Secret Sharing with Publicly Verifiable Deletion

Jonathan Katz[1] Ben Sela[2]

[1]Google

[2]University of Maryland

Certified deletion

Certified deletion

- Secure key leasing: revoking access to functionality

Certified deletion

- Secure key leasing: revoking access to functionality
- Certified deletion in our setting: deleting classical information

Certified deletion

- Secure key leasing: revoking access to functionality
- Certified deletion in our setting: deleting classical information

User

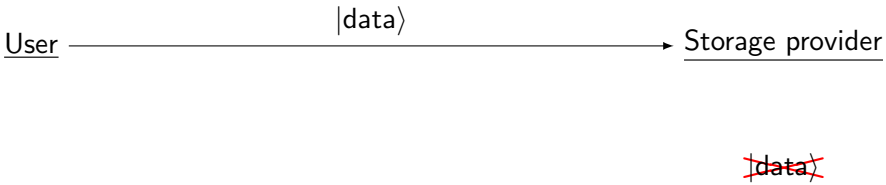
Storage provider

Certified deletion

- Secure key leasing: revoking access to functionality
- Certified deletion in our setting: deleting classical information

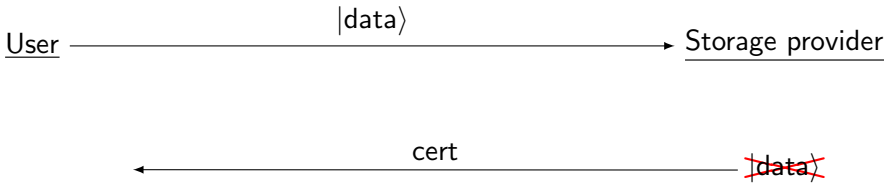
Certified deletion

- Secure key leasing: revoking access to functionality
- Certified deletion in our setting: deleting classical information



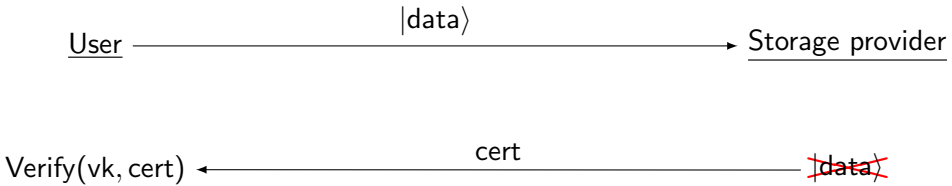
Certified deletion

- Secure key leasing: revoking access to functionality
- Certified deletion in our setting: deleting classical information



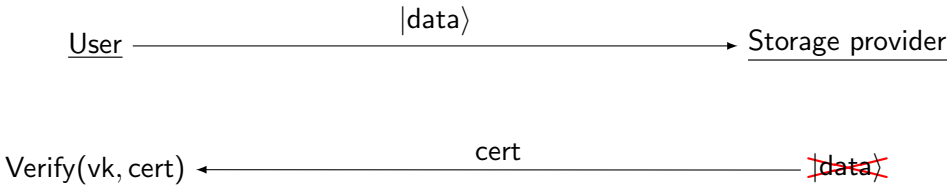
Certified deletion

- Secure key leasing: revoking access to functionality
- Certified deletion in our setting: deleting classical information



Certified deletion

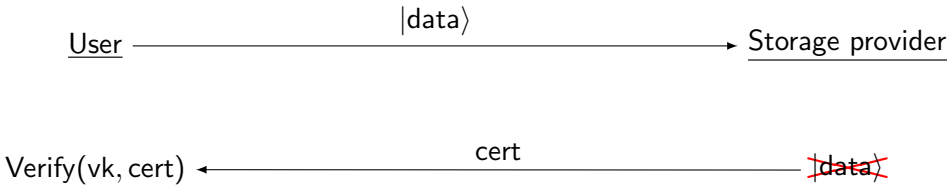
- Secure key leasing: revoking access to functionality
- Certified deletion in our setting: deleting classical information



- **privately verifiable deletion**: Need to hide vk for deletion security to hold.

Certified deletion

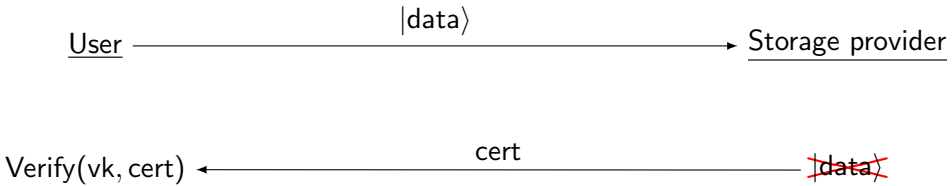
- Secure key leasing: revoking access to functionality
- Certified deletion in our setting: deleting classical information



- **privately verifiable deletion**: Need to hide vk for deletion security to hold.
- Advantages of **publicly verifiable deletion (PVD)**:

Certified deletion

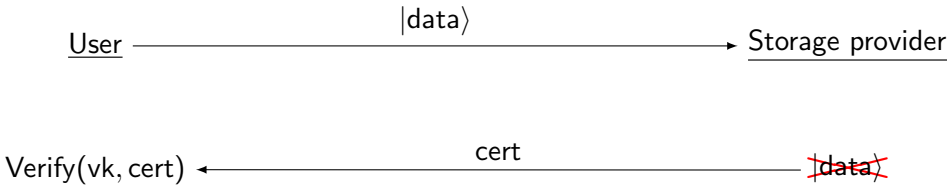
- Secure key leasing: revoking access to functionality
- Certified deletion in our setting: deleting classical information



- **privately verifiable deletion**: Need to hide vk for deletion security to hold.
- Advantages of **publicly verifiable deletion (PVD)**:
  - Any third party can verify deletion proofs

Certified deletion

- Secure key leasing: revoking access to functionality
- Certified deletion in our setting: deleting classical information



- **privately verifiable deletion**: Need to hide vk for deletion security to hold.
- Advantages of **publicly verifiable deletion (PVD)**:
  - Any third party can verify deletion proofs
  - Do not need to store verification keys securely

Secret sharing

Definition: Secret sharing scheme

Secret sharing

Definition: Secret sharing scheme

Classical Secret Sharing

- Parameterized by a monotone access structure defining the authorized sets

Secret sharing

Definition: Secret sharing scheme

Classical Secret Sharing

- Parameterized by a monotone access structure defining the authorized sets
- Share(s) : Outputs shares $\mathsf{sh}_1, ..., \mathsf{sh}_n$.

Secret sharing

Definition: Secret sharing scheme

Classical Secret Sharing

- Parameterized by a monotone access structure defining the authorized sets
- Share(s) : Outputs shares $\mathrm{sh}_1, ..., \mathrm{sh}_n$.
- Reconstruct($\{\mathrm{sh}_i\}_{i \in A}$) : If A is an authorized set, outputs the secret s.

Secret sharing

Definition: Secret sharing scheme

Classical Secret Sharing

- Parameterized by a monotone access structure defining the authorized sets
- Share(s) : Outputs shares $\mathsf{sh}_1, ..., \mathsf{sh}_n$.
- Reconstruct($\{\mathsf{sh}_i\}_{i \in A}$) : If A is an authorized set, outputs the secret s.

Security: If $A \subseteq [n]$ is not authorized, then

$$\mathsf{SD}\left(\{\mathsf{sh}_i^0\}_{i \in A}, \{\mathsf{sh}_i^1\}_{i \in A}\right) \leq \epsilon.$$

Secret sharing with certified deletion [BR24]

Definition: Secret sharing with certified deletion [BR24]

- $\mathsf{Share}(1^\lambda, s)$: Outputs shares $|\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle$, and verification key vk.

Secret sharing with certified deletion [BR24]

Definition: Secret sharing with certified deletion [BR24]

- $\mathsf{Share}(1^\lambda, s)$: Outputs shares $|\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle$, and verification key vk.
- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: If A is an authorized set, outputs the secret s.

Secret sharing with certified deletion [BR24]

Definition: Secret sharing with certified deletion [BR24]

- $\mathsf{Share}(1^\lambda, s)$: Outputs shares $|\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle$, and verification key vk.
- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: If A is an authorized set, outputs the secret s.
- $\mathsf{Delete}(|\mathsf{qsh}\rangle)$: Outputs a classical deletion proof cert.

Secret sharing with certified deletion [BR24]

Definition: Secret sharing with certified deletion [BR24]

- $\mathsf{Share}(1^\lambda, s)$: Outputs shares $|\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle$, and verification key vk.
- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: If A is an authorized set, outputs the secret s.
- $\mathsf{Delete}(|\mathsf{qsh}\rangle)$: Outputs a classical deletion proof cert.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert})$: Outputs $\top$ if cert is valid, and outputs $\bot$ otherwise.

Secret sharing with certified deletion [BR24]

Definition: Secret sharing with certified deletion [BR24]

- $\mathsf{Share}(1^\lambda, s)$: Outputs shares $|\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle$, and verification key vk.
- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: If A is an authorized set, outputs the secret s.
- $\mathsf{Delete}(|\mathsf{qsh}\rangle)$: Outputs a classical deletion proof cert.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert})$: Outputs $\top$ if cert is valid, and outputs $\bot$ otherwise.

Adaptive certified deletion experiment [BR24]

Secret sharing with certified deletion [BR24]

Definition: Secret sharing with certified deletion [BR24]

- $\mathsf{Share}(1^\lambda, s)$: Outputs shares $|\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle$, and verification key vk.
- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: If A is an authorized set, outputs the secret s.
- $\mathsf{Delete}(|\mathsf{qsh}\rangle)$: Outputs a classical deletion proof cert.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert})$: Outputs $\top$ if cert is valid, and outputs $\bot$ otherwise.

Adaptive certified deletion experiment [BR24]

- Generate verification key and quantum shares $\mathsf{vk}, |\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle \leftarrow \mathsf{Share}(1^\lambda, s)$

Secret sharing with certified deletion [BR24]

Definition: Secret sharing with certified deletion [BR24]

- $\mathsf{Share}(1^\lambda, s)$: Outputs shares $|\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle$, and verification key vk.
- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: If A is an authorized set, outputs the secret s.
- $\mathsf{Delete}(|\mathsf{qsh}\rangle)$: Outputs a classical deletion proof cert.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert})$: Outputs $\top$ if cert is valid, and outputs $\bot$ otherwise.

Adaptive certified deletion experiment [BR24]

- Generate verification key and quantum shares $\mathsf{vk}, |\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle \leftarrow \mathsf{Share}(1^\lambda, s)$
- Adversary $\mathcal{A}$ adaptively corrupts and deletes shares such that...

Secret sharing with certified deletion [BR24]

Definition: Secret sharing with certified deletion [BR24]

- $\mathsf{Share}(1^\lambda, s)$: Outputs shares $|\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle$, and verification key vk.
- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: If A is an authorized set, outputs the secret s.
- $\mathsf{Delete}(|\mathsf{qsh}\rangle)$: Outputs a classical deletion proof cert.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert})$: Outputs $\top$ if cert is valid, and outputs $\bot$ otherwise.

Adaptive certified deletion experiment [BR24]

- Generate verification key and quantum shares $\mathsf{vk}, |\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle \leftarrow \mathsf{Share}(1^\lambda, s)$
- Adversary $\mathcal{A}$ adaptively corrupts and deletes shares such that...
  - the set of shares that are corrupted but not deleted is never authorized.

Secret sharing with certified deletion [BR24]

Definition: Secret sharing with certified deletion [BR24]

- $\mathsf{Share}(1^\lambda, s)$: Outputs shares $|\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle$, and verification key vk.
- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: If A is an authorized set, outputs the secret s.
- $\mathsf{Delete}(|\mathsf{qsh}\rangle)$: Outputs a classical deletion proof cert.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert})$: Outputs $\top$ if cert is valid, and outputs $\bot$ otherwise.

Adaptive certified deletion experiment [BR24]

- Generate verification key and quantum shares $\mathsf{vk}, |\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle \leftarrow \mathsf{Share}(1^\lambda, s)$
- Adversary $\mathcal{A}$ adaptively corrupts and deletes shares such that...
  - the set of shares that are corrupted but not deleted is never authorized.
- Output $|\mathsf{State}_s\rangle$, the state of $\mathcal{A}$.

Adaptive certified deletion

Adaptive certified deletion experiment [BR24]

- Generate verification key and quantum shares $\mathsf{vk}, |\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle \leftarrow \mathsf{Share}(1^\lambda, s)$
- Adversary $\mathcal{A}$ adaptively corrupts and deletes shares such that...
  - the set of shares that are corrupted but not deleted is never authorized.
- Output $|\mathsf{State}_s\rangle$, the state of $\mathcal{A}$.

- Computational security: If $\mathcal{A}$ is bounded, then secret s is **computationally** hidden given State_s, i.e.,

$$|\mathsf{State}_{s_0}\rangle \approx_c |\mathsf{State}_{s_1}\rangle \quad \forall s_0, s_1$$

Adaptive certified deletion

Adaptive certified deletion experiment [BR24]

- Generate verification key and quantum shares $\mathsf{vk}, |\mathsf{qsh}_1\rangle, ..., |\mathsf{qsh}_n\rangle \leftarrow \mathsf{Share}(1^\lambda, s)$
- Adversary $\mathcal{A}$ adaptively corrupts and deletes shares such that...
  - the set of shares that are corrupted but not deleted is never authorized.
- Output $|\mathsf{State}_s\rangle$, the state of $\mathcal{A}$.

- Computational security: If $\mathcal{A}$ is bounded, then secret s is **computationally** hidden given State_s, i.e.,

$$|\mathsf{State}_{s_0}\rangle \approx_c |\mathsf{State}_{s_1}\rangle \quad \forall s_0, s_1$$

- Everlasting security: If $\mathcal{A}$ is bounded *during adaptive certified deletion experiment*, then secret s is **information-theoretically** hidden given State_s, i.e.,

$$\mathsf{TD}(|\mathsf{State}_{s_0}\rangle, |\mathsf{State}_{s_1}\rangle) \leq \mathsf{negl}(\lambda) \quad \forall s_0, s_1$$

Prior work on secret sharing with certified deletion

Prior results [BR24]:

Prior work on secret sharing with certified deletion

Prior results [BR24]:

- Two (incomparable) security notions:

Prior work on secret sharing with certified deletion

Prior results [BR24]:

- Two (incomparable) security notions:
  - No-signaling certified deletion

Prior work on secret sharing with certified deletion

Prior results [BR24]:

- Two (incomparable) security notions:
  - No-signaling certified deletion
  - Adaptive certified deletion

Prior work on secret sharing with certified deletion

Prior results [BR24]:

- Two (incomparable) security notions:
  - No-signaling certified deletion
  - Adaptive certified deletion
- Constructions with **privately verifiable** deletion (for both security notions)

Prior work on secret sharing with certified deletion

Prior results [BR24]:

- Two (incomparable) security notions:
  - No-signaling certified deletion
  - Adaptive certified deletion
- Constructions with **privately verifiable** deletion (for both security notions)
  - Secure against unbounded adversary

Summary of our contributions

	Publicly verifiable deletion	Monotone access structures
No-signaling security	• Prior work: ✗	• Prior work: ✓
Adaptive security	• Prior work: ✗	• Prior work: ✗

Summary of our contributions

	Publicly verifiable deletion	Monotone access structures
No-signaling security	• Prior work: ✗ • Our result: ✓	• Prior work: ✓ • Our result: ✓
Adaptive security	• Prior work: ✗ • Our result: ✓	• Prior work: ✗ • Our result: ✓

Summary of our contributions

- Assume information-theoretic classical secret sharing scheme for monotone access structure $\mathbb{A}$ (with share size m)

	Security	Assumptions	Number of qubits
Construction 1	Adaptive PVD	OWF	$O(n \cdot m)$
Construction 2	Adaptive PVD	LWE	$O(m)$
Construction 3	No-signaling PVD	sub-exponentially secure OWF	$O(m)$

Summary of our contributions

- Assume information-theoretic classical secret sharing scheme for monotone access structure $\mathbb{A}$ (with share size m)

	Security	Assumptions	Number of qubits
Construction 1	Adaptive PVD	OWF	$O(n \cdot m)$
Construction 2	Adaptive PVD	LWE	$O(m)$
Construction 3	No-signaling PVD	sub-exponentially secure OWF	$O(m)$

- Adaptive PVD with computational security $\implies$ Adaptive PVD with everlasting security.

Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

- **Encode**$(b \in \{0,1\})$:

Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

- **Encode**$(b \in \{0,1\})$:
  - Sample uniform $x_0, x_1 \leftarrow \{0,1\}^{\ell_{\text{owf}}}$

Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

- **Encode**$(b \in \{0,1\})$:
  - Sample uniform $x_0, x_1 \leftarrow \{0,1\}^{\ell_{\text{owf}}}$
  - prepare the state

$$|\psi_b\rangle := |x_0\rangle + (-1)^b |x_1\rangle,$$

Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

- **Encode**$(b \in \{0,1\})$:
  - Sample uniform $x_0, x_1 \leftarrow \{0,1\}^{\ell_{\text{owf}}}$
  - prepare the state
$$|\psi_b\rangle := |x_0\rangle + (-1)^b |x_1\rangle,$$
  - and verification key $\text{vk} = \{f(x_0), f(x_1)\}$

Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

- **Encode**$(b \in \{0,1\})$:
  - Sample uniform $x_0, x_1 \leftarrow \{0,1\}^{\ell_{\text{owf}}}$
  - prepare the state
$$|\psi_b\rangle := |x_0\rangle + (-1)^b |x_1\rangle,$$
  - and verification key $\text{vk} = \{f(x_0), f(x_1)\}$
- **Extract**$(|\psi_b\rangle, x_0 \oplus x_1)$: Measuring $|\psi_b\rangle$ in the Hadamard basis yields a string d such that $d \cdot (x_0 \oplus x_1) = b$

Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

- **Encode**($b \in \{0,1\}$):
  - Sample uniform $x_0, x_1 \leftarrow \{0,1\}^{\ell_{\text{owf}}}$
  - prepare the state
$$|\psi_b\rangle := |x_0\rangle + (-1)^b |x_1\rangle,$$
  - and verification key $\mathsf{vk} = \{f(x_0), f(x_1)\}$
- **Extract**($|\psi_b\rangle$,$x_0 \oplus x_1$): Measuring $|\psi_b\rangle$ in the Hadamard basis yields a string d such that $d \cdot (x_0 \oplus x_1) = b$
- **Delete**($|\psi_b\rangle$): Measure in the computational basis ($\rightarrow |x_0\rangle$ or $|x_1\rangle$) and release the result

Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

- **Encode**$(b \in \{0,1\})$:
  - Sample uniform $x_0, x_1 \leftarrow \{0,1\}^{\ell_{\text{owf}}}$
  - prepare the state
$$|\psi_b\rangle := |x_0\rangle + (-1)^b |x_1\rangle,$$
  - and verification key $\mathsf{vk} = \{f(x_0), f(x_1)\}$
- **Extract**$(|\psi_b\rangle, x_0 \oplus x_1)$: Measuring $|\psi_b\rangle$ in the Hadamard basis yields a string d such that $d \cdot (x_0 \oplus x_1) = b$
- **Delete**$(|\psi_b\rangle)$: Measure in the computational basis ($\rightarrow |x_0\rangle$ or $|x_1\rangle$) and release the result
  - **Verify**$(\mathsf{cert} = x, \mathsf{vk} = \{f(x_0), f(x_1)\})$: Accept if $f(x) \in \{f(x_0), f(x_1)\}$

Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

- **Encode**$(b \in \{0,1\})$:
  - Sample uniform $x_0, x_1 \leftarrow \{0,1\}^{\ell_{\text{owf}}}$
  - prepare the state
$$|\psi_b\rangle := |x_0\rangle + (-1)^b |x_1\rangle,$$
  - and verification key $\mathsf{vk} = \{f(x_0), f(x_1)\}$
- **Extract**$(|\psi_b\rangle, x_0 \oplus x_1)$: Measuring $|\psi_b\rangle$ in the Hadamard basis yields a string d such that $d \cdot (x_0 \oplus x_1) = b$
- **Delete**$(|\psi_b\rangle)$: Measure in the computational basis ($\rightarrow |x_0\rangle$ or $|x_1\rangle$) and release the result
  - **Verify**$(\mathsf{cert} = x, \mathsf{vk} = \{f(x_0), f(x_1)\})$: Accept if $f(x) \in \{f(x_0), f(x_1)\}$
  - **Security:** Assume $x_0 \oplus x_1$ is hidden from $\mathcal{A}$

Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

- **Encode**$(b \in \{0,1\})$:
  - Sample uniform $x_0, x_1 \leftarrow \{0,1\}^{\ell_{\text{owf}}}$
  - prepare the state
$$|\psi_b\rangle := |x_0\rangle + (-1)^b |x_1\rangle,$$
  - and verification key $\mathsf{vk} = \{f(x_0), f(x_1)\}$
- **Extract**$(|\psi_b\rangle, x_0 \oplus x_1)$: Measuring $|\psi_b\rangle$ in the Hadamard basis yields a string d such that $d \cdot (x_0 \oplus x_1) = b$
- **Delete**$(|\psi_b\rangle)$: Measure in the computational basis ($\rightarrow |x_0\rangle$ or $|x_1\rangle$) and release the result
  - **Verify**$(\mathsf{cert} = x, \mathsf{vk} = \{f(x_0), f(x_1)\})$: Accept if $f(x) \in \{f(x_0), f(x_1)\}$
  - **Security:** Assume $x_0 \oplus x_1$ is hidden from $\mathcal{A}$
    - $(x, \mathsf{state}_b) \leftarrow \mathcal{A}(|\psi_b\rangle, \mathsf{vk} := \{f(x_0), f(x_1)\})$

Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

- **Encode**$(b \in \{0,1\})$:
  - Sample uniform $x_0, x_1 \leftarrow \{0,1\}^{\ell_{\text{owf}}}$
  - prepare the state
$$|\psi_b\rangle := |x_0\rangle + (-1)^b |x_1\rangle,$$
  - and verification key $\mathsf{vk} = \{f(x_0), f(x_1)\}$
- **Extract**$(|\psi_b\rangle, x_0 \oplus x_1)$: Measuring $|\psi_b\rangle$ in the Hadamard basis yields a string d such that $d \cdot (x_0 \oplus x_1) = b$
- **Delete**$(|\psi_b\rangle)$: Measure in the computational basis ($\rightarrow |x_0\rangle$ or $|x_1\rangle$) and release the result
  - **Verify**$(\mathsf{cert} = x, \mathsf{vk} = \{f(x_0), f(x_1)\})$: Accept if $f(x) \in \{f(x_0), f(x_1)\}$
  - **Security:** Assume $x_0 \oplus x_1$ is hidden from $\mathcal{A}$
    - $(x, \mathsf{state}_b) \leftarrow \mathcal{A}(|\psi_b\rangle, \mathsf{vk} := \{f(x_0), f(x_1)\})$
    - If $x \in \{f(x_0), f(x_1)\}$, then
$$\mathsf{TD}(\mathsf{state}_0, \mathsf{state}_1) \leq \mathsf{negl}(\lambda)$$

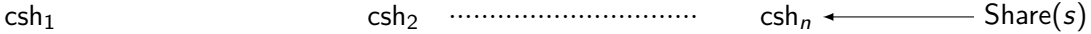
Publicly verifiable deletion—prior work [BKM+23]

Let $f : \{0,1\}^{\ell_{\text{in}}^{\text{owf}}} \mapsto \{0,1\}^{\ell_{\text{out}}^{\text{owf}}}$ be a one-way function.

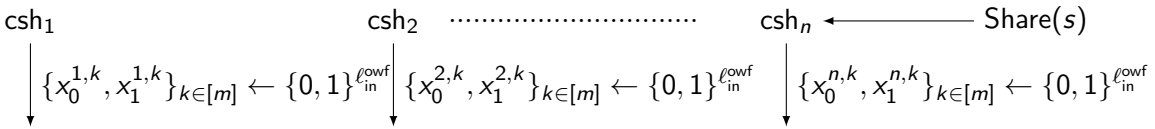
- **Encode**$(b \in \{0,1\})$:
  - Sample uniform $x_0, x_1 \leftarrow \{0,1\}^{\ell_{\text{owf}}}$
  - prepare the state
$$|\psi_b\rangle := |x_0\rangle + (-1)^b |x_1\rangle,$$
  - and verification key $\mathsf{vk} = \{f(x_0), f(x_1)\}$
- **Extract**$(|\psi_b\rangle, x_0 \oplus x_1)$: Measuring $|\psi_b\rangle$ in the Hadamard basis yields a string d such that $d \cdot (x_0 \oplus x_1) = b$
- **Delete**$(|\psi_b\rangle)$: Measure in the computational basis ($\rightarrow |x_0\rangle$ or $|x_1\rangle$) and release the result
  - **Verify**$(\mathsf{cert} = x, \mathsf{vk} = \{f(x_0), f(x_1)\})$: Accept if $f(x) \in \{f(x_0), f(x_1)\}$
  - **Security:** Assume $x_0 \oplus x_1$ is hidden from $\mathcal{A}$
    - $(x, \mathsf{state}_b) \leftarrow \mathcal{A}(|\psi_b\rangle, \mathsf{vk} := \{f(x_0), f(x_1)\})$
    - If $x \in \{f(x_0), f(x_1)\}$, then
$$\mathsf{TD}(\mathsf{state}_0, \mathsf{state}_1) \leq \mathsf{negl}(\lambda)$$

- Note: Preimages must be hidden for deletion security, but knowledge of preimages are required to extract encoded bit b.

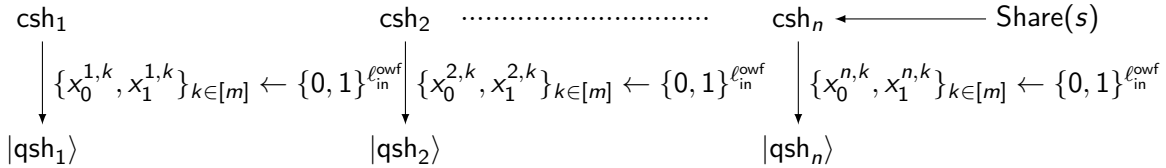
Secret sharing with PVD from OWFs—first attempt



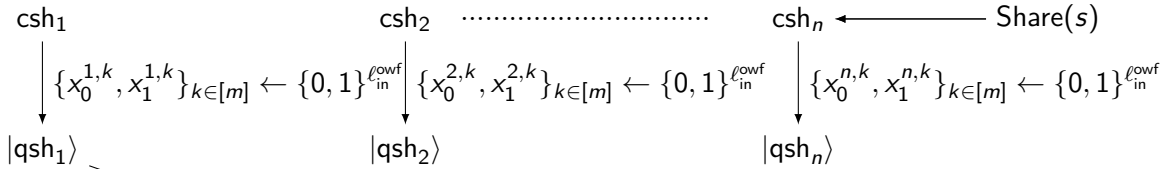
Secret sharing with PVD from OWFs—first attempt



Secret sharing with PVD from OWFs—first attempt

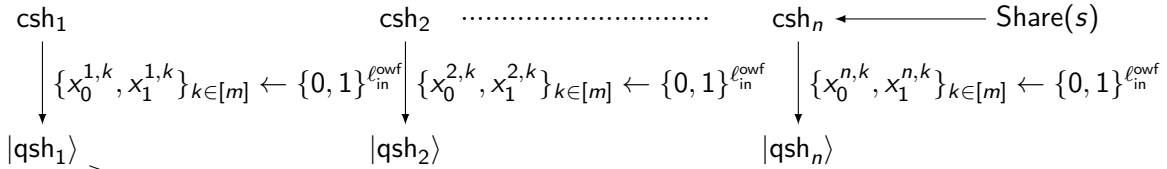


Secret sharing with PVD from OWFs—first attempt



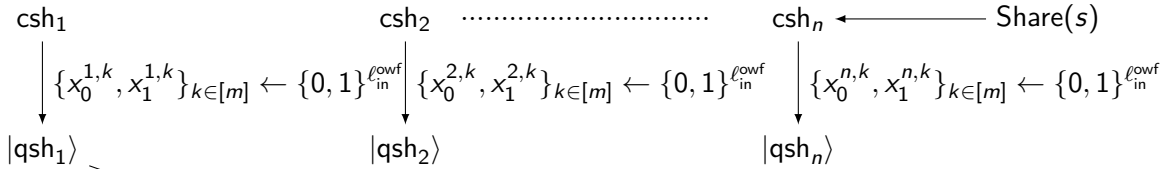
$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]} |x_i^{1,k}\rangle \right)$$

Secret sharing with PVD from OWFs—first attempt



$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]}|x_i^{1,k}\rangle\right) \qquad \mathsf{vk} = \{f(x_0^{i,k}), f(x_1^{i,k})\}_{i\in[n],k\in[m]}$$

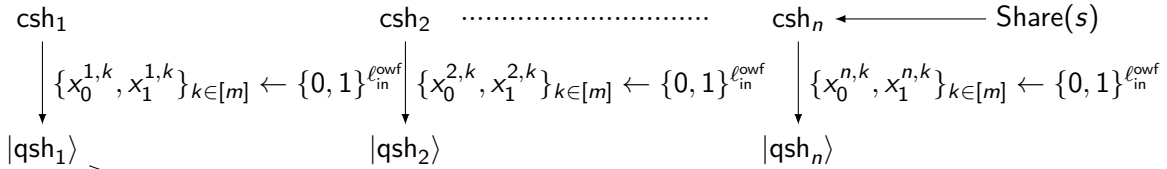
Secret sharing with PVD from OWFs—first attempt



$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]}|x_i^{1,k}\rangle\right) \qquad \mathsf{vk} = \{f(x_0^{i,k}), f(x_1^{i,k})\}_{i\in[n],k\in[m]}$$

- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}$.

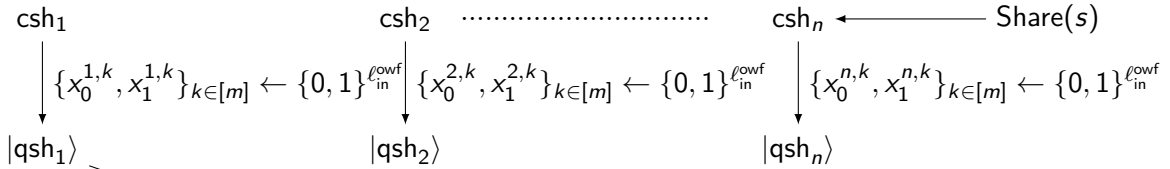
Secret sharing with PVD from OWFs—first attempt



$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]}|x_i^{1,k}\rangle\right) \qquad \mathsf{vk} = \{f(x_0^{i,k}), f(x_1^{i,k})\}_{i\in[n],k\in[m]}$$

- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}$.
  - Compute each bit of the ith classical share as $\mathsf{csh}_i[k] = d_{i,k} \cdot (x_0^{i,k} \oplus x_1^{i,k})$

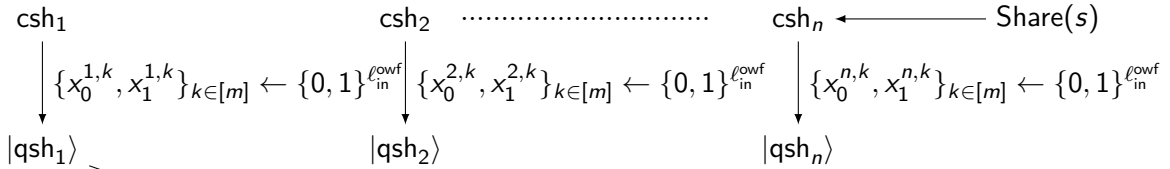
Secret sharing with PVD from OWFs—first attempt



$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]}|x_i^{1,k}\rangle\right) \qquad \mathsf{vk} = \{f(x_0^{i,k}), f(x_1^{i,k})\}_{i\in[n],k\in[m]}$$

- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}$.
  - Compute each bit of the ith classical share as $\mathsf{csh}_i[k] = d_{i,k} \cdot (x_0^{i,k} \oplus x_1^{i,k})$
- $\mathsf{Delete}(|\mathsf{qsh}_i\rangle)$: Measure $|\mathsf{qsh}_i\rangle$ in the computational basis $\rightarrow x_{c_1}^{i,1}, \ldots, x_{c_m}^{i,m}$.

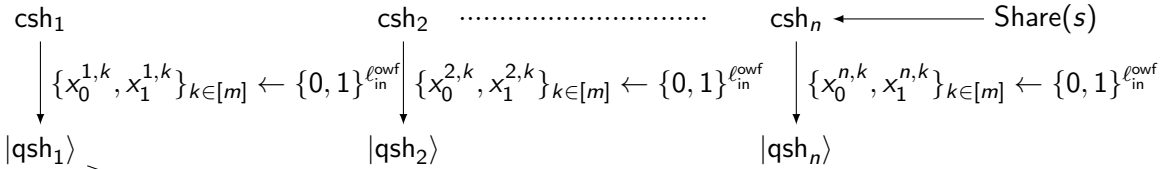
Secret sharing with PVD from OWFs—first attempt



$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]}|x_i^{1,k}\rangle\right) \qquad \mathsf{vk} = \{f(x_0^{i,k}), f(x_1^{i,k})\}_{i\in[n],k\in[m]}$$

- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}$.
  - Compute each bit of the ith classical share as $\mathsf{csh}_i[k] = d_{i,k} \cdot (x_0^{i,k} \oplus x_1^{i,k})$
- $\mathsf{Delete}(|\mathsf{qsh}_i\rangle)$: Measure $|\mathsf{qsh}_i\rangle$ in the computational basis $\rightarrow x_{c_1}^{i,1}, ..., x_{c_m}^{i,m}$.
- $\mathsf{Verify}(\{x_{c_k}^{i,k}\})$: Check that $f(x_{c_k}^{i,k})$ matches correct image in vk for each $k \in [m]$.

Secret sharing with PVD from OWFs—first attempt

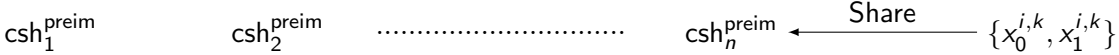


$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]}|x_i^{1,k}\rangle\right) \qquad \mathsf{vk} = \{f(x_0^{i,k}), f(x_1^{i,k})\}_{i\in[n],k\in[m]}$$

- $\mathsf{Reconstruct}(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$: Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}$.
  - Compute each bit of the ith classical share as $\mathsf{csh}_i[k] = d_{i,k} \cdot (x_0^{i,k} \oplus x_1^{i,k})$
- $\mathsf{Delete}(|\mathsf{qsh}_i\rangle)$: Measure $|\mathsf{qsh}_i\rangle$ in the computational basis $\rightarrow x_{c_1}^{i,1}, ..., x_{c_m}^{i,m}$.
- $\mathsf{Verify}(\{x_{c_k}^{i,k}\})$: Check that $f(x_{c_k}^{i,k})$ matches correct image in vk for each $k \in [m]$.
  - Problem: need preimages $x_0^{i,k} \oplus x_1^{i,k}$ to extract the classical share.

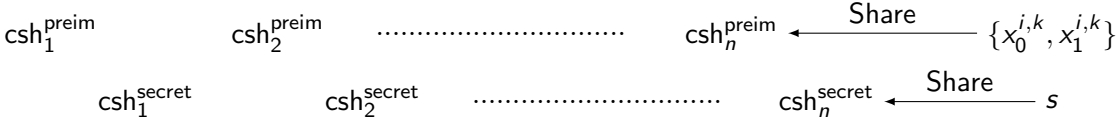
Secret sharing with PVD from OWFs—second attempt

- Idea: Give each party classical share of preimages in addition to quantum share.



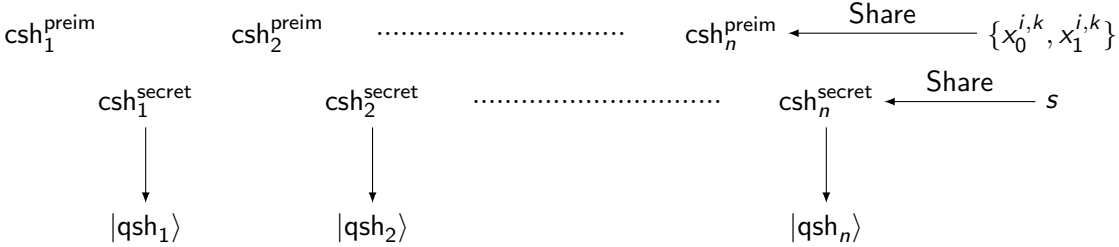
Secret sharing with PVD from OWFs—second attempt

- Idea: Give each party classical share of preimages in addition to quantum share.



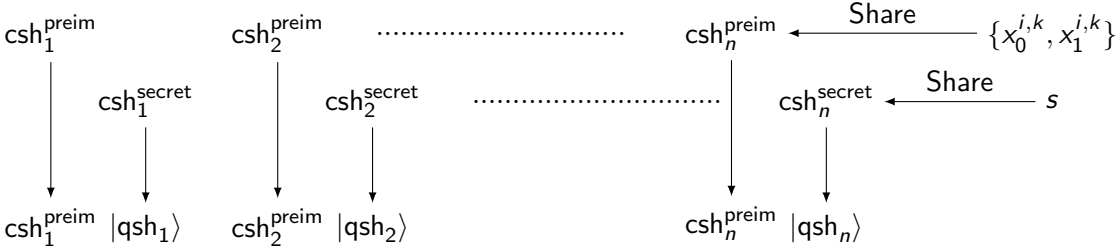
Secret sharing with PVD from OWFs—second attempt

- Idea: Give each party classical share of preimages in addition to quantum share.



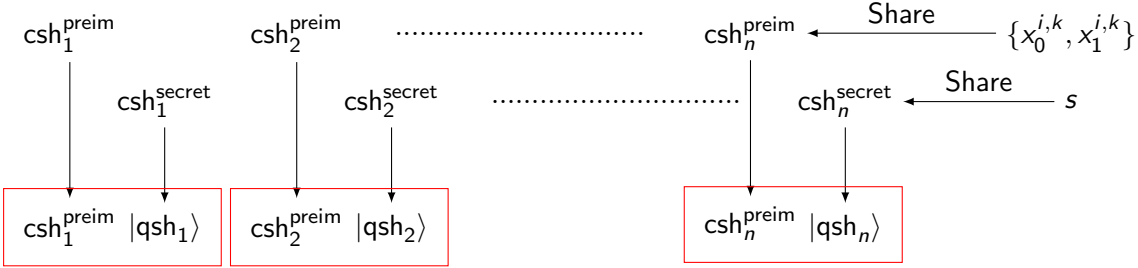
Secret sharing with PVD from OWFs—second attempt

- Idea: Give each party classical share of preimages in addition to quantum share.



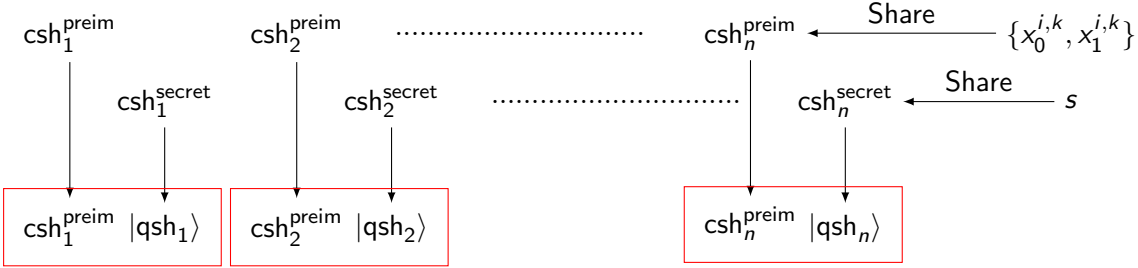
Secret sharing with PVD from OWFs—second attempt

- Idea: Give each party classical share of preimages in addition to quantum share.



Secret sharing with PVD from OWFs—second attempt

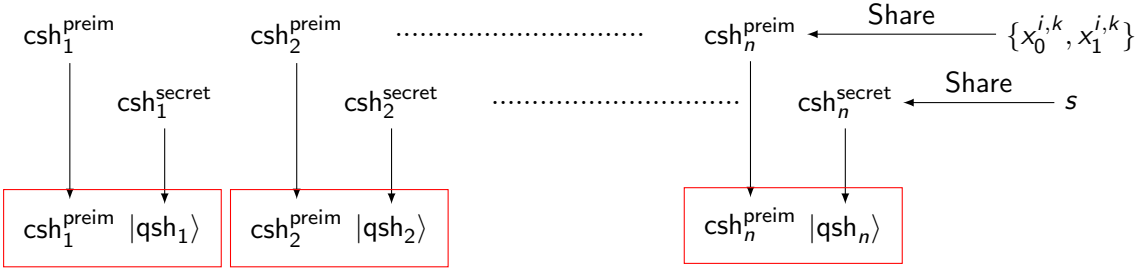
- Idea: Give each party classical share of preimages in addition to quantum share.



- Authorized set of parties can reconstruct preimages.

Secret sharing with PVD from OWFs—second attempt

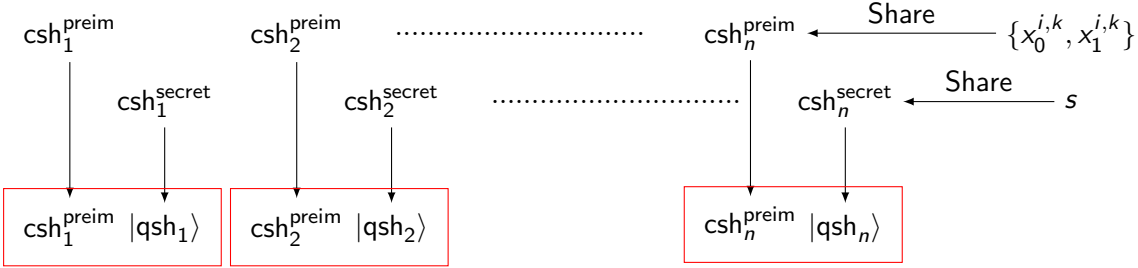
- Idea: Give each party classical share of preimages in addition to quantum share.



- Authorized set of parties can reconstruct preimages.
- Preimages are hidden from adversary holding an unauthorized set of shares.

Secret sharing with PVD from OWFs—second attempt

- Idea: Give each party classical share of preimages in addition to quantum share.



- Authorized set of parties can reconstruct preimages.
- Preimages are hidden from adversary holding an unauthorized set of shares.
- Problem: Classical part of each share does not get deleted. Adversary can accumulate an authorized set of classical preimage shares by adaptively corrupting shares

Example attack - (3,4)-secret sharing

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)} \left(\{ x_0^{i,k} \oplus x_1^{i,k} \} \right)$$

$$|\mathsf{qsh}_1\rangle \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \leftarrow \mathsf{Share}_{(3,4)} (s)$$

Example attack - (3,4)-secret sharing

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_0^{i,k} \oplus x_1^{i,k}\}\right)$$

$$|\mathsf{qsh}_1\rangle \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \leftarrow \mathsf{Share}_{(3,4)}(s)$$

Example attack - (3,4)-secret sharing

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_0^{i,k} \oplus x_1^{i,k}\}\right)$$

$$|\mathsf{qsh}_1\rangle \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \leftarrow \mathsf{Share}_{(3,4)}(s)$$

Example attack - (3,4)-secret sharing

$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \quad \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_0^{i,k} \oplus x_1^{i,k}\}\right)$

$\cancel{|\mathsf{qsh}_1\rangle} \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \quad \leftarrow \mathsf{Share}_{(3,4)}(s)$

cert_1

Example attack - (3,4)-secret sharing

$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \quad \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_0^{i,k} \oplus x_1^{i,k}\}\right)$

$\cancel{|\mathsf{qsh}_1\rangle} \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \quad \leftarrow \mathsf{Share}_{(3,4)}(s)$

cert_1

Example attack - (3,4)-secret sharing

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_0^{i,k} \oplus x_1^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1\rangle} \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \leftarrow \mathsf{Share}_{(3,4)}(s)$$

cert_1

Example attack - (3,4)-secret sharing

$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \quad \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_0^{i,k} \oplus x_1^{i,k}\}\right)$

$\cancel{|\mathsf{qsh}_1\rangle} \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \quad \leftarrow \mathsf{Share}_{(3,4)}(s)$

$\mathsf{cert}_1 \quad \mathsf{cert}_2$

Example attack - (3,4)-secret sharing

$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \quad \leftarrow \mathsf{Share}_{(3,4)} \left(\{ x_0^{i,k} \oplus x_1^{i,k} \} \right)$

$\cancel{|\mathsf{qsh}_1\rangle} \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \quad \leftarrow \mathsf{Share}_{(3,4)} (s)$

$\mathsf{cert}_1 \quad \mathsf{cert}_2$

Example attack - (3,4)-secret sharing

$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \quad \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_0^{i,k} \oplus x_1^{i,k}\}\right)$

$\cancel{|\mathsf{qsh}_1\rangle} \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \quad \leftarrow \mathsf{Share}_{(3,4)}(s)$

$\mathsf{cert}_1 \quad \mathsf{cert}_2$

Example attack - (3,4)-secret sharing

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_0^{i,k} \oplus x_1^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1\rangle} \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \leftarrow \mathsf{Share}_{(3,4)}(s)$$

$$\mathsf{cert}_1 \quad \mathsf{cert}_2$$

- Observation: At least one quantum share must be deleted properly.

Example attack - (3,4)-secret sharing

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \quad \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_0^{i,k} \oplus x_1^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1\rangle} \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \quad \leftarrow \mathsf{Share}_{(3,4)}(s)$$

$$\mathsf{cert}_1 \quad \mathsf{cert}_2$$

- Observation: At least one quantum share must be deleted properly.
- Scheme is secure if it is $(4, 4)$ instead of $(3, 4)$.

Example attack - (3,4)-secret sharing

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \quad \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_0^{i,k} \oplus x_1^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1\rangle} \quad |\mathsf{qsh}_2\rangle \quad |\mathsf{qsh}_3\rangle \quad |\mathsf{qsh}_4\rangle \quad \leftarrow \mathsf{Share}_{(3,4)}(s)$$

$$\mathsf{cert}_1 \quad \mathsf{cert}_2$$

- Observation: At least one quantum share must be deleted properly.
- Scheme is secure if it is $(4, 4)$ instead of $(3, 4)$.
- Idea: Make (3,4)-schemes secure by adding another layer to construction.

Secure $(3,4)$-secret sharing scheme

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,2}^{i,k} \oplus x_{1,2}^{i,k}\}\right)$$

$$|\mathsf{qsh}_1^1\rangle \quad |\mathsf{qsh}_2^1\rangle \quad |\mathsf{qsh}_3^1\rangle \quad |\mathsf{qsh}_4^1\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,1}^{i,k} \oplus x_{1,1}^{i,k}\}\right)$$

$$|\mathsf{qsh}_1^0\rangle \quad |\mathsf{qsh}_2^0\rangle \quad |\mathsf{qsh}_3^0\rangle \quad |\mathsf{qsh}_4^0\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(s\right)$$

Secure $(3,4)$-secret sharing scheme

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,2}^{i,k} \oplus x_{1,2}^{i,k}\}\right)$$

$$|\mathsf{qsh}_1^1\rangle \quad |\mathsf{qsh}_2^1\rangle \quad |\mathsf{qsh}_3^1\rangle \quad |\mathsf{qsh}_4^1\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,1}^{i,k} \oplus x_{1,1}^{i,k}\}\right)$$

$$|\mathsf{qsh}_1^0\rangle \quad |\mathsf{qsh}_2^0\rangle \quad |\mathsf{qsh}_3^0\rangle \quad |\mathsf{qsh}_4^0\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(s\right)$$

Secure $(3, 4)$-secret sharing scheme

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,2}^{i,k} \oplus x_{1,2}^{i,k}\}\right)$$

$$|\mathsf{qsh}_1^1\rangle \quad |\mathsf{qsh}_2^1\rangle \quad |\mathsf{qsh}_3^1\rangle \quad |\mathsf{qsh}_4^1\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,1}^{i,k} \oplus x_{1,1}^{i,k}\}\right)$$

$$|\mathsf{qsh}_1^0\rangle \quad |\mathsf{qsh}_2^0\rangle \quad |\mathsf{qsh}_3^0\rangle \quad |\mathsf{qsh}_4^0\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(s\right)$$

Secure $(3,4)$-secret sharing scheme

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,2}^{i,k} \oplus x_{1,2}^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1^1\rangle} \quad |\mathsf{qsh}_2^1\rangle \quad |\mathsf{qsh}_3^1\rangle \quad |\mathsf{qsh}_4^1\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,1}^{i,k} \oplus x_{1,1}^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1^0\rangle} \quad |\mathsf{qsh}_2^0\rangle \quad |\mathsf{qsh}_3^0\rangle \quad |\mathsf{qsh}_4^0\rangle \leftarrow \mathsf{Share}_{(3,4)}(s)$$

cert_1

Secure $(3,4)$-secret sharing scheme

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,2}^{i,k} \oplus x_{1,2}^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1^1\rangle} \quad |\mathsf{qsh}_2^1\rangle \quad |\mathsf{qsh}_3^1\rangle \quad |\mathsf{qsh}_4^1\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,1}^{i,k} \oplus x_{1,1}^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1^0\rangle} \quad |\mathsf{qsh}_2^0\rangle \quad |\mathsf{qsh}_3^0\rangle \quad |\mathsf{qsh}_4^0\rangle \leftarrow \mathsf{Share}_{(3,4)}(s)$$

cert_1

Secure $(3,4)$-secret sharing scheme

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,2}^{i,k} \oplus x_{1,2}^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1^1\rangle} \quad |\mathsf{qsh}_2^1\rangle \quad |\mathsf{qsh}_3^1\rangle \quad |\mathsf{qsh}_4^1\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,1}^{i,k} \oplus x_{1,1}^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1^0\rangle} \quad |\mathsf{qsh}_2^0\rangle \quad |\mathsf{qsh}_3^0\rangle \quad |\mathsf{qsh}_4^0\rangle \leftarrow \mathsf{Share}_{(3,4)}(s)$$

cert_1

Secure $(3,4)$-secret sharing scheme

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,2}^{i,k} \oplus x_{1,2}^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1^1\rangle} \quad |\mathsf{qsh}_2^1\rangle \quad |\mathsf{qsh}_3^1\rangle \quad |\mathsf{qsh}_4^1\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,1}^{i,k} \oplus x_{1,1}^{i,k}\}\right)$$

$$\underset{\mathsf{cert}_1}{\cancel{|\mathsf{qsh}_1^0\rangle}} \quad \underset{\mathsf{cert}_2}{\cancel{|\mathsf{qsh}_2^0\rangle}} \quad |\mathsf{qsh}_3^0\rangle \quad |\mathsf{qsh}_4^0\rangle \leftarrow \mathsf{Share}_{(3,4)}(s)$$

Secure $(3,4)$-secret sharing scheme

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,2}^{i,k} \oplus x_{1,2}^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1^1\rangle} \quad |\mathsf{qsh}_2^1\rangle \quad |\mathsf{qsh}_3^1\rangle \quad |\mathsf{qsh}_4^1\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,1}^{i,k} \oplus x_{1,1}^{i,k}\}\right)$$

$$\underset{\mathsf{cert}_1}{\cancel{|\mathsf{qsh}_1^0\rangle}} \quad \underset{\mathsf{cert}_2}{\cancel{|\mathsf{qsh}_2^0\rangle}} \quad |\mathsf{qsh}_3^0\rangle \quad |\mathsf{qsh}_4^0\rangle \leftarrow \mathsf{Share}_{(3,4)}(s)$$

Secure $(3,4)$-secret sharing scheme

$$\mathsf{csh}_1 \quad \mathsf{csh}_2 \quad \mathsf{csh}_3 \quad \mathsf{csh}_4 \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,2}^{i,k} \oplus x_{1,2}^{i,k}\}\right)$$

$$\cancel{|\mathsf{qsh}_1^1\rangle} \quad |\mathsf{qsh}_2^1\rangle \quad |\mathsf{qsh}_3^1\rangle \quad |\mathsf{qsh}_4^1\rangle \leftarrow \mathsf{Share}_{(3,4)}\left(\{x_{0,1}^{i,k} \oplus x_{1,1}^{i,k}\}\right)$$

$$\underset{\mathsf{cert}_1}{\cancel{|\mathsf{qsh}_1^0\rangle}} \quad \underset{\mathsf{cert}_2}{\cancel{|\mathsf{qsh}_2^0\rangle}} \quad |\mathsf{qsh}_3^0\rangle \quad |\mathsf{qsh}_4^0\rangle \leftarrow \mathsf{Share}_{(3,4)}(s)$$

Final construction from one-way functions

- $\mathsf{Share}_{\mathbb{A}}$: Classical secret sharing scheme for access structure $\mathbb{A}$.

$$
\begin{array}{cccc}
\mathsf{csh}_1 & \dots & \dots & \mathsf{csh}_n \\
|\mathsf{qsh}_1^{n-1}\rangle & \ddots & & |\mathsf{qsh}_n^{n-1}\rangle \\
\vdots & & \ddots & \vdots \\
|\mathsf{qsh}_1^{0}\rangle & \dots & \dots & |\mathsf{qsh}_n^{0}\rangle
\end{array}
\quad
\begin{array}{l}
\leftarrow \mathsf{Share}_{\mathbb{A}}\left(\{x_{0,n}^{i,k}, x_{1,n}^{i,k}\}\right) \\
\leftarrow \mathsf{Share}_{\mathbb{A}}\left(\{x_{0,n-1}^{i,k}, x_{1,n-1}^{i,k}\}\right) \\
\leftarrow \mathsf{Share}_{\mathbb{A}}\left(\{x_{0,1}^{i,k}, x_{1,1}^{i,k}\}\right) \\
\leftarrow \mathsf{Share}_{\mathbb{A}}(s)
\end{array}
$$

Final construction from one-way functions

- $\mathsf{Share}_{\mathbb{A}}$: Classical secret sharing scheme for access structure $\mathbb{A}$.

$$\begin{array}{ccccc}
\mathsf{csh}_1 & \dots & \dots & \mathsf{csh}_n & \leftarrow \mathsf{Share}_{\mathbb{A}}\left(\{x_{0,n}^{i,k}, x_{1,n}^{i,k}\}\right) \\
|\mathsf{qsh}_1^{n-1}\rangle & \ddots & & |\mathsf{qsh}_n^{n-1}\rangle & \leftarrow \mathsf{Share}_{\mathbb{A}}\left(\{x_{0,n-1}^{i,k}, x_{1,n-1}^{i,k}\}\right) \\
\vdots & & \ddots & \vdots & \leftarrow \mathsf{Share}_{\mathbb{A}}\left(\{x_{0,1}^{i,k}, x_{1,1}^{i,k}\}\right) \\
|\mathsf{qsh}_1^{0}\rangle & \dots & \dots & |\mathsf{qsh}_n^{0}\rangle & \leftarrow \mathsf{Share}_{\mathbb{A}}(s)
\end{array}$$

- Problem: The size of the secret being shared at each level is a multiple of the size of the shares in the previous level.

Final construction from one-way functions

- $\mathsf{Share}_{\mathbb{A}}$: Classical secret sharing scheme for access structure $\mathbb{A}$.

$$\begin{array}{cccc} \mathsf{csh}_1 & \dots & \dots & \mathsf{csh}_n \\ |\mathsf{qsh}_1^{n-1}\rangle & \ddots & & |\mathsf{qsh}_n^{n-1}\rangle \\ \vdots & & \ddots & \vdots \\ |\mathsf{qsh}_1^{0}\rangle & \dots & \dots & |\mathsf{qsh}_n^{0}\rangle \end{array} \quad \begin{array}{l} \leftarrow \mathsf{Share}_{\mathbb{A}}\left(\{x_{0,n}^{i,k}, x_{1,n}^{i,k}\}\right) \\ \leftarrow \mathsf{Share}_{\mathbb{A}}\left(\{x_{0,n-1}^{i,k}, x_{1,n-1}^{i,k}\}\right) \\ \leftarrow \mathsf{Share}_{\mathbb{A}}\left(\{x_{0,1}^{i,k}, x_{1,1}^{i,k}\}\right) \\ \leftarrow \mathsf{Share}_{\mathbb{A}}(s) \end{array}$$

- Problem: The size of the secret being shared at each level is a multiple of the size of the shares in the previous level.
- Solution: Use a PRF to generate the preimages, and share the PRF key at each level

Final construction from one-way functions

- $\mathsf{Share}_{\mathbb{A}}$: Classical secret sharing scheme for access structure $\mathbb{A}$.

csh_1	$\dots$	$\dots$	csh_n	$\leftarrow \mathsf{Share}_{\mathbb{A}}(k_n)$	
$\lvert\mathsf{qsh}_1^{n-1}\rangle$	$\ddots$		$\lvert\mathsf{qsh}_1^{n-1}\rangle$	$\leftarrow \mathsf{Share}_{\mathbb{A}}(k_{n-1})$	$\{x_{b,n-1}^{i,k}\} = \{\mathsf{PRF}(k_n, b\|\|i\|\|k)\}$
$\vdots$		$\ddots$	$\vdots$	$\leftarrow \mathsf{Share}_{\mathbb{A}}(k_1)$	$\{x_{b,1}^{i,k}\} = \{\mathsf{PRF}(k_2, b\|\|i\|\|k)\}$
$\lvert\mathsf{qsh}_1^{0}\rangle$	$\dots$	$\dots$	$\lvert\mathsf{qsh}_n^{0}\rangle$	$\leftarrow \mathsf{Share}_{\mathbb{A}}(s)$	$\{x_{b,0}^{i,k}\} = \{\mathsf{PRF}(k_1, b\|\|i\|\|k)\}$

- Problem: The size of the secret being shared at each level is a multiple of the size of the shares in the previous level.
- Solution: Use a PRF to generate the preimages, and share a seperate PRF key at each level.

Summary of first result

Theorem 1

There exists a secret sharing scheme satisfying **adaptive publicly verifiable deletion** *for any monotone access structure* $\mathbb{A}$ *using* $O(n \cdot m)$ *qubits per share assuming:*

- *OWFs*
- *classical (information-theoretic) secret sharing scheme for access structure* $\mathbb{A}$ *(with share size* m*)*

Secret sharing with smaller qubit complexity

- Construction from OWFs required $O(n \cdot (\text{classical share size}))$ qubits for n parties.

Secret sharing with smaller qubit complexity

- Construction from OWFs required $O(n \cdot (\text{classical share size}))$ qubits for n parties.
- Can we eliminate dependence on n?

Construction from compute-and-compare obfuscation

Starting point for construction:

- $\mathsf{csh}_1, \ldots, \mathsf{csh}_n \leftarrow \mathsf{Share}(s)$

$$|\mathsf{qsh}_i\rangle := \bigotimes_{k \in [m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]} |x_1^{i,k}\rangle \right)$$

Construction from compute-and-compare obfuscation

Starting point for construction:

- $\mathsf{csh}_1, \ldots, \mathsf{csh}_n \leftarrow \mathsf{Share}(s)$

$$|\mathsf{qsh}_i\rangle := \bigotimes_{k \in [m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]} |x_1^{i,k}\rangle \right)$$

- Problem:

Construction from compute-and-compare obfuscation

Starting point for construction:

- $\mathsf{csh}_1, \ldots, \mathsf{csh}_n \leftarrow \mathsf{Share}(s)$

$$|\mathsf{qsh}_i\rangle := \bigotimes_{k \in [m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]} |x_1^{i,k}\rangle \right)$$

- Problem:
  - Need to hide $x_0 \oplus x_1$ for valid deletion.

Construction from compute-and-compare obfuscation

Starting point for construction:

- $\mathsf{csh}_1, \ldots, \mathsf{csh}_n \leftarrow \mathsf{Share}(s)$

$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]} |x_1^{i,k}\rangle \right)$$

- Problem:
  - Need to hide $x_0 \oplus x_1$ for valid deletion.
  - Preimages are required for extraction/reconstruction.

Construction from compute-and-compare obfuscation

Starting point for construction:

- $\mathsf{csh}_1, ..., \mathsf{csh}_n \leftarrow \mathsf{Share}(s)$

$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]}|x_1^{i,k}\rangle\right)$$

- Problem:
  - Need to hide $x_0 \oplus x_1$ for valid deletion.
  - Preimages are required for extraction/reconstruction.
- Idea: Hide the the preimages inside a reconstruction oracle:

Construction from compute-and-compare obfuscation

Starting point for construction:

- $\mathsf{csh}_1, ..., \mathsf{csh}_n \leftarrow \mathsf{Share}(s)$

$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]}|x_1^{i,k}\rangle\right)$$

- Problem:
  - Need to hide $x_0 \oplus x_1$ for valid deletion.
  - Preimages are required for extraction/reconstruction.
- Idea: Hide the the preimages inside a reconstruction oracle:

Construction from compute-and-compare obfuscation

Starting point for construction:

- $\mathsf{csh}_1, ..., \mathsf{csh}_n \leftarrow \mathsf{Share}(s)$

$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]} |x_1^{i,k}\rangle \right)$$

- Problem:
  - Need to hide $x_0 \oplus x_1$ for valid deletion.
  - Preimages are required for extraction/reconstruction.
- Idea: Hide the the preimages inside a reconstruction oracle:

$\mathsf{Rec}(\{d_{i,k}\}_{i\in A})$

- Hardcode preimages $\{x_0^{i,k}, x_1^{i,k}\}_{i\in[n],k\in[m]}$

Construction from compute-and-compare obfuscation

Starting point for construction:

- $\mathsf{csh}_1, ..., \mathsf{csh}_n \leftarrow \mathsf{Share}(s)$

$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]}|x_1^{i,k}\rangle\right)$$

- Problem:
  - Need to hide $x_0 \oplus x_1$ for valid deletion.
  - Preimages are required for extraction/reconstruction.
- Idea: Hide the the preimages inside a reconstruction oracle:

$\mathsf{Rec}(\{d_{i,k}\}_{i\in A})$

- Hardcode preimages $\{x_0^{i,k}, x_1^{i,k}\}_{i\in[n],k\in[m]}$
- Compute $\mathsf{csh}'_i[k] = d_{i,k} \cdot (x_0^{i,k} \oplus x_1^{i,k})$. Set $\mathsf{csh}'_i = \mathsf{csh}'_i[1]...\mathsf{csh}'_i[m]$.

Construction from compute-and-compare obfuscation

Starting point for construction:

- $\mathsf{csh}_1, ..., \mathsf{csh}_n \leftarrow \mathsf{Share}(s)$

$$|\mathsf{qsh}_i\rangle := \bigotimes_{k\in[m]} \left(|x_0^{i,k}\rangle + (-1)^{\mathsf{csh}_i[k]} |x_1^{i,k}\rangle \right)$$

- Problem:
  - Need to hide $x_0 \oplus x_1$ for valid deletion.
  - Preimages are required for extraction/reconstruction.
- Idea: Hide the the preimages inside a reconstruction oracle:

$\mathsf{Rec}(\{d_{i,k}\}_{i\in A})$

- Hardcode preimages $\{x_0^{i,k}, x_1^{i,k}\}_{i\in[n],k\in[m]}$
- Compute $\mathsf{csh}'_i[k] = d_{i,k} \cdot (x_0^{i,k} \oplus x_1^{i,k})$. Set $\mathsf{csh}'_i = \mathsf{csh}'_i[1]...\mathsf{csh}'_i[m]$.
- Output $\mathsf{Reconstruct}(\{\mathsf{csh}'_i\}_{i\in A})$

Instantiating the reconstruction oracle

- Consider the compute-and-compare program:

$$\mathsf{CC}[P, \mathsf{lock}, z](x) := \begin{cases} z & P(x) = \mathsf{lock} \\ \perp & \text{otherwise.} \end{cases}$$

Instantiating the reconstruction oracle

- Consider the compute-and-compare program:

$$\mathsf{CC}[P, \mathsf{lock}, z](x) := \begin{cases} z & P(x) = \mathsf{lock} \\ \perp & \text{otherwise.} \end{cases}$$

Compute-and-compare obfuscator

- **Functionality**: $\widetilde{\mathsf{CC}} \leftarrow \mathsf{CC.Obf}(\mathsf{CC})$, where $\widetilde{\mathsf{CC}}(x) = \mathsf{CC}(x)$

Instantiating the reconstruction oracle

- Consider the compute-and-compare program:

$$\mathsf{CC}[P, \mathsf{lock}, z](x) := \begin{cases} z & P(x) = \mathsf{lock} \\ \perp & \text{otherwise.} \end{cases}$$

Compute-and-compare obfuscator

- **Functionality**: $\widetilde{\mathsf{CC}} \leftarrow \mathsf{CC.Obf}(\mathsf{CC})$, where $\widetilde{\mathsf{CC}}(x) = \mathsf{CC}(x)$
- **Security:** If lock is unpredictable given P, then $\widetilde{\mathsf{CC}}$ hides the details of $\mathsf{CC}[P, \mathsf{lock}, z]$

Instantiating the reconstruction oracle

- Consider the compute-and-compare program:

$$\mathsf{CC}[P, \mathsf{lock}, z](x) := \begin{cases} z & P(x) = \mathsf{lock} \\ \perp & \text{otherwise.} \end{cases}$$

Compute-and-compare obfuscator

- **Functionality**: $\widetilde{\mathsf{CC}} \leftarrow \mathsf{CC.Obf}(\mathsf{CC})$, where $\widetilde{\mathsf{CC}}(x) = \mathsf{CC}(x)$
- **Security:** If lock is unpredictable given P, then $\widetilde{\mathsf{CC}}$ hides the details of $\mathsf{CC}[P, \mathsf{lock}, z]$

- Can be constructed assuming LWE [WZ17]
- Idea: Set $P = \mathsf{Rec}$ and $\mathsf{lock} = z =$ secret

Secret sharing from compute-and-compare obfuscation

Share(s):

- Generate classical shares $\{\mathsf{csh}_i\}_{i\in[n]} \leftarrow \mathsf{Share}(s)$

Secret sharing from compute-and-compare obfuscation

Share(s):

- Generate classical shares $\{\mathsf{csh}_i\}_{i\in[n]} \leftarrow \mathsf{Share}(s)$
- Generate corresponding quantum shares $\{|\mathsf{qsh}_i\rangle\}_{i\in[n]} \leftarrow \{\mathsf{csh}_i\}_{i\in[n]}$.

Secret sharing from compute-and-compare obfuscation

Share(s):

- Generate classical shares $\{\mathsf{csh}_i\}_{i\in[n]} \leftarrow \mathsf{Share}(s)$
- Generate corresponding quantum shares $\{|\mathsf{qsh}_i\rangle\}_{i\in[n]} \leftarrow \{\mathsf{csh}_i\}_{i\in[n]}$.
- $\widetilde{\mathsf{Rec}} \leftarrow \mathsf{Obf}(\mathsf{CC}[\mathsf{Rec}, s, s])$

Secret sharing from compute-and-compare obfuscation

Share(s):

- Generate classical shares $\{\mathsf{csh}_i\}_{i\in[n]} \leftarrow \mathsf{Share}(s)$
- Generate corresponding quantum shares $\{|\mathsf{qsh}_i\rangle\}_{i\in[n]} \leftarrow \{\mathsf{csh}_i\}_{i\in[n]}$.
- $\widetilde{\mathsf{Rec}} \leftarrow \mathsf{Obf}(\mathsf{CC}[\mathsf{Rec}, s, s])$

Reconstruct$(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$

Secret sharing from compute-and-compare obfuscation

Share(s):

- Generate classical shares $\{\mathsf{csh}_i\}_{i\in[n]} \leftarrow \mathsf{Share}(s)$
- Generate corresponding quantum shares $\{|\mathsf{qsh}_i\rangle\}_{i\in[n]} \leftarrow \{\mathsf{csh}_i\}_{i\in[n]}$.
- $\widetilde{\mathsf{Rec}} \leftarrow \mathsf{Obf}(\mathsf{CC}[\mathsf{Rec}, s, s])$

Reconstruct$(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$

- Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}_{i\in A, k\in[m]}$

Secret sharing from compute-and-compare obfuscation

Share(s):

- Generate classical shares $\{\mathsf{csh}_i\}_{i\in[n]} \leftarrow \mathsf{Share}(s)$
- Generate corresponding quantum shares $\{|\mathsf{qsh}_i\rangle\}_{i\in[n]} \leftarrow \{\mathsf{csh}_i\}_{i\in[n]}$.
- $\widetilde{\mathsf{Rec}} \leftarrow \mathsf{Obf}(\mathsf{CC}[\mathsf{Rec}, s, s])$

Reconstruct$(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$

- Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}_{i\in A,k\in[m]}$
- Output $s \leftarrow \widetilde{\mathsf{Rec}}(\{d_{i,k}\}_{i\in A,k\in[m]})$

Secret sharing from compute-and-compare obfuscation

Share(s):

- Generate classical shares $\{\mathsf{csh}_i\}_{i\in[n]} \leftarrow \mathsf{Share}(s)$
- Generate corresponding quantum shares $\{|\mathsf{qsh}_i\rangle\}_{i\in[n]} \leftarrow \{\mathsf{csh}_i\}_{i\in[n]}$.
- $\widetilde{\mathsf{Rec}} \leftarrow \mathsf{Obf}(\mathsf{CC}[\mathsf{Rec}, s, s])$

Reconstruct$(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$

- Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}_{i\in A,k\in[m]}$
- Output $s \leftarrow \widetilde{\mathsf{Rec}}(\{d_{i,k}\}_{i\in A,k\in[m]})$

Security: Rec is independent of classical shares/lock. Therefore $\widetilde{\mathsf{Rec}}$ should hide preimages.

Secret sharing from compute-and-compare obfuscation

Share(s):

- Generate classical shares $\{\mathsf{csh}_i\}_{i\in[n]} \leftarrow \mathsf{Share}(s)$
- Generate corresponding quantum shares $\{|\mathsf{qsh}_i\rangle\}_{i\in[n]} \leftarrow \{\mathsf{csh}_i\}_{i\in[n]}$.
- $\widetilde{\mathsf{Rec}} \leftarrow \mathsf{Obf}(\mathsf{CC}[\mathsf{Rec}, s, s])$

Reconstruct$(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$

- Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}_{i\in A,k\in[m]}$
- Output $s \leftarrow \widetilde{\mathsf{Rec}}(\{d_{i,k}\}_{i\in A,k\in[m]})$

Security: Rec is independent of classical shares/lock. Therefore $\widetilde{\mathsf{Rec}}$ should hide preimages.

$\mathsf{Rec}(\{d_{i,k}\}_{i\in A})$

- Compute $\mathsf{csh}'_i[k] = d_{i,k} \cdot (x_0^{i,k} \oplus x_1^{i,k})$
- Set $\mathsf{csh}'_i = \mathsf{csh}'_i[1]...\mathsf{csh}'_i[m]$
- Output $\mathsf{Reconstruct}(\{\mathsf{csh}'_i\}_{i\in A})$

Secret sharing from compute-and-compare obfuscation

Share(s):

- Generate classical shares $\{\mathsf{csh}_i\}_{i\in[n]} \leftarrow \mathsf{Share}(s)$
- Generate corresponding quantum shares $\{|\mathsf{qsh}_i\rangle\}_{i\in[n]} \leftarrow \{\mathsf{csh}_i\}_{i\in[n]}$.
- $\widetilde{\mathsf{Rec}} \leftarrow \mathsf{Obf}(\mathsf{CC}[\mathsf{Rec}, s, s])$

Reconstruct$(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$

- Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}_{i\in A,k\in[m]}$
- Output $s \leftarrow \widetilde{\mathsf{Rec}}(\{d_{i,k}\}_{i\in A,k\in[m]})$

Security: Rec is independent of classical shares/lock. Therefore $\widetilde{\mathsf{Rec}}$ should hide preimages.

$\mathsf{Rec}(\{d_{i,k}\}_{i\in A})$

- Compute $\mathsf{csh}'_i[k] = d_{i,k} \cdot (x_0^{i,k} \oplus x_1^{i,k})$
- Set $\mathsf{csh}'_i = \mathsf{csh}'_i[1]...\mathsf{csh}'_i[m]$
- Output $\mathsf{Reconstruct}(\{\mathsf{csh}'_i\}_{i\in A})$

- Problem: The secret s is not unpredictable

Secret sharing from compute-and-compare obfuscation

Share(s):

- Generate classical shares $\{\mathsf{csh}_i\}_{i\in[n]} \leftarrow \mathsf{Share}(s)$
- Generate corresponding quantum shares $\{|\mathsf{qsh}_i\rangle\}_{i\in[n]} \leftarrow \{\mathsf{csh}_i\}_{i\in[n]}$.
- $\widetilde{\mathsf{Rec}} \leftarrow \mathsf{Obf}(\mathsf{CC}[\mathsf{Rec}, s, s])$

Reconstruct$(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$

- Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}_{i\in A,k\in[m]}$
- Output $s \leftarrow \widetilde{\mathsf{Rec}}(\{d_{i,k}\}_{i\in A,k\in[m]})$

Security: Rec is independent of classical shares/lock. Therefore $\widetilde{\mathsf{Rec}}$ should hide preimages.

Rec$(\{d_{i,k}\}_{i\in A})$

- Compute $\mathsf{csh}'_i[k] = d_{i,k} \cdot (x_0^{i,k} \oplus x_1^{i,k})$
- Set $\mathsf{csh}'_i = \mathsf{csh}'_i[1]...\mathsf{csh}'_i[m]$
- Output $\mathsf{Reconstruct}(\{\mathsf{csh}'_i\}_{i\in A})$

- Problem: The secret s is not unpredictable
- Solution: Use a uniform value independent of s as the lock

Secret sharing from compute-and-compare obfuscation

Share(s):

- Generate classical shares $\{\mathsf{csh}_i\}_{i\in[n]} \leftarrow$ ~~Share(s)~~ Share(lock) (lock is uniform)
- Generate corresponding quantum shares $\{|\mathsf{qsh}_i\rangle\}_{i\in[n]} \leftarrow \{\mathsf{csh}_i\}_{i\in[n]}$.
- ~~$\widetilde{\mathsf{Rec}} \leftarrow \mathsf{Obf}(\mathsf{CC}[\mathsf{Rec}, s, s])$~~ $\widetilde{\mathsf{Rec}} \leftarrow \mathsf{Obf}(\mathsf{CC}[\mathsf{Rec}, \mathsf{lock}, s])$

Reconstruct$(\{|\mathsf{qsh}_i\rangle\}_{i\in A})$

- Measure each $|\mathsf{qsh}_i\rangle$ in the Hadamard basis to obtain $\{d_{i,k}\}_{i\in A,k\in[m]}$.
- Output $s \leftarrow \widetilde{\mathsf{Rec}}(\{d_{i,k}\}_{i\in A,k\in[m]})$.

Security: Rec is independent of classical shares/lock. Therefore $\widetilde{\mathsf{Rec}}$ should hide preimages.

Rec$(\{d_{i,k}\}_{i\in A})$

- Compute $\mathsf{csh}'_i[k] = d_{i,k} \cdot (x_0^{i,k} \oplus x_1^{i,k})$.
- Set $\mathsf{csh}'_i = \mathsf{csh}'_i[1]...\mathsf{csh}'_i[m]$.
- Output Reconstruct$(\{\mathsf{csh}'_i\}_{i\in A})$

- Problem: The secret s is not unpredictable.
- Solution: Use a uniform value independent of s as the lock

Summary of second result

Theorem 2

There exists a secret sharing scheme with **adaptive publicly verifiable deletion** *for any monotone access structure* $\mathbb{A}$ *using* $O(m)$ *qubits per share assuming:*

- *LWE*
- *classical (information-theoretic) secret sharing scheme for access structure* $\mathbb{A}$

Open problems

- Secret sharing with PVD from weaker assumptions than OWF (e.g., from hard quantum planted problems [KNY23])?
- No-signaling security (with PVD) without sub-exponentially secure OWF?
- **Information-theoretic** secret sharing with adaptive certified deletion for any monotone access structure in the privately verifiable setting?

Thank You!