Fully Malicious Authenticated PIR

Marian Dietz
ETH Zürich*

Stefano Tessaro
University of Washington

* Work done while at University of Washington

Conventional PIR

$\mathbf{x} \in \{0,1\}^N$

Server

$i \in [N]$

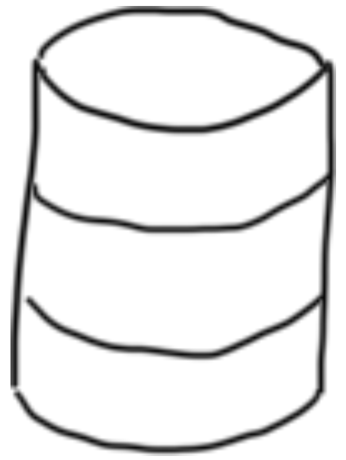
Client

wants: $\mathbf{x}_i$

[CGKS95]

Conventional PIR

$\mathbf{x} \in \{0,1\}^N$



Server

q ←

$q, st \leftarrow \texttt{Query}(i)$

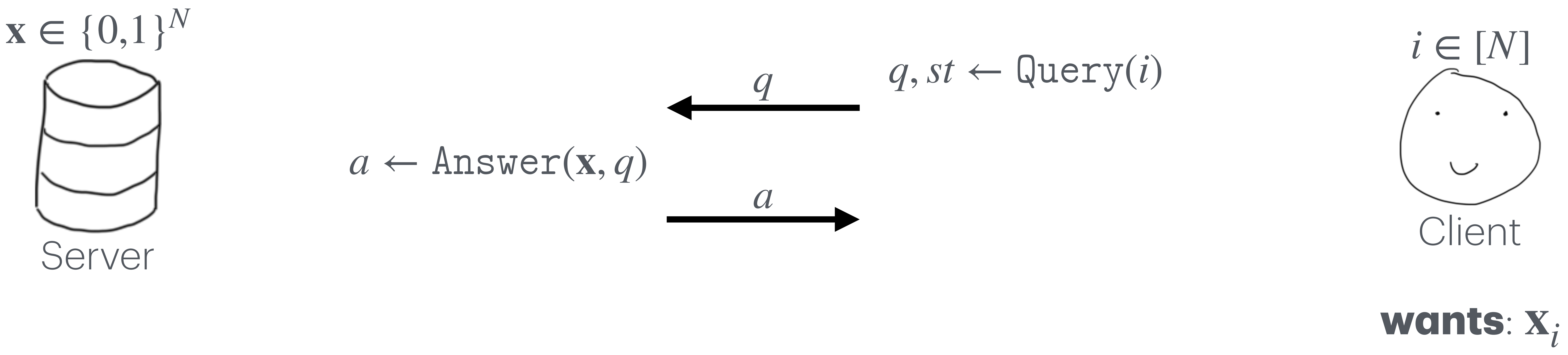
$i \in [N]$

Client

wants: $\mathbf{x}_i$

[CGKS95]

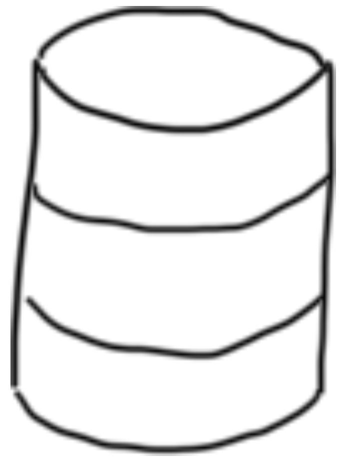
Conventional PIR



[CGKS95]

Conventional PIR

$\mathbf{x} \in \{0,1\}^N$



Server

q

$q, st \leftarrow \texttt{Query}(i)$

$a \leftarrow \texttt{Answer}(\mathbf{x}, q)$

a

$\mathbf{x}_i \leftarrow \texttt{Reconstruct}(st, a)$

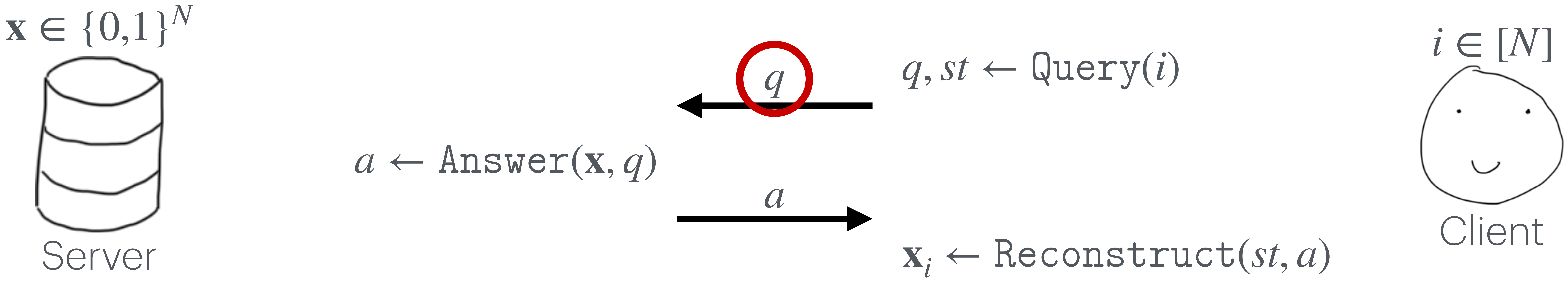
$i \in [N]$

Client

[CGKS95]

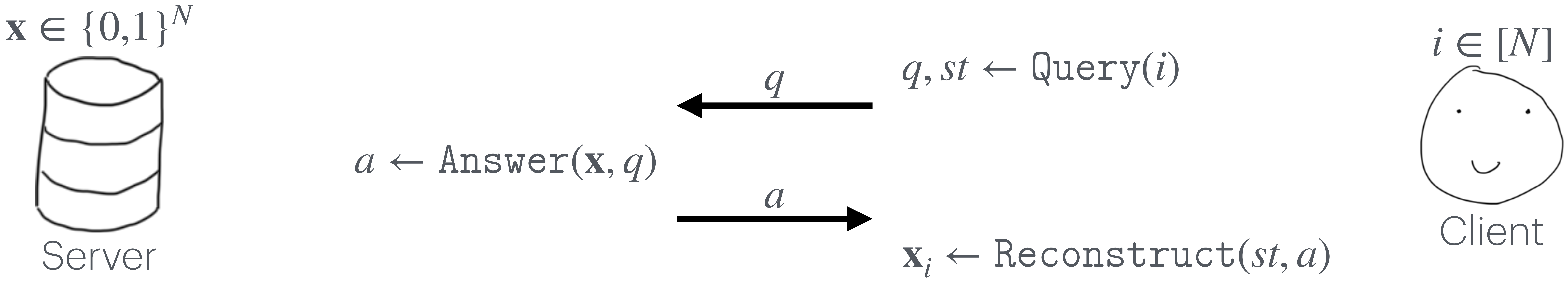
Conventional PIR

Privacy: q reveals nothing about i



[CGKS95]

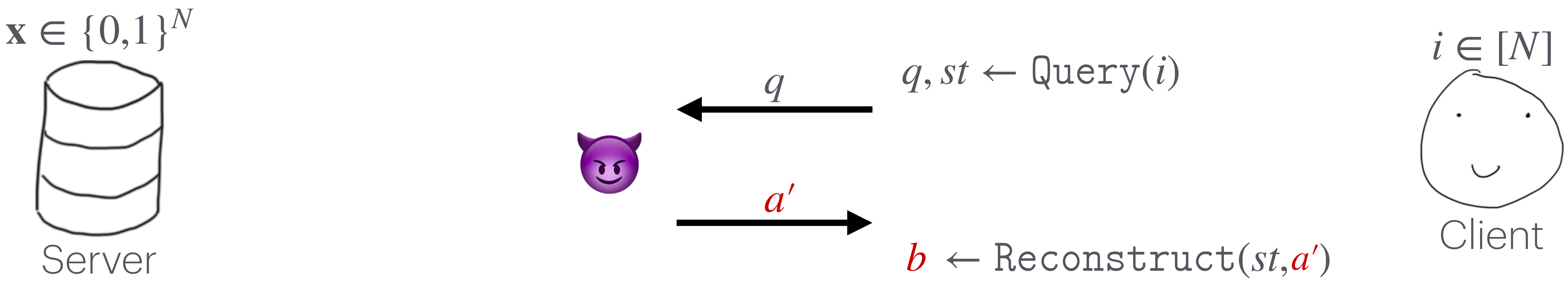
Conventional PIR



This has no integrity guarantees!

[CGKS95], [CNCWF23]

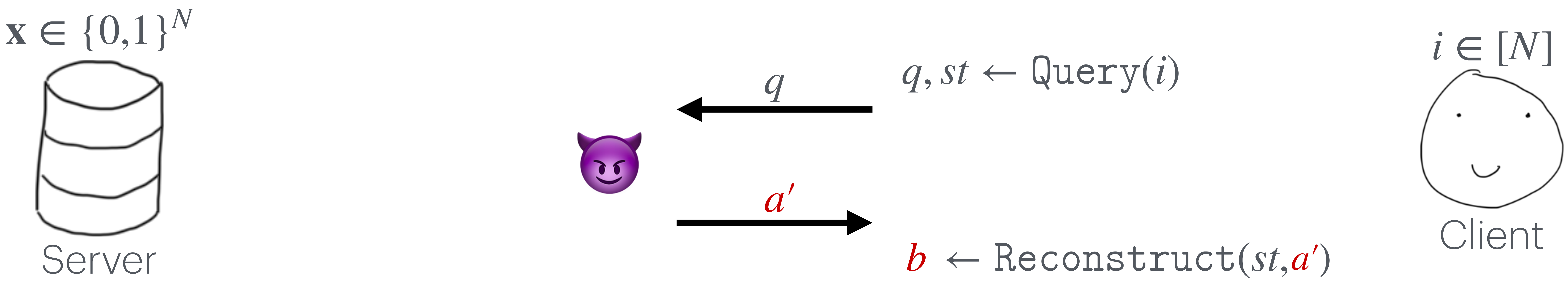
Conventional PIR



This has no integrity guarantees!

[CGKS95], [CNCWF23]

Conventional PIR

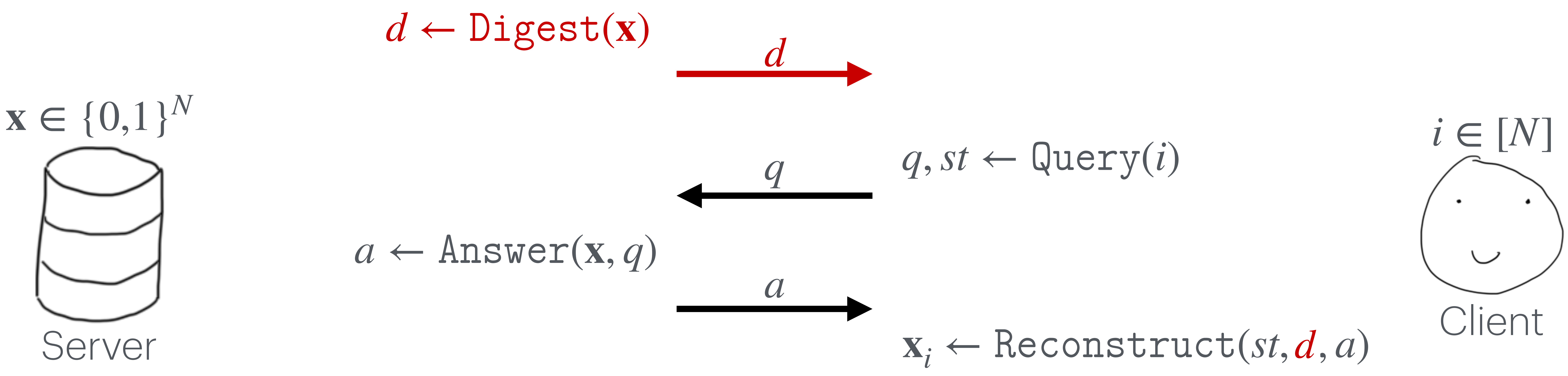


This has no integrity guarantees!

Example: If $\mathbf{x}$ is a public-key directory, server may inject arbitrary keys

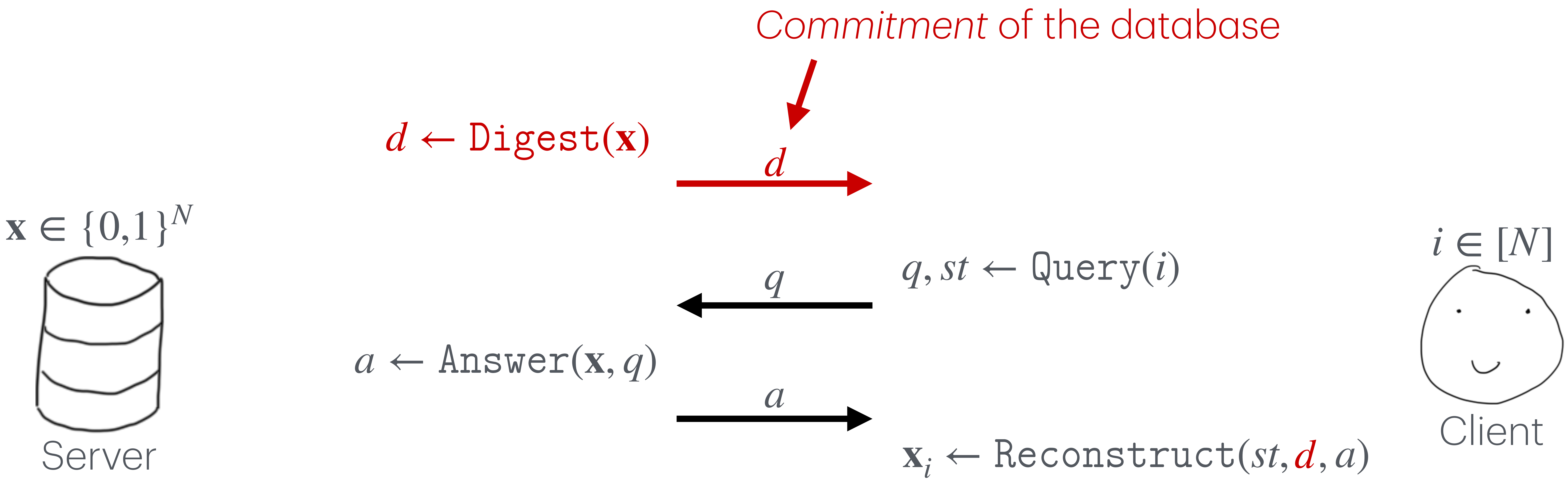
[CGKS95], [CNCWF23]

Authenticated PIR



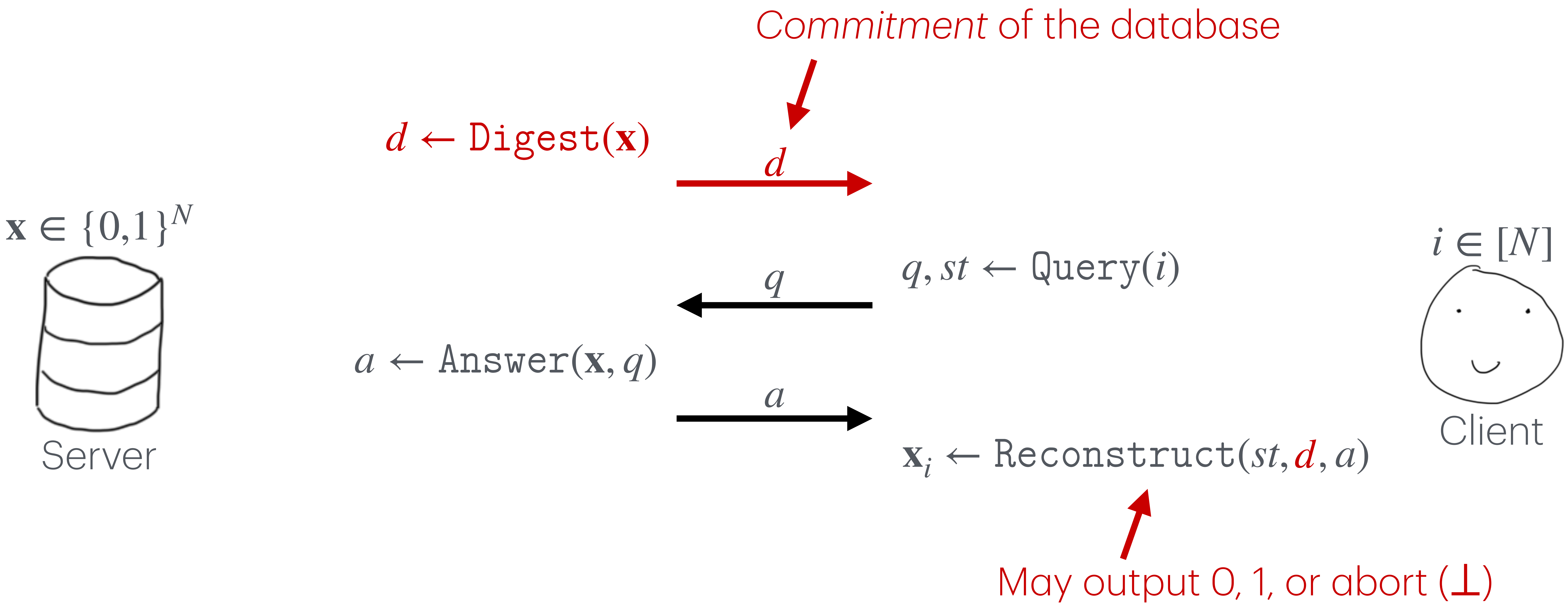
[CNCWF23]

Authenticated PIR



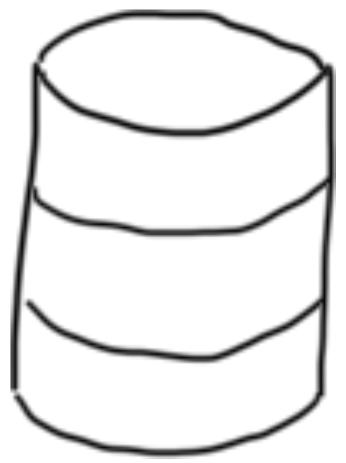
[CNCWF23]

Authenticated PIR



[CNCWF23]

Authenticated PIR



Server

Integrity game

d →

$q_0, st_0 \leftarrow \texttt{Query}(i)$
$q_1, st_1 \leftarrow \texttt{Query}(i)$

← q_0, q_1

a_0, a_1 →

$b_0 \leftarrow \texttt{Reconstruct}(st_0, d, a_0)$
$b_1 \leftarrow \texttt{Reconstruct}(st_1, d, a_1)$

$i \in [N]$
Challenger

Win if $b_0 = 0$ **&** $b_1 = 1$

[CNCWF23]

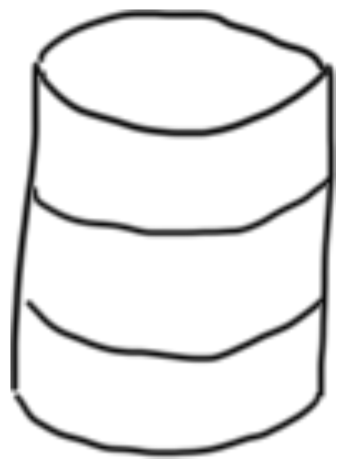
Authenticated PIR

(insufficient) **Privacy** game

😈

d →

q ← $q, st \leftarrow \texttt{Query}(i)$



Server

$i \in [N]$

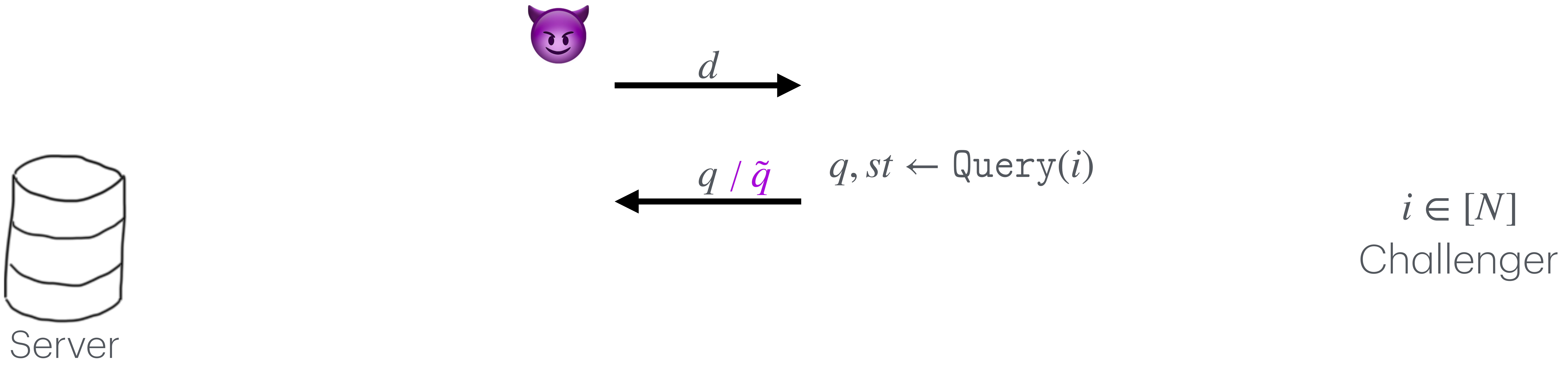
Challenger

[CNCWF23]

Authenticated PIR

Server must be able to simulate q.

(insufficient) **Privacy** game



[CNCWF23]

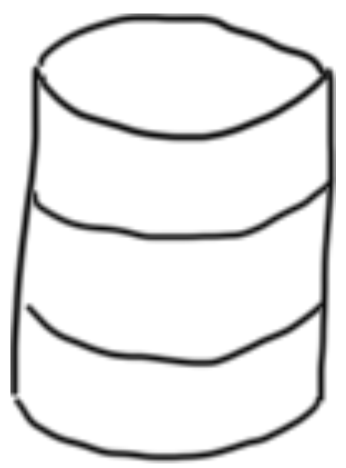
Authenticated PIR

(insufficient) **Privacy** game

😈

d →

q ← $q, st \leftarrow \texttt{Query}(i)$



Server

$i \in [N]$

Challenger

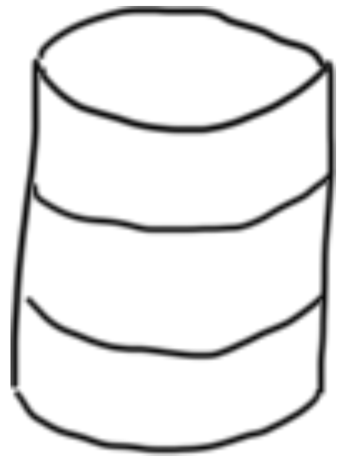
Vulnerable to selective-failure attacks:

[KO97]

[CNCWF23]

Authenticated PIR

(insufficient) **Privacy** game



Server

d →

$q, st \leftarrow \texttt{Query}(i)$

← q

$i \in [N]$
Challenger

Vulnerable to selective-failure attacks:

[KO97]

a →

$b \leftarrow \texttt{Reconstruct}(st, d, a)$

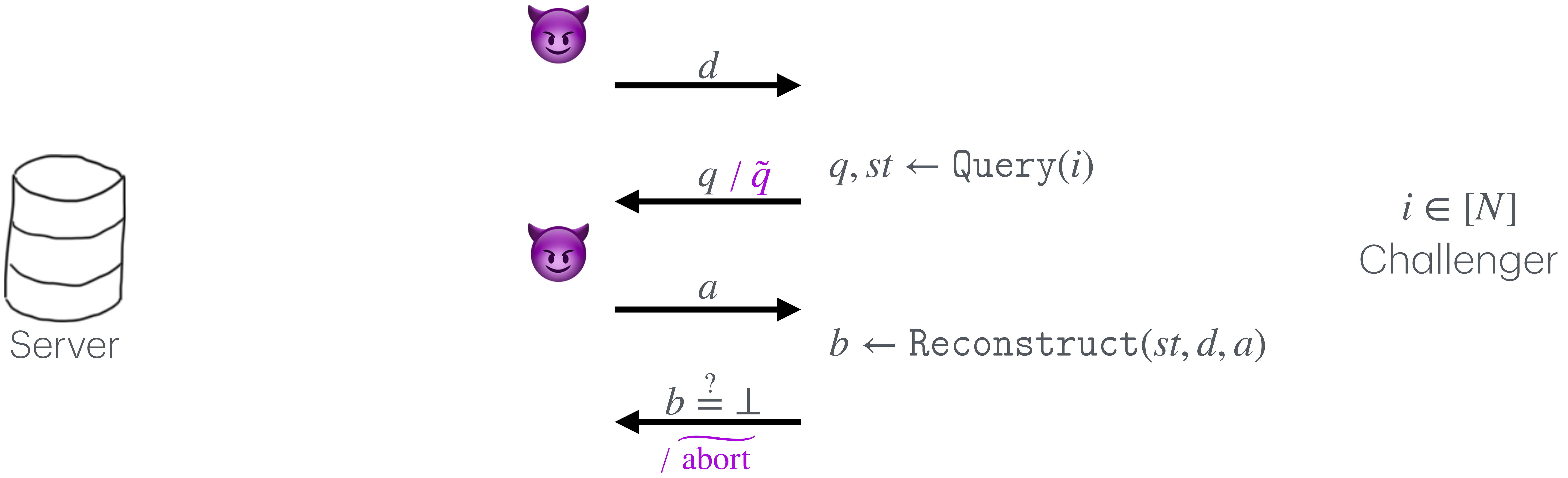
If $b = \bot$:

← "I received an error"

[CNCWF23]

Authenticated PIR

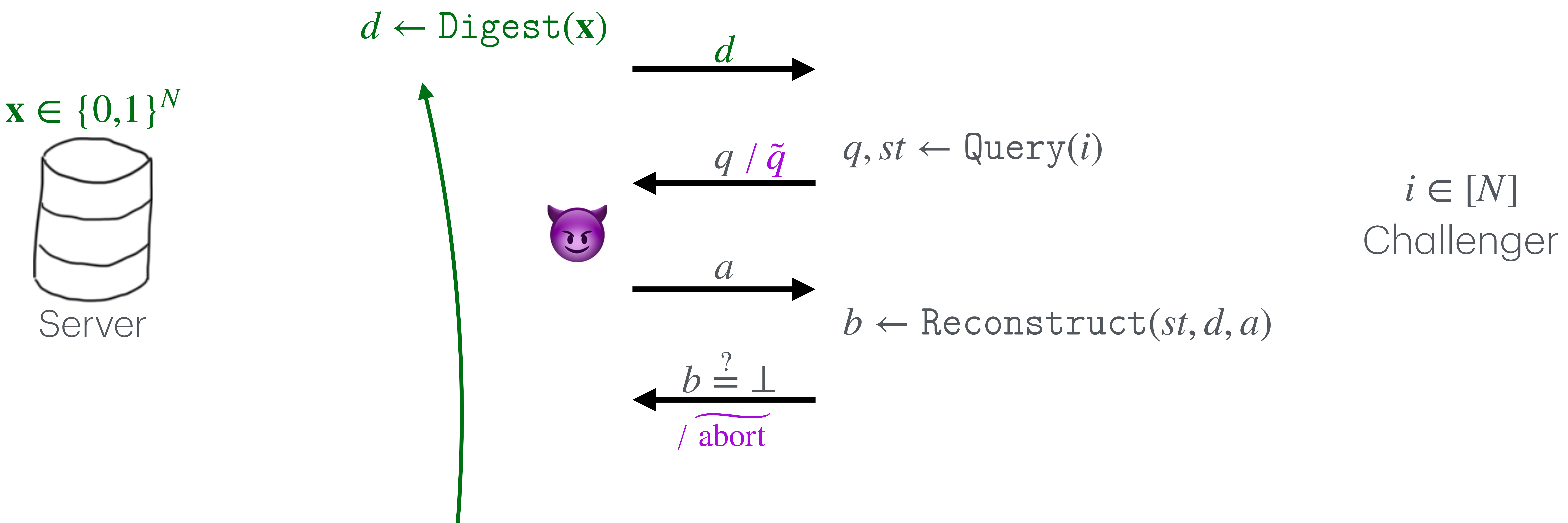
Privacy with abort game



[CNCWF23]

Authenticated PIR

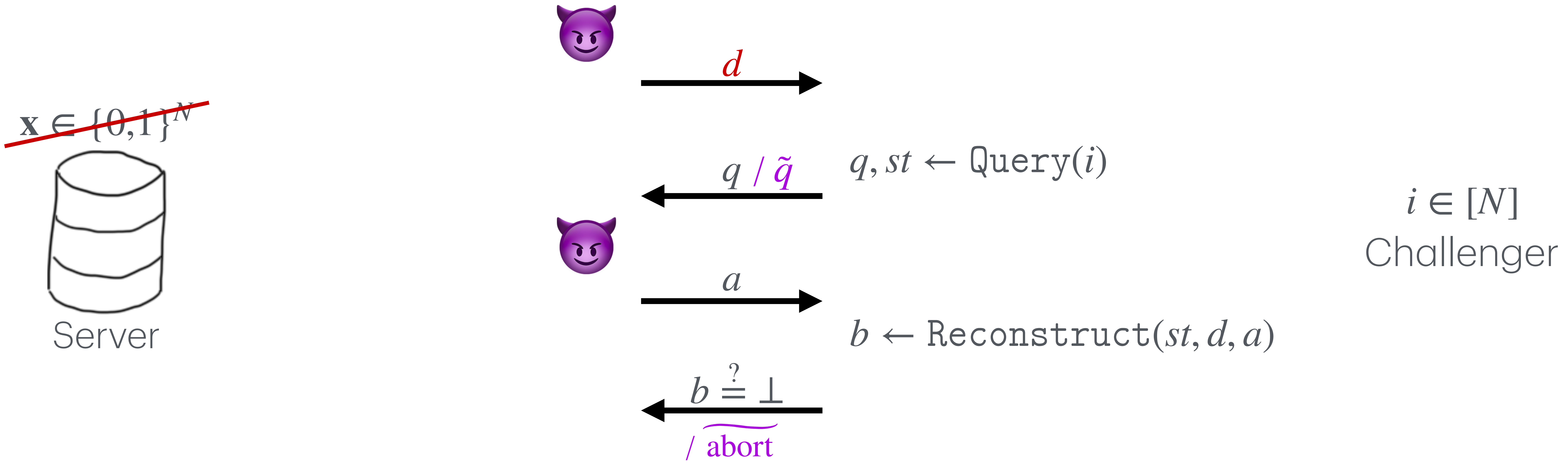
Privacy with abort game



Prior work [CNCWF23]:
Honest-digest assumption

Authenticated PIR

Privacy with abort game



Prior work [CNCWF23]:
Honest-digest assumption

This work:
Malicious-digest privacy

Authenticated PIR

Applications:

- Password breach database
- Certificate Transparency
- Streaming service

Prior work [CNCWF23]:
Honest-digest assumption

This work:
Malicious-digest privacy

Main Contributions

(1) Concrete attack
if malicious digests
are allowed

↓

Authenticated PIR [CNCWF23] with
Honest-digest assumption

Main Contributions

(1) Concrete attack
if malicious digests
are allowed

(2) Lightweight
“digest validation”

Authenticated PIR [CNCWF23] with
Honest-digest assumption

Authenticated PIR with
Malicious-digest privacy

Main Contributions

Everything is based on DDH!

(1) Concrete attack
if malicious digests
are allowed

(2) Lightweight
"digest validation"

Authenticated PIR [CNCWF23] with
Honest-digest assumption

Authenticated PIR with
Malicious-digest privacy

Baselines

Transforming [CNCWF23] into a malicious-digest version

Prove validity of digest d using:

Baselines

Transforming [CNCWF23] into a malicious-digest version

Prove validity of digest d using:

- SNARKs

Cannot do so with plain DDH [GW11]

Baselines

Transforming [CNCWF23] into a malicious-digest version

Prove validity of digest d using:

- SNARKs

  Cannot do so with plain DDH [GW11]

- *Interactive* protocols (e.g. Kilian's 4-round protocol [Kilian94])

  Requires non-black-box techniques

Baselines

Transforming [CNCWF23] into a malicious-digest version

Prove validity of digest d using:

- SNARKs

  Cannot do so with plain DDH [GW11]

- *Interactive* protocols (e.g. Kilian's 4-round protocol [Kilian94])

  Requires non-black-box techniques

- Bulletproof-like techniques [BBBPWM17]

  Linear verification time

Concurrent work: VeriSimplePIR [dCL24]

SimplePIR [HHCMV23] → VeriSimplePIR

SIS-based proofs

Concurrent work: VeriSimplePIR [dCL24]

SimplePIR [HHCMV23] → VeriSimplePIR

SIS-based proofs

- Lower computation in practice
- More client storage
- ROM

Honest-digest Authenticated PIR [CNCWF23]

$\mathbf{x} \in \{0,1\}^N$

Server

$i \in [N]$

Client

Honest-digest Authenticated PIR [CNCWF23]

Public: $\mathbf{h} = (\mathbf{h}_1, \ldots, \mathbf{h}_N) \in \mathbb{G}^N$

$\mathbf{x} \in \{0,1\}^N$

Server

$i \in [N]$

Client

Honest-digest Authenticated PIR [CNCWF23]

Public: $\mathbf{h} = (\mathbf{h}_1, \ldots, \mathbf{h}_N) \in \mathbb{G}^N$

$$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$$

$\xrightarrow{\quad d \quad}$

$\mathbf{x} \in \{0,1\}^N$

"non-hiding vector Pedersen commitment"

$i \in [N]$

Server

Client

Honest-digest Authenticated PIR [CNCWF23]

Public: $\mathbf{h} = (\mathbf{h}_1, \ldots, \mathbf{h}_N) \in \mathbb{G}^N$

$$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$$

$\xrightarrow{\quad d \quad}$

$\mathbf{x} \in \{0,1\}^N$

Server

sample $r \leftarrow \mathbb{Z}_q$, $\alpha \leftarrow \mathbb{Z}_q^*$

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

$\xleftarrow{\quad \mathbf{q} \quad}$

$i \in [N]$

Client

Honest-digest Authenticated PIR [CNCWF23]

Public: $\mathbf{h} = (\mathbf{h}_1, \ldots, \mathbf{h}_N) \in \mathbb{G}^N$

$\mathbf{x} \in \{0,1\}^N$

Server

$i \in [N]$

Client

$$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$$

d →

sample $r \leftarrow \mathbb{Z}_q$, $\alpha \leftarrow \mathbb{Z}_q^*$

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

← $\mathbf{q}$

$$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$$

a →

Honest-digest Authenticated PIR [CNCWF23]

Public: $\mathbf{h} = (\mathbf{h}_1, \ldots, \mathbf{h}_N) \in \mathbb{G}^N$

$\mathbf{x} \in \{0,1\}^N$

Server

$i \in [N]$

Client

$$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$$

d →

sample $r \leftarrow \mathbb{Z}_q$, $\alpha \leftarrow \mathbb{Z}_q^*$

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

← $\mathbf{q}$

$$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$$

a →

$$b := \begin{cases} 0 & \text{if } a = d^r \\ 1 & \text{if } a = d^r \cdot g^\alpha \\ \perp & \text{otherwise} \end{cases}$$

Honest-digest Authenticated PIR [CNCWF23]

$\mathbf{x} = (1,0,1)$

Server

$$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$$

$d = \mathbf{h}_1 \cdot \mathbf{h}_3$ →

$i = 1$

Client

$$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$$

← $\mathbf{q} = (\mathbf{h}_1^r \cdot g^\alpha, \mathbf{h}_2^r, \mathbf{h}_3^r)$

$$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$$

$a = d^r \cdot g^\alpha$ →

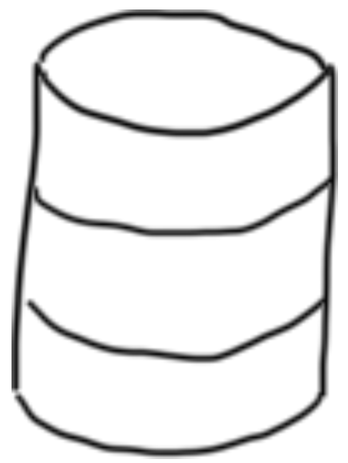
$$b := \begin{cases} 0 & \text{if } a = d^r \\ 1 & \text{if } a = d^r \cdot g^\alpha \\ \bot & \text{otherwise} \end{cases}$$

Honest-digest Authenticated PIR [CNCWF23]

Privacy with abort game

$$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$$

$\mathbf{x} \in \{0,1\}^N$



Server

$\xrightarrow{d}$

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

$i \in [N]$

Challenger

$\xleftarrow{\mathbf{q}}$

$\xrightarrow{a}$

$$b := \begin{cases} 0 & \text{if } a = d^r \\ 1 & \text{if } a = d^r \cdot g^\alpha \\ \bot & \text{otherwise} \end{cases}$$

$\xleftarrow{b \stackrel{?}{=} \bot}$

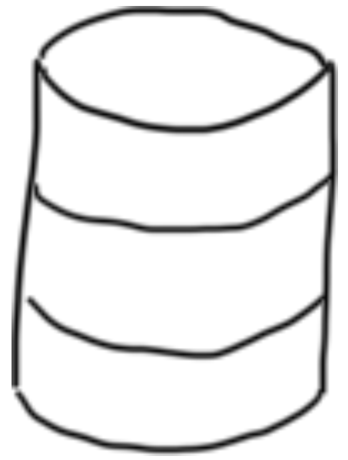
Simulation for
privacy with abort

Honest-digest Authenticated PIR [CNCWF23]

Privacy with abort game

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$

$\mathbf{x} \in \{0,1\}^N$



Server

d →

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

$i \in [N]$

Challenger

← $\mathbf{q}$

$\tilde{\mathbf{q}} \leftarrow \mathbb{G}^N$

😈

a →

$$b := \begin{cases} 0 & \text{if } a = d^r \\ 1 & \text{if } a = d^r \cdot g^\alpha \\ \perp & \text{otherwise} \end{cases}$$

← $b \stackrel{?}{=} \perp$

Simulation for *privacy with abort*

Honest-digest Authenticated PIR [CNCWF23]

Privacy with abort game

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$

$\mathbf{x} \in \{0,1\}^N$

Server

$\xrightarrow{d}$

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

$i \in [N]$

Challenger

$\xleftarrow{\mathbf{q}}$

$\tilde{\mathbf{q}} \leftarrow \mathbb{G}^N$

$\xrightarrow{a}$

$$b := \begin{cases} 0 & \text{if } a = d^r \\ 1 & \text{if } a = d^r \cdot g^\alpha \\ \perp & \text{otherwise} \end{cases}$$

$\xleftarrow{b \stackrel{?}{=} \perp}$

$\widetilde{\text{abort}} := \left[a \stackrel{?}{\neq} \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j} \right]$

Simulation for
privacy with abort

Honest-digest Authenticated PIR [CNCWF23]

Communication Complexity

- Digest: $O(1)$
- Query:
  $O(N)$ upload
  $O(1)$ download

After rebalancing

- Digest: $O(1)$
- Query:
  $O(\sqrt{N})$ upload
  $O(\sqrt{N})$ download

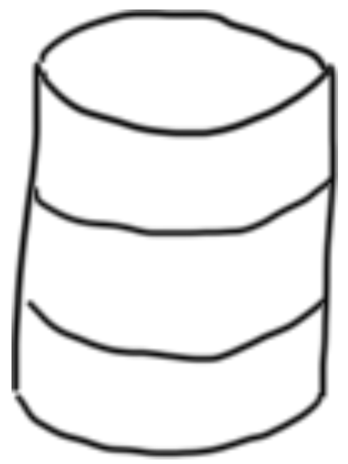
[CNCWF23], [KO97]

Is there an attack on [CNCWF23] when the digest is malicious?

Malicious-digest attack on [CNCWF23]

Privacy with abort game

$\mathbf{x} = (1,0,1)$



Server

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$

$d = \mathbf{h}_1 \cdot \mathbf{h}_3$ →

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

$i \in [N]$

Challenger

← $\mathbf{q}$

$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$

$a = d^r \cdot (g^\alpha)^{\mathbf{x}_i}$ →

$d^r \cdot g^\alpha$ if $i = 1$

d^r if $i = 2$

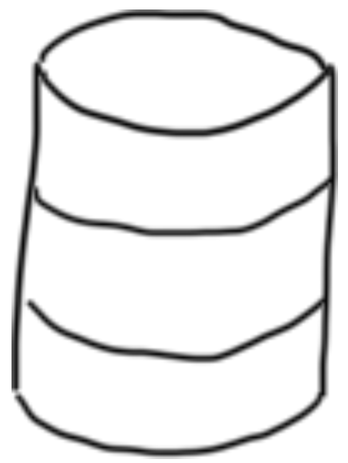
$d^r \cdot g^\alpha$ if $i = 3$

← $a \in \{d^r, d^r \cdot g^\alpha\}$

Malicious-digest attack on [CNCWF23]

Privacy with abort game

$\mathbf{x} = (1,0,1)$



Server

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$

$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$

These formulas would even allow non-binary $\mathbf{x}_i \notin \{0,1\}$

$d = \mathbf{h}_1 \cdot \mathbf{h}_3$ →

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

← $\mathbf{q}$

$a = d^r \cdot (g^\alpha)^{\mathbf{x}_i}$ →

- $d^r \cdot g^\alpha$ if $i = 1$
- d^r if $i = 2$
- $d^r \cdot g^\alpha$ if $i = 3$

← $a \in \{d^r, d^r \cdot g^\alpha\}$

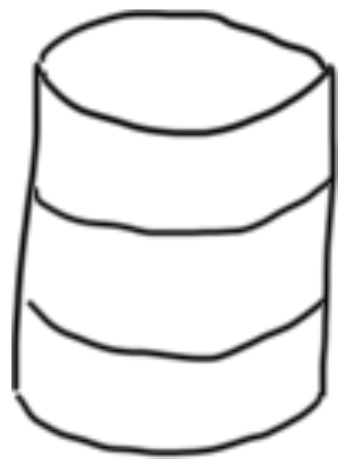
$i \in [N]$

Challenger

Malicious-digest attack on [CNCWF23]

Privacy with abort game

$\mathbf{x} = (2,0,1)$



Server

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$

$d = \mathbf{h}_1^{2} \cdot \mathbf{h}_3$ →

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

$i \in [N]$

Challenger

← $\mathbf{q}$

$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$

$a = d^r \cdot (g^\alpha)^{\mathbf{x}_i}$ →

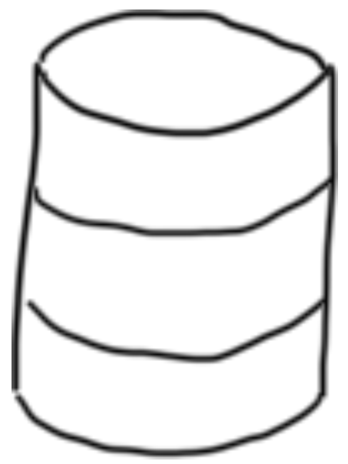
- $d^r \cdot g^{2\alpha}$ if $i = 1$
- d^r if $i = 2$
- $d^r \cdot g^\alpha$ if $i = 3$

← $a \in \{d^r, d^r \cdot g^\alpha\}$

Malicious-digest attack on [CNCWF23]

<u>**Privacy with abort** game</u>

$\mathbf{x} = (\mathbf{2}, 0, 1)$



Server

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$

$d = \mathbf{h}_1^{\mathbf{2}} \cdot \mathbf{h}_3$ →

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

$i \in [N]$

Challenger

← $\mathbf{q}$

$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$

$a = d^r \cdot (g^\alpha)^{\mathbf{x}_i}$ →

- $d^r \cdot g^{\mathbf{2}\alpha}$ if $i = 1$
- d^r if $i = 2$
- $d^r \cdot g^\alpha$ if $i = 3$

← $a \in \{d^r, d^r \cdot g^\alpha\} \Leftrightarrow i \in \{2,3\}$

Malicious-digest attack on [CNCWF23]

Privacy with abort game

$\mathbf{x} = (2,0,1)$

Server

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$ → $d = \mathbf{h}_1^2 \cdot \mathbf{h}_3$ →

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

$i \in [N]$

Challenger

← $\mathbf{q}$

$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$ → $a = d^r \cdot (g^\alpha)^{\mathbf{x}_i}$

- $d^r \cdot g^{2\alpha}$ if $i = 1$
- d^r if $i = 2$
- $d^r \cdot g^\alpha$ if $i = 3$

← $a \in \{d^r, d^r \cdot g^\alpha\} \Leftrightarrow i \in \{2,3\}$

Server learns whether a non-binary entry was queried!

Can we make [CNCWF23] secure against malicious-digests?

Validating the digest

$$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$$

d

$\mathbf{x} \in \mathbb{Z}_q^N$

Server

$i \in [N]$

Client

For now: "honest-digest assumption",
except that $\mathbf{x}$ may be non-binary

Validating the digest

$$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$$

d →

$\mathbf{x} \in \mathbb{Z}_q^N$

Server

$i \in [N]$

Client

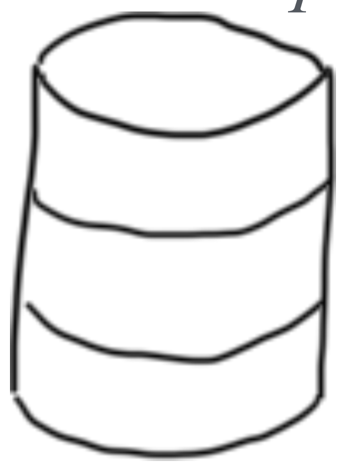
For now: "honest-digest assumption",
except that $\mathbf{x}$ may be non-binary

Goal: protocol to ensure that d was generated from a *binary* $\mathbf{x}$

Validating the digest

Query Protocol *(same as before)*

$\mathbf{x} \in \mathbb{Z}_q^N$



Server

$$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$$

d →

$$\mathbf{q} := \mathbf{h}^r \circ (g^{\alpha})^{\mathbf{e}_i}$$

← $\mathbf{q}$

$$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$$

a →

Expected: $(d^{-r} \cdot a)^{1/\alpha} = g^{\mathbf{x}_i}$

$i \in [N]$

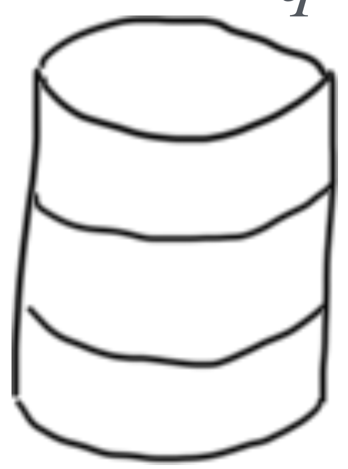
Client

Validating the digest

Query Protocol *(same as before)*

What happens if we "query a vector" that differs from the unit vector $\mathbf{e}_i$?

$\mathbf{x} \in \mathbb{Z}_q^N$



Server

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$

d →

$i \in [N]$

Client

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

← $\mathbf{q}$

$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$

a →

Expected: $(d^{-r} \cdot a)^{1/\alpha} = g^{\mathbf{x}_i}$

Validating the digest

<u>*Generalized* Query Protocol</u>

$\mathbf{x} \in \mathbb{Z}_q^N$

Server

$\mathbf{s} \in \mathbb{Z}_q^N$

Client

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$ $\xrightarrow{\quad d \quad}$

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{s}}$

$\xleftarrow{\quad \mathbf{q} \quad}$

$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$ $\xrightarrow{\quad a \quad}$

Expected: $(d^{-r} \cdot a)^{1/\alpha} = g^{\langle \mathbf{x}, \mathbf{s} \rangle}$

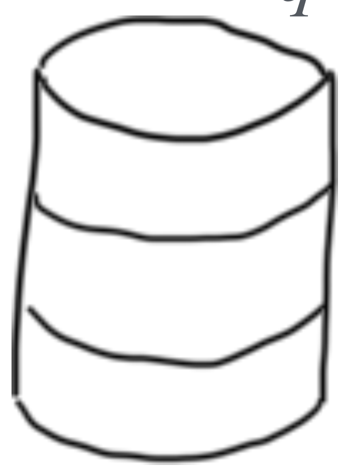
Validating the digest

<u>***Generalized* Query Protocol**</u>

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$

$\xrightarrow{\quad d \quad}$

$\mathbf{x} \in \mathbb{Z}_q^N$



Server

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{s}}$

$\mathbf{s} \in \mathbb{Z}_q^N$

Client

$\xleftarrow{\quad \mathbf{q} \quad}$

$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$

$\xrightarrow{\quad a \quad}$

Expected: $(d^{-r} \cdot a)^{1/\alpha} = g^{\langle \mathbf{x}, \mathbf{s} \rangle}$

Client can ask for arbitrary *inner products* of $\mathbf{x}$!

Validating the digest

<u>Inner Product Test</u>

$$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$$

$\mathbf{x} \in \mathbb{Z}_q^N$

Server

d →

$\mathbf{s} \leftarrow \{0,1\}^N$

$\mathbf{q} := \mathbf{h}^r \circ (g^{\alpha})^{\mathbf{s}}$

Client

← $\mathbf{q}$

$$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$$

a →

Expected: $(d^{-r} \cdot a)^{1/\alpha} = g^{\langle \mathbf{x}, \mathbf{s} \rangle}$

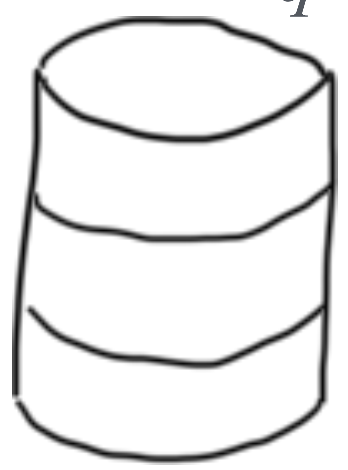
Validating the digest

Inner Product Test

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$

$\xrightarrow{\quad d \quad}$

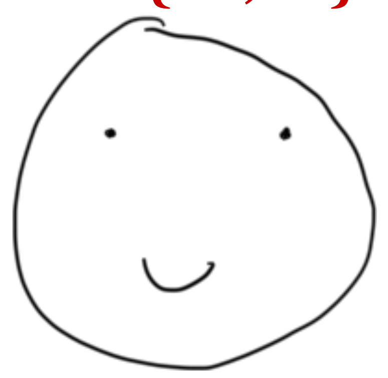
$\mathbf{x} \in \mathbb{Z}_q^N$



Server

$\mathbf{s} \leftarrow \{0,1\}^N$

$\mathbf{q} := \mathbf{h}^r \circ (g^{\alpha})^{\mathbf{s}}$



Client

$\xleftarrow{\quad \mathbf{q} \quad}$

$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$

$\xrightarrow{\quad a \quad}$

Expected: $(d^{-r} \cdot a)^{1/\alpha} = g^{\langle \mathbf{x}, \mathbf{s} \rangle}$

if $\mathbf{x}$ is binary, output is in $\{1, g^1, \ldots, g^N\}$

Validating the digest

Inner Product Test

$\mathbf{x} \in \mathbb{Z}_q^N$

Server

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$ → d

$\mathbf{s} \leftarrow \{0,1\}^N$

Client

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^\mathbf{s}$

← $\mathbf{q}$

$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$ → a

Expected: $(d^{-r} \cdot a)^{1/\alpha} = g^{\langle \mathbf{x}, \mathbf{s} \rangle}$

if $\mathbf{x}$ is binary, output is in $\{1, g^1, \ldots, g^N\}$

if $\mathbf{x}_j \notin \{-N, \ldots, N\}$, then w.p. $\geq \frac{1}{2}$, output is *not* in $\{1, g^1, \ldots, g^N\}$

Validating the digest

Validation

λ independent inner product tests

$\mathbf{x} \in \mathbb{Z}_q^N$

Server

$a^{(i)} = \prod_{j \in [N]} (\mathbf{q}_j^{(i)})^{\mathbf{x}_j}$

$\mathbf{q}^{(i)} := \mathbf{h}^{r^{(i)}} \circ (g^{\alpha^{(i)}})^{\mathbf{s}^{(i)}}$

$\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\lambda)}$ ←

$a^{(1)}, \ldots, a^{(\lambda)}$ →

Client

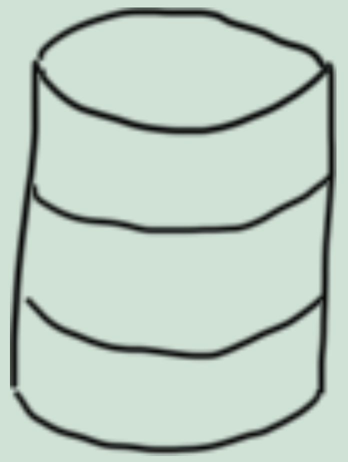
Pass if for all i: $(d^{-r^{(i)}} \cdot a^{(i)})^{1/\alpha^{(i)}} \in \{1, g^1, \ldots, g^N\}$

Validating the digest

Validation

λ independent inner product tests

$\mathbf{x} \in \mathbb{Z}_q^N$



Server

$\mathbf{q}^{(i)} := \mathbf{h}^{r^{(i)}} \circ (g^{\alpha^{(i)}})^{\mathbf{s}^{(i)}}$

$\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\lambda)}$ ←

$$a^{(i)} = \prod_{j \in [N]} (\mathbf{q}_j^{(i)})^{\mathbf{x}_j}$$

$a^{(1)}, \ldots, a^{(\lambda)}$ →

Client

Pass if for all i: $(d^{-r^{(i)}} \cdot a^{(i)})^{1/\alpha^{(i)}} \in \{1, g^1, \ldots, g^N\}$

if $\mathbf{x}_j \notin \{-N, \ldots, N\}$ for some j, then with probability $\geq 1 - \frac{1}{2^\lambda} -$ negl(λ), validation fails

Validating the digest

Validation

λ independent inner product tests

$\mathbf{x} \in \mathbb{Z}_q^N$

Server

$\mathbf{q}^{(i)} := \mathbf{h}^{r^{(i)}} \circ (g^{\alpha^{(i)}})^{\mathbf{s}^{(i)}}$

$\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\lambda)}$ ←

$$a^{(i)} = \prod_{j \in [N]} (\mathbf{q}_j^{(i)})^{\mathbf{x}_j}$$

$a^{(1)}, \ldots, a^{(\lambda)}$ →

Client

Pass if for all i: $(d^{-r^{(i)}} \cdot a^{(i)})^{1/\alpha^{(i)}} \in \{1, g^1, \ldots, g^N\}$

Communication cost:

$O(N \cdot \lambda)$ upload
$O(\lambda)$ download

rebalancing →

$O(\sqrt{N} \cdot \lambda)$ upload
$O(\sqrt{N} \cdot \lambda)$ download

Validating the digest

Modified **Query Protocol**

(Assuming that digest was validated successfully)

$\mathbf{x} \in \mathbb{Z}_q^N$

Server

$\mathbf{q} := \mathbf{h}^r \circ (g^\alpha)^{\mathbf{e}_i}$

$i \in [N]$

Client

$\mathbf{q}$ (sent from Client to Server)

$$a = \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j}$$

a (sent from Server to Client)

$$b := \begin{cases} 0 & \text{if } (d^{-r} \cdot a)^{1/\alpha} = 1 \\ 1 & \text{if } (d^{-r} \cdot a)^{1/\alpha} \in \{g^{-N}, \ldots, g^{-1}, g^{1}, \ldots, g^{N}\} \\ \perp & \text{otherwise} \end{cases}$$

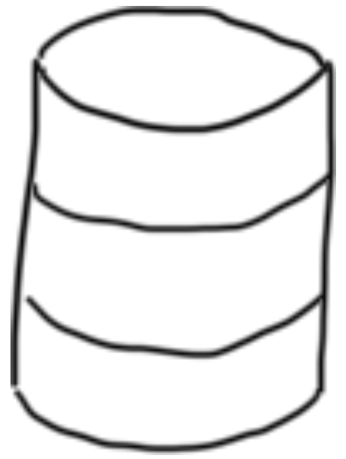
Necessary, since validation only ensures $\mathbf{x} \in \{-N, \ldots, N\}^N$!

Security Proof

Privacy with abort game

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$

$\mathbf{x} \in \mathbb{Z}_q^N$



Server

d →

successful validation

← $\mathbf{q}$

a →

← abort

$i \in [N]$

Challenger

Security Proof

Privacy with abort game

$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$

d

⋮

successful validation

⋮

$\mathbf{x} \in \mathbb{Z}_q^N$

Server

$i \in [N]$

Challenger

$\mathbf{q}$

$\widetilde{\mathbf{q}} \leftarrow \mathbb{G}^N$

a

abort

$\widetilde{\text{abort}} := \left[a \stackrel{?}{\neq} \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j} \right]$

Simulation for *privacy with abort*

Security Proof

Privacy with abort game

$$d = \prod_{j \in [N]} \mathbf{h}_j^{\mathbf{x}_j}$$

$\xrightarrow{\quad d \quad}$

successful validation

$\mathbf{x} \in \mathbb{Z}_q^N$

Server

$i \in [N]$

Challenger

$\xleftarrow{\quad \mathbf{q} \quad}$

$\widetilde{\mathbf{q}} \leftarrow \mathbb{G}^N$

$\xrightarrow{\quad a \quad}$

$\xleftarrow{\quad \text{abort} \quad}$

$$\widetilde{\text{abort}} := \left[a \overset{?}{\neq} \prod_{j \in [N]} \mathbf{q}_j^{\mathbf{x}_j} \right]$$

Simulation requires knowing database $\mathbf{x}$ that matches d

Simulation for *privacy with abort*

Security Proof

Privacy with abort game

d →

⋮

successful validation

⋮

$\mathbf{q}$ ←

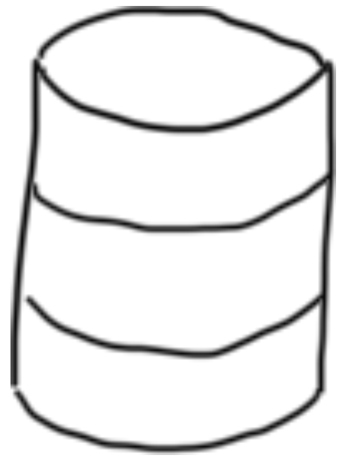
$\tilde{\mathbf{q}} \leftarrow \mathbb{G}^N$

a →

abort ←

???

~~$\mathbf{x} \in \mathbb{Z}_q^N$~~



Server

$i \in [N]$

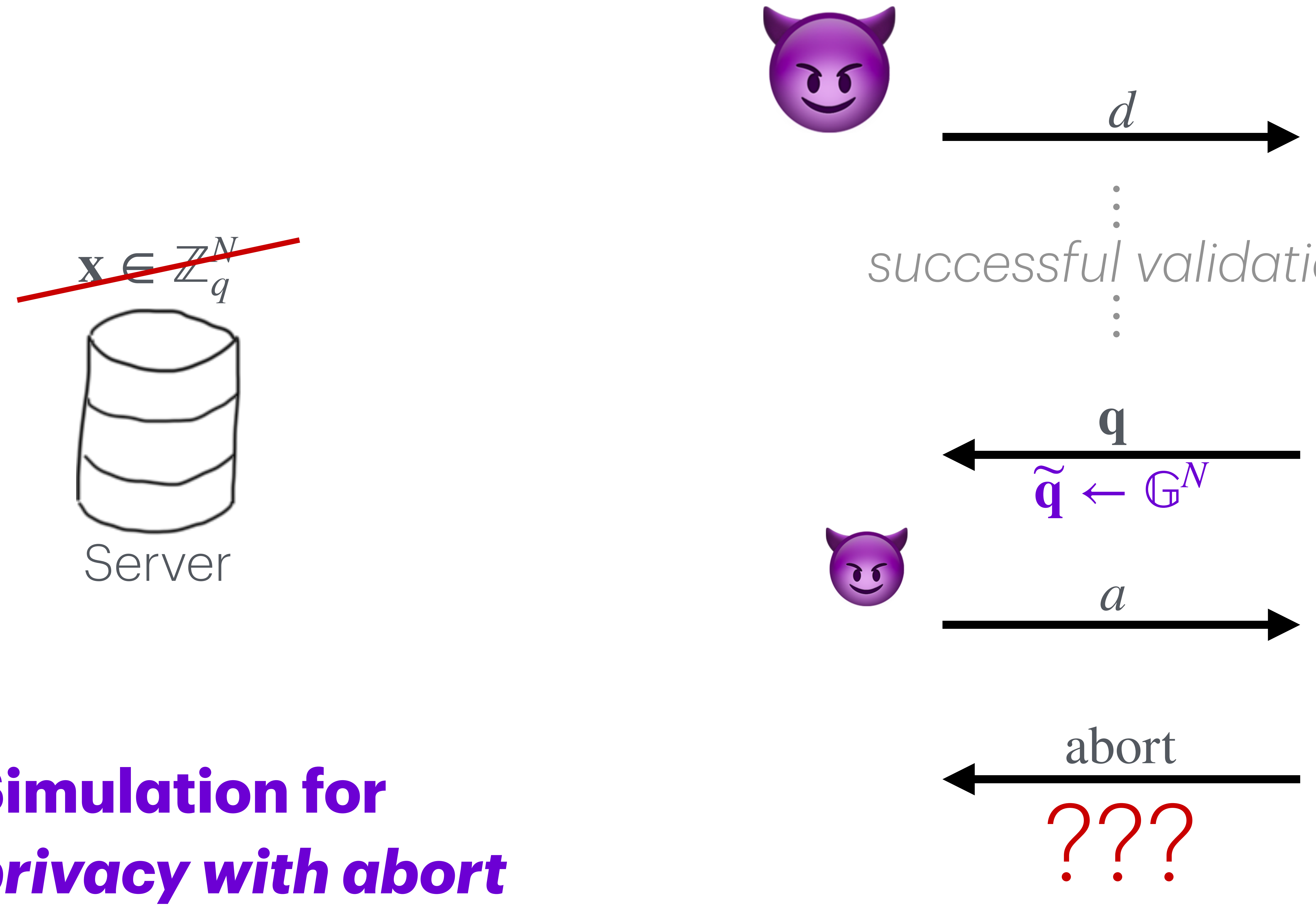
Challenger

Simulation for

privacy with abort

Security Proof

Privacy with abort game



$i \in [N]$
Challenger

As long as the adversary can find *some* answer a' that will *not* abort, we could simulate $\widetilde{\text{abort}} := \left[a \overset{?}{\neq} a'\right]$

Simulation for *privacy with abort*

Security Proof

Answer Extraction: the (malicious) server always has a way of answering any query without the client aborting

Security Proof

Answer Extraction: the (malicious) server always has a way of answering any query without the client aborting

Integrity

Standard Privacy

Privacy with abort

Security Proof

1-time successful validation step

"probability amplification"

Answer Extraction: the (malicious) server always has a way of answering any query without the client aborting

Integrity

Standard Privacy

Privacy with abort

Security Proof

1-time successful validation step

↓ *“probability amplification”*

Answer Extraction: the (malicious) server always has a way of answering any query without the client aborting

Non-trivial, because we still need a way of picking a “good” answer from a large pool of options!

Integrity → **Privacy with abort** ← **Standard Privacy**

Open Problems

- Adaptation towards lattice-based schemes
- Reduce overhead compared to plain PIR schemes

Open Problems

- Adaptation towards lattice-based schemes
- Reduce overhead compared to plain PIR schemes

Thank you!

ePrint: **2023/1804**