}, \tilde{\gamma}_i^{(p+1)}\right) - \ell' - 1 \\
&\geq \tilde{\mathbb{H}}\left(\boldsymbol{\Sigma}_i \mid (\boldsymbol{\Sigma}_j)_{j\neq i}\right) - \ell' - 1 - |\gamma| - O(\log(\lambda)).
\end{aligned}$$

Therefore, the overall performed leakage by the reduction amounts to $\ell = \ell' + 1 + |\gamma| + O(\log(\lambda))$.

[BFV19] "Continuously Non-Malleable Secret Sharing for General Access Structures", *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

Open problems

- p-time/continuously non-malleable secret sharing against joint leakage and tampering.

Open problems

- p-time/continuously non-malleable secret sharing against joint leakage and tampering.
- Non-malleability against adaptive partitioning.

Open problems

- p-time/continuously non-malleable secret sharing against joint leakage and tampering.
- Non-malleability against adaptive partitioning.
- Optimal rate.

Open problems

- p-time/continuously non-malleable secret sharing against joint leakage and tampering.
- Non-malleability against adaptive partitioning.
- Optimal rate.
    - Informally, is the ratio between $|\mu|$ and $\max_{i \in [n]} |\sigma_i|$.

[Kra93] "Secret Sharing made Short", *Hugo Krawczyk*, CRYPTO 1993

Open problems

- p-time/continuously non-malleable secret sharing against joint leakage and tampering.
- Non-malleability against adaptive partitioning.
- Optimal rate.
  - Informally, is the ratio between $|\mu|$ and $\max_{i \in [n]} |\sigma_i|$.
  - t for standard secret sharing [Kra93].

[Kra93] "Secret Sharing made Short", *Hugo Krawczyk*, CRYPTO 1993

References

[ADN+20] “Stronger Leakage-Resilient and Non-Malleable Secret-Sharing Schemes for General Access Structures”, *Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João Ribeiro, Mark Simkin*, CRYPTO 2019

[BFOSV20] “Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model”, *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

[BFV19] “Continuously Non-Malleable Secret Sharing for General Access Structures”, *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

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

[CGGL] “Leakage-Resilient Extractors and Secret-Sharing against Bounded Collusion Protocols”, *Eshan Chattopadhyay, Jesse Goodman, Vipul Goyal, Xin Li*, Electronic Colloquium on Computational Complexity, Volume 27, 2020

[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

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

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

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

[GSZ20] “Multi-Source Non-Malleable Extractors and Applications”, *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

[Kra93] “Secret Sharing made Short”, *Hugo Krawczyk*, CRYPTO 1993

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

[KMZ20] “Bounded Collusion Protocols, Cylinder-Intersection Extractors and Leakage-Resilient Secret Sharing”, *Ashutosh Kumar, Raghu Meka, David Zuckerman*, Electronic Colloquium on Computational Complexity, Volume 27, 2020

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

[SV18] “Leakage Resilient Secret Sharing and Applications”, *Akshayaram Srinivasan, Prashant Nalini Vasudevan*, CRYPTO 2019

References

[ADN+20] "Stronger Leakage-Resilient and Non-Malleable Secret-Sharing Schemes for General Access Structures", *Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João Ribeiro, Mark Simkin*, CRYPTO 2019

[BFOSV20] "Non-Malleable Secret Sharing against Bounded Joint-Tampering Attacks in the Plain Model", *Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, Daniele Venturi*, CRYPTO 2020

[BFV19] "Continuously Non-Malleable Secret Sharing for General Access Structures", *Gianluca Brian, Antonio Faonio, Daniele Venturi*, TCC 2019

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

[CGGL] "Leakage-Resilient Extractors and Secret-Sharing against Bounded Collusion Protocols", *Eshan Chattopadhyay, Jesse Goodman, Vipul Goyal, Xin Li*, Electronic Colloquium on Computational Complexity, Volume 27, 2020

[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

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

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

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

[GSZ20] "Multi-Source Non-Malleable Extractors and Applications", *Vipul Goyal, Akshayaram Srinivasan, Chenzhi Zhu*, IACR Cryptology ePrint Archive, Vol.2020/157

[Kra93] "Secret Sharing made Short", *Hugo Krawczyk*, CRYPTO 1993

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

[KMZ20] "Bounded Collusion Protocols, Cylinder-Intersection Extractors and Leakage-Resilient Secret Sharing", *Ashutosh Kumar, Raghu Meka, David Zuckerman*, Electronic Colloquium on Computational Complexity, Volume 27, 2020

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

[SV18] "Leakage Resilient Secret Sharing and Applications", *Akshayaram Srinivasan, Prashant Nalini Vasudevan*, CRYPTO 2019

THANK YOU!!!