Holding Secrets Accountable: Auditing Private ML Algorithms

Hidde Lycklama
ETH zürich

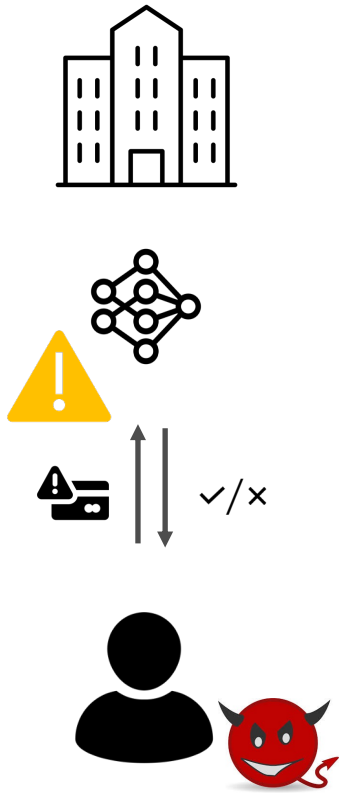
Alexander Viand
intel

Nicolas Küchler
ETH zürich

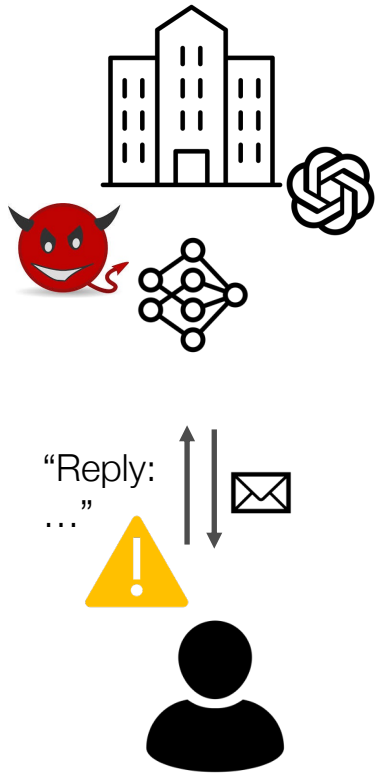
Christian Knabenhans
EPFL

Anwar Hithnawi
ETH zürich

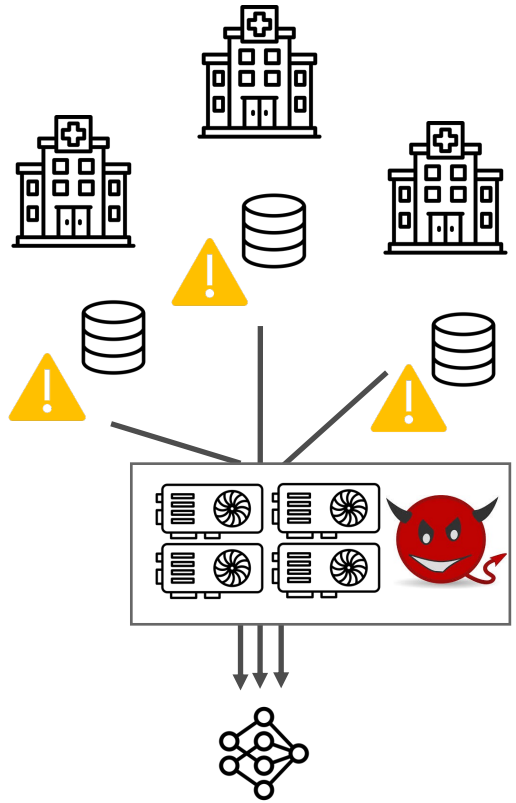
Privacy-Preserving Machine Learning (PPML)



Model Privacy

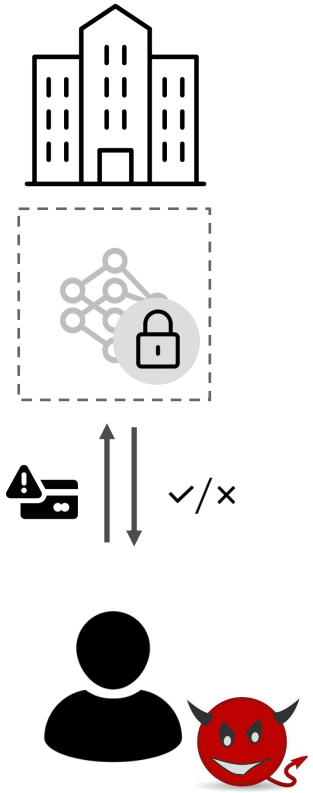


Query Privacy

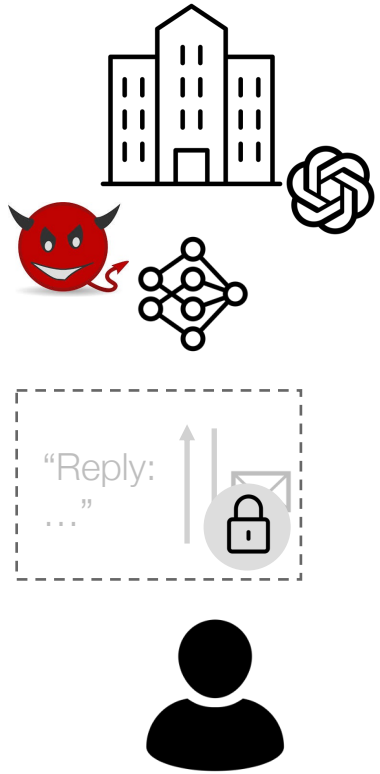


Data Privacy

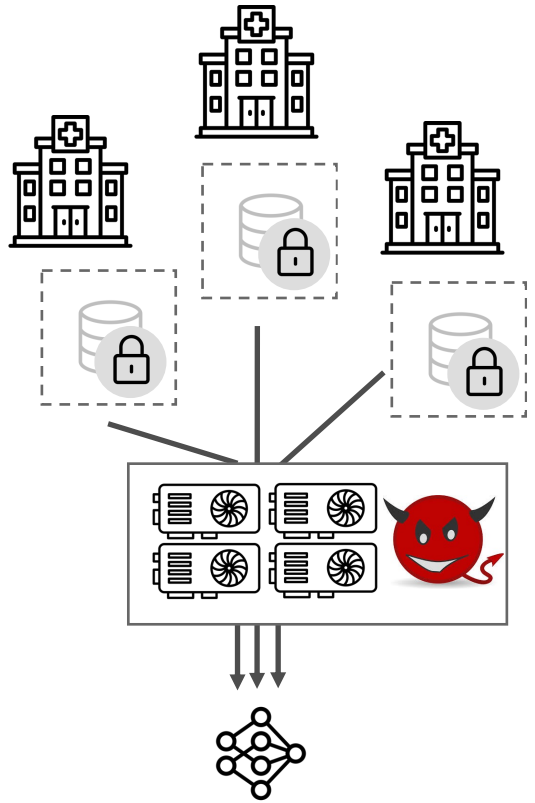
Privacy-Preserving Machine Learning (PPML)



Model Privacy



Query Privacy



Data Privacy

Privacy-Transparency Dichotomy

Privacy

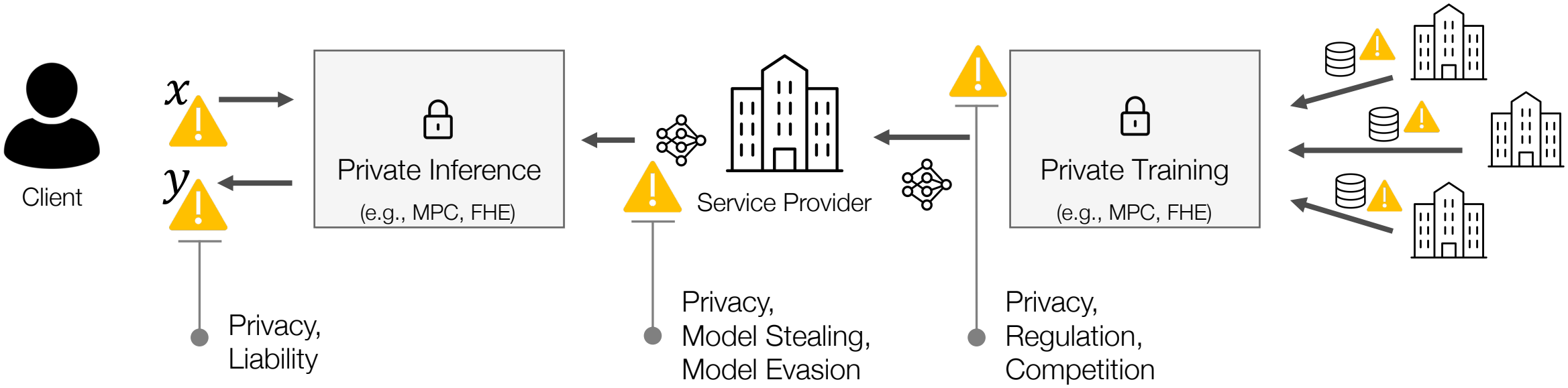
Transparency

Fairness

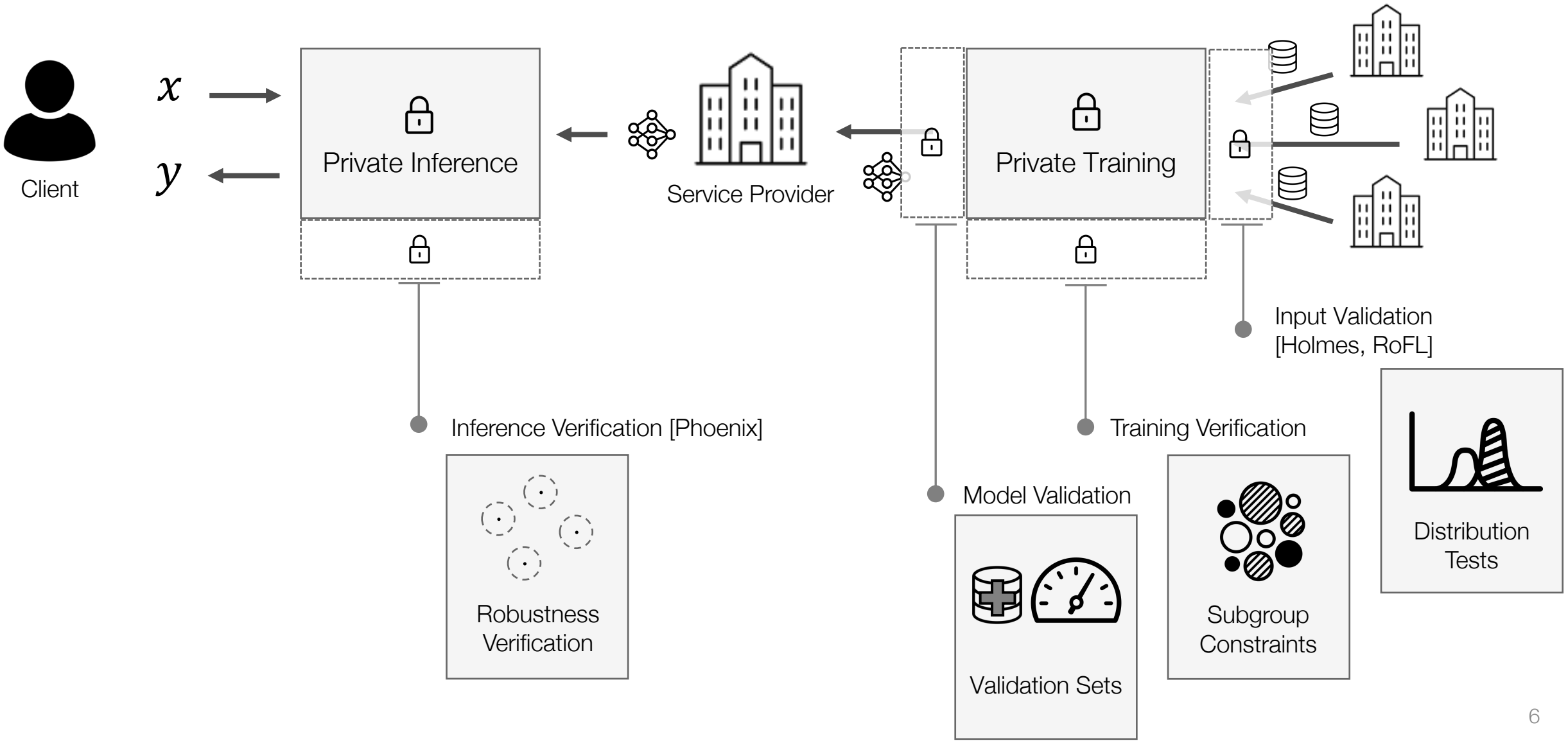
Accountability

Safety

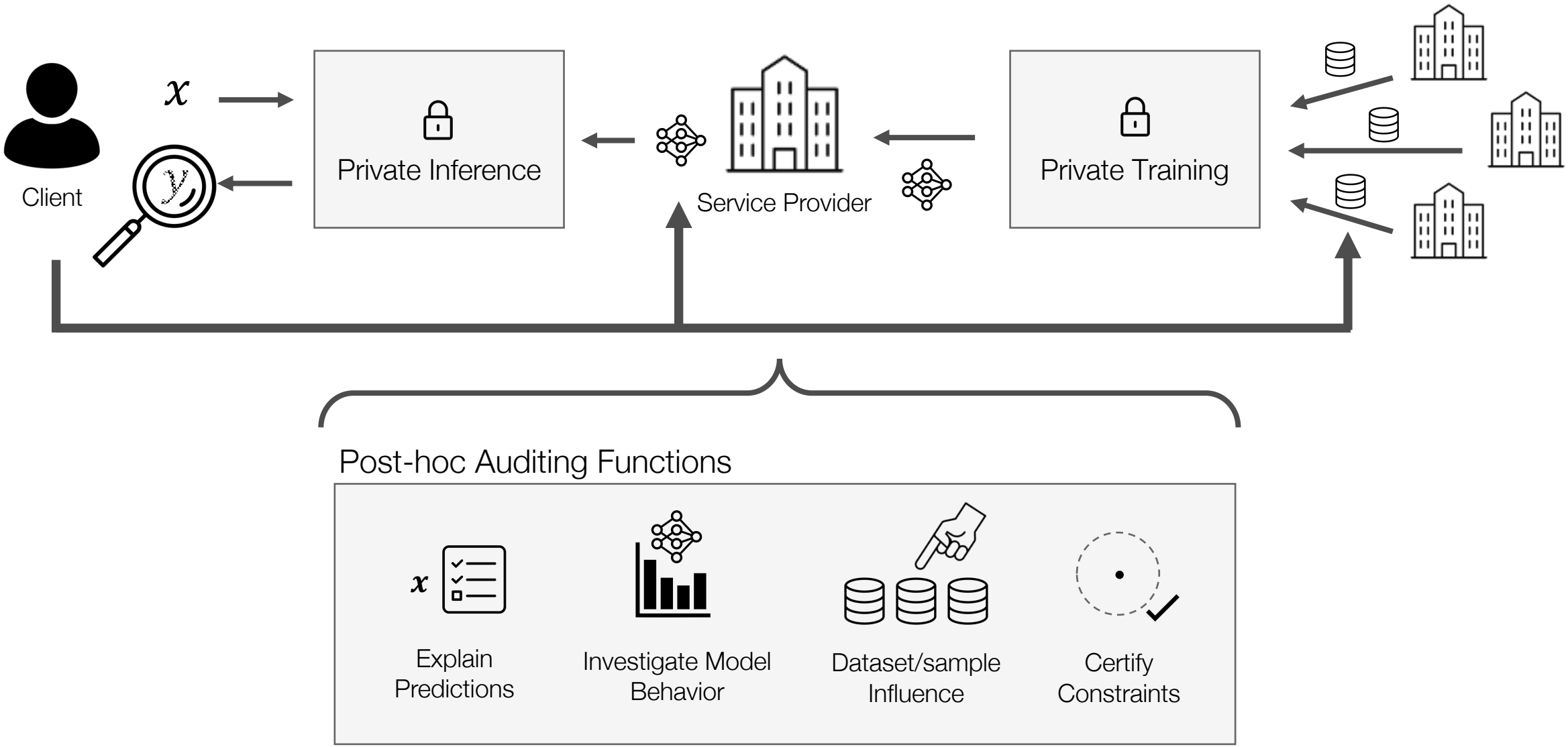
Privacy-Preserving Machine Learning (PPML)



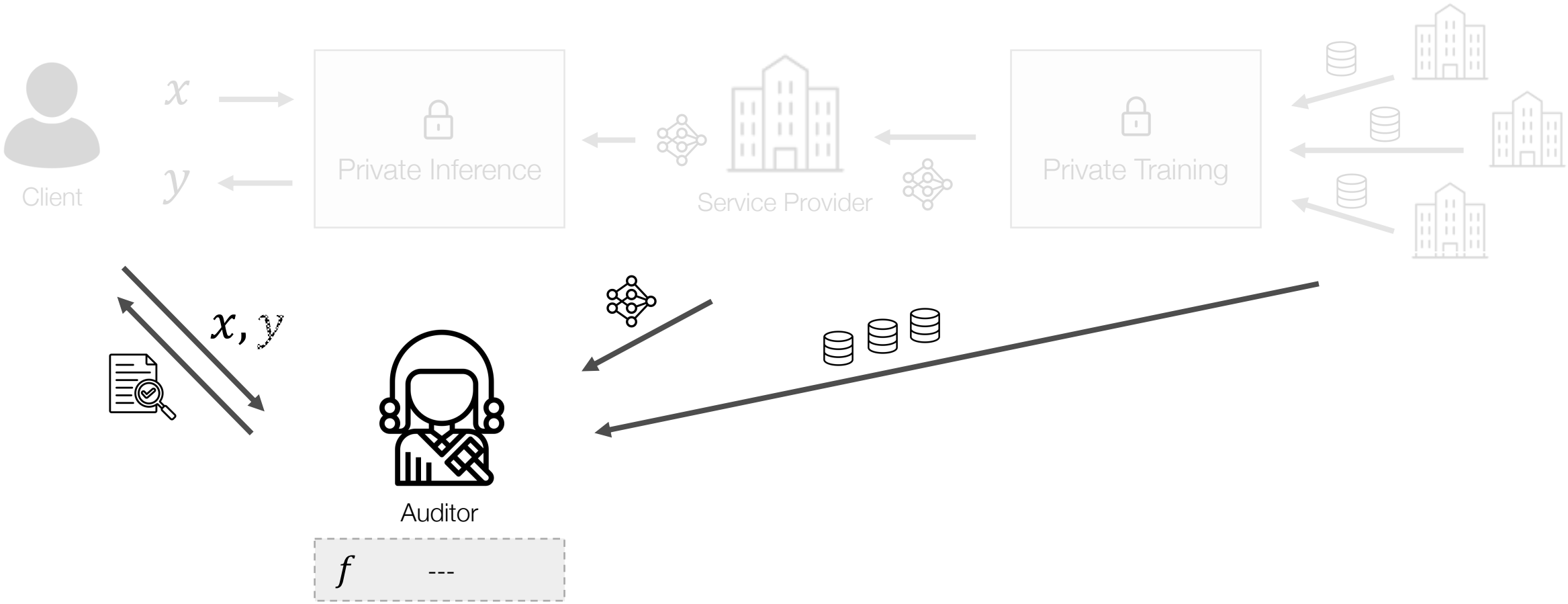
Auditing Privacy-Preserving Machine Learning



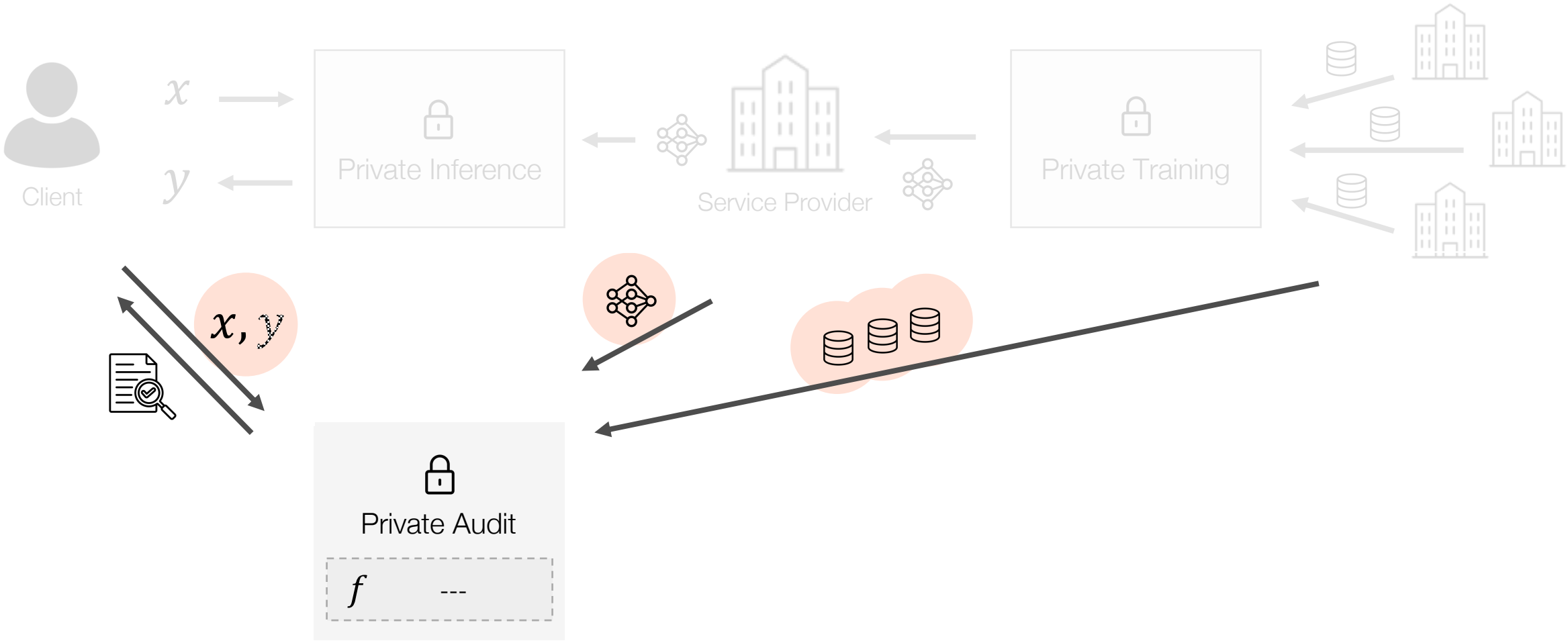
Post-Hoc Auditing of PPML



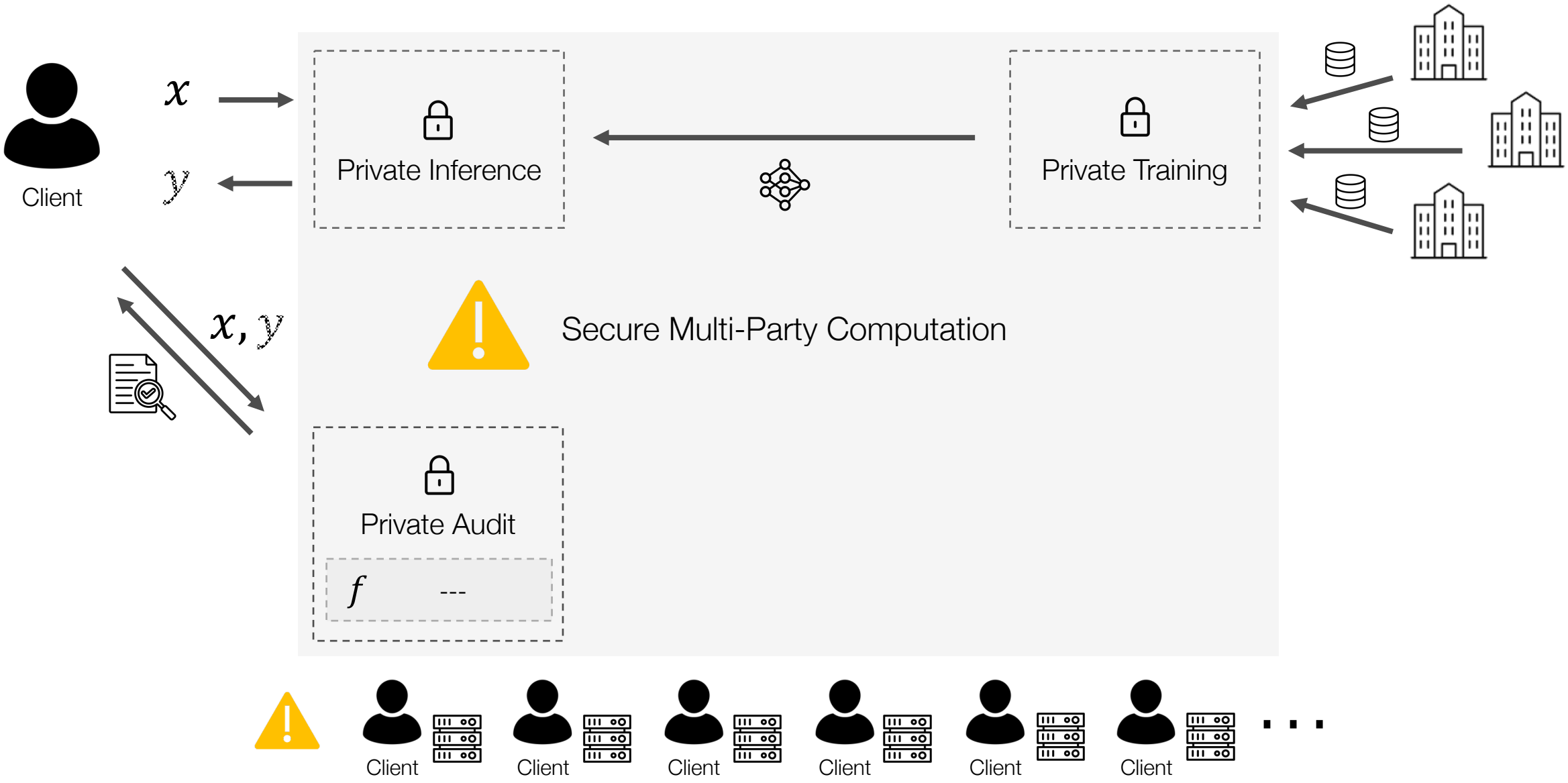
Post-Hoc Auditing of PPML



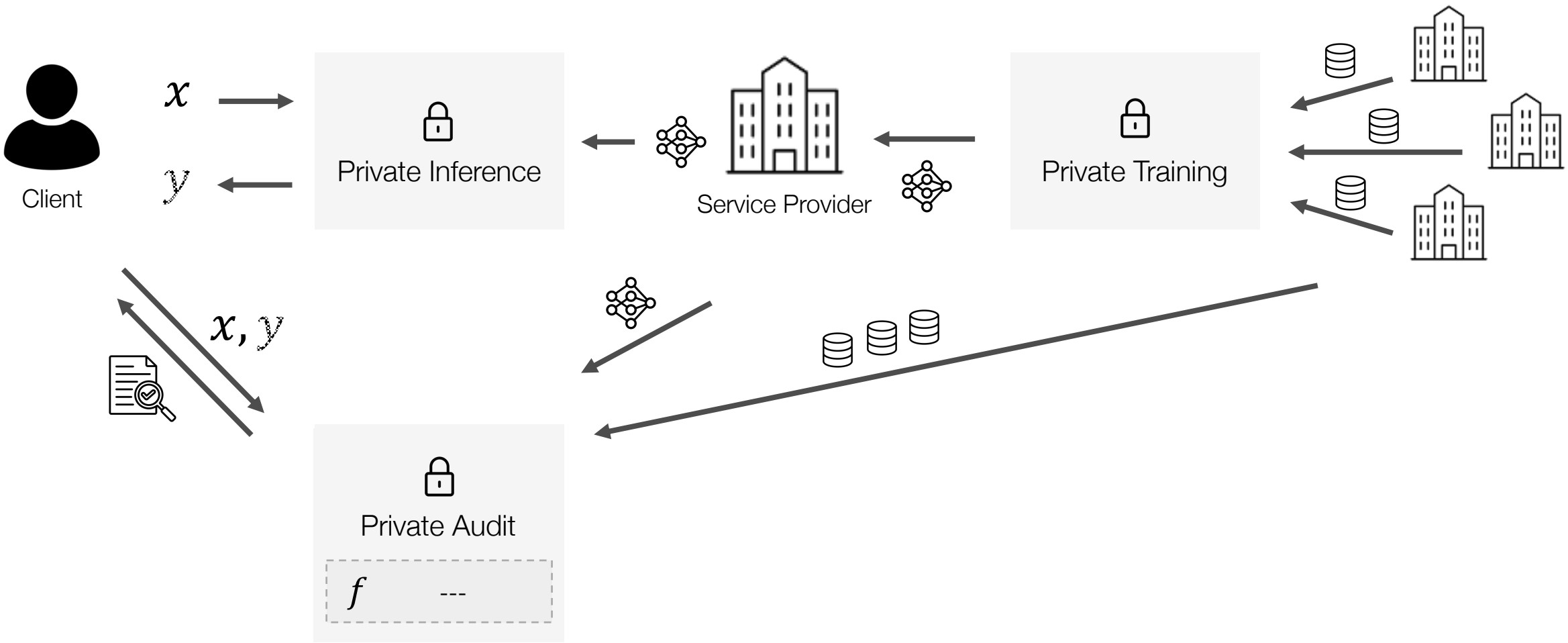
Post-Hoc Auditing of PPML



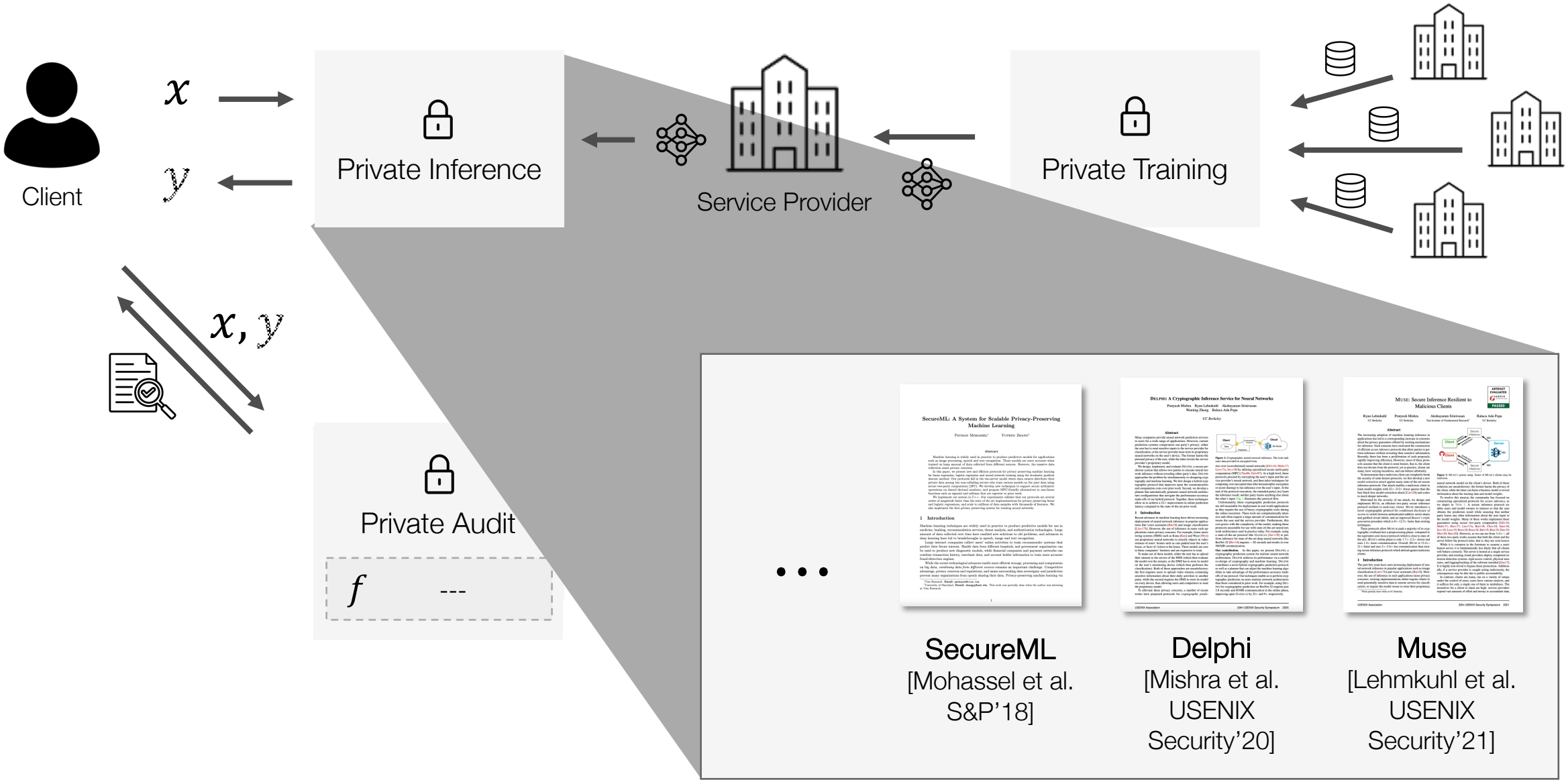
Post-Hoc Auditing of PPML



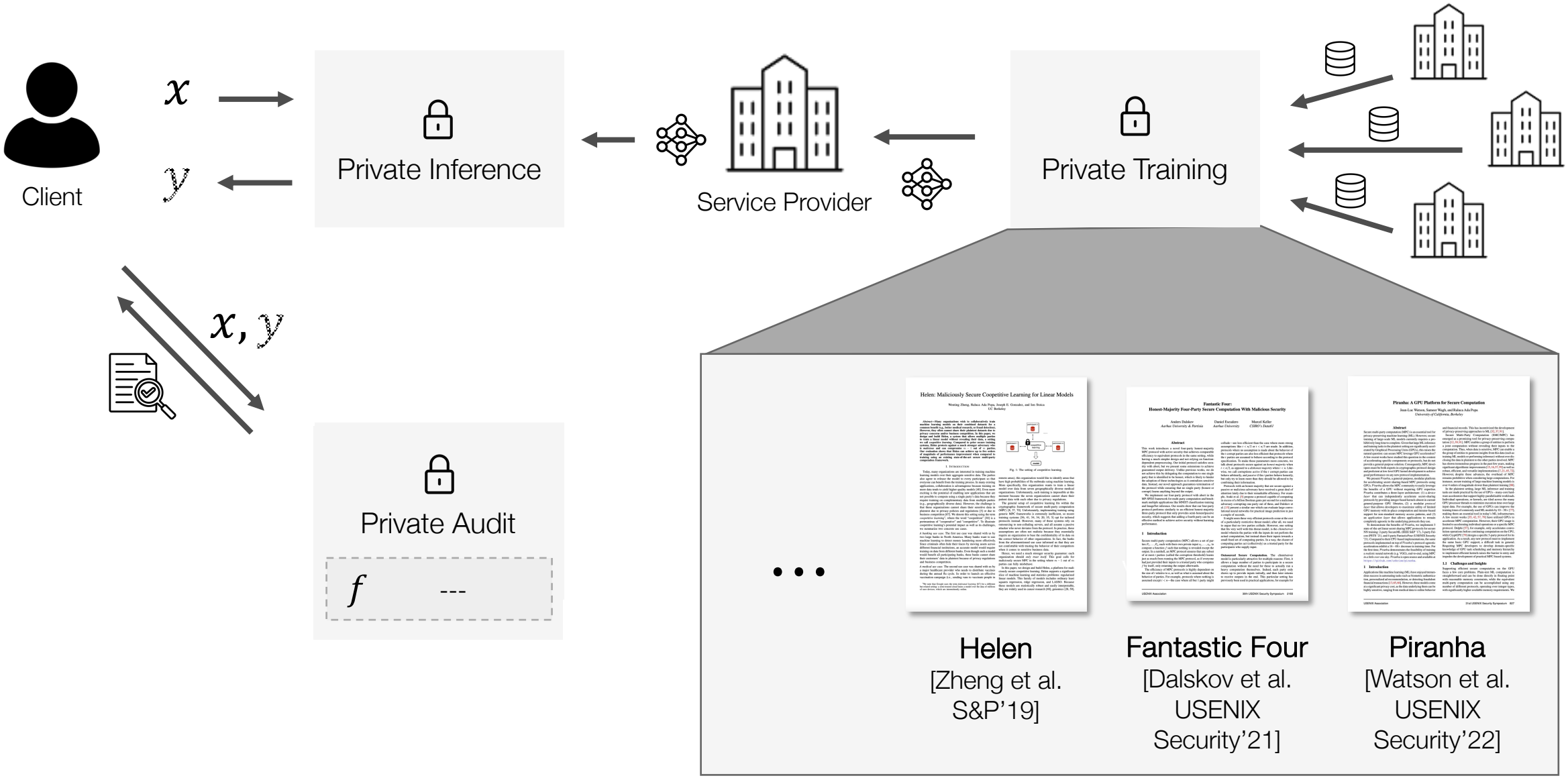
Post-Hoc Auditing of PPML: MPC Phases



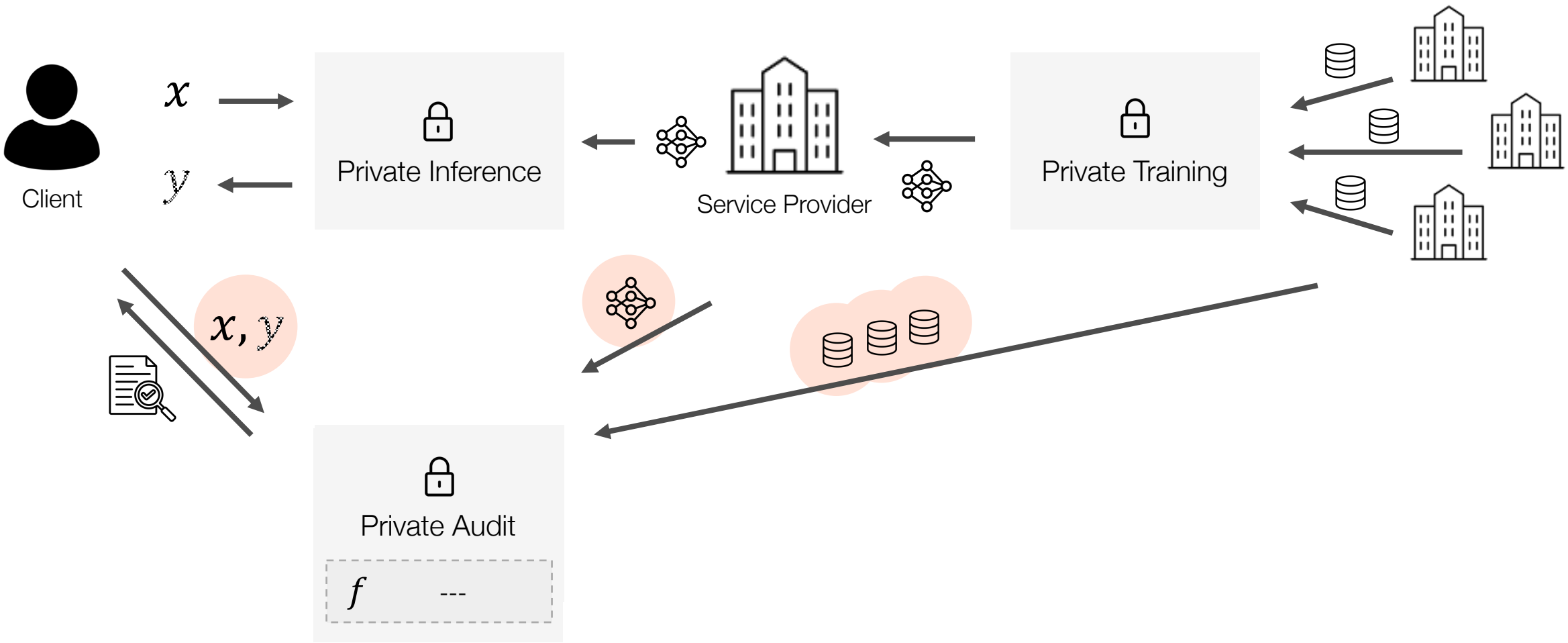
Post-Hoc Auditing of PPML: MPC Phases



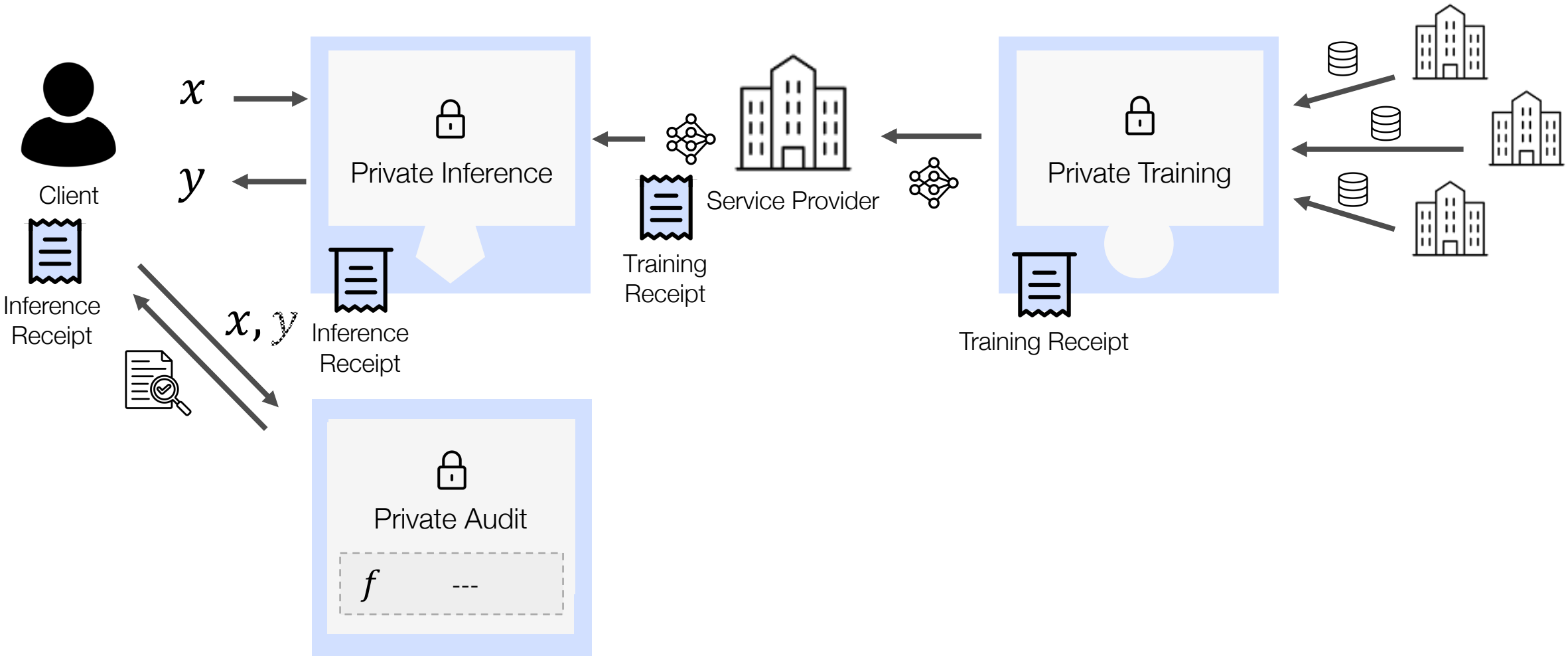
Post-Hoc Auditing of PPML: MPC Phases



Post-Hoc Auditing of PPML



This Work: Arc, a Framework for End-to-End Auditing of PPML



Arc Protocol: Training (MPC)

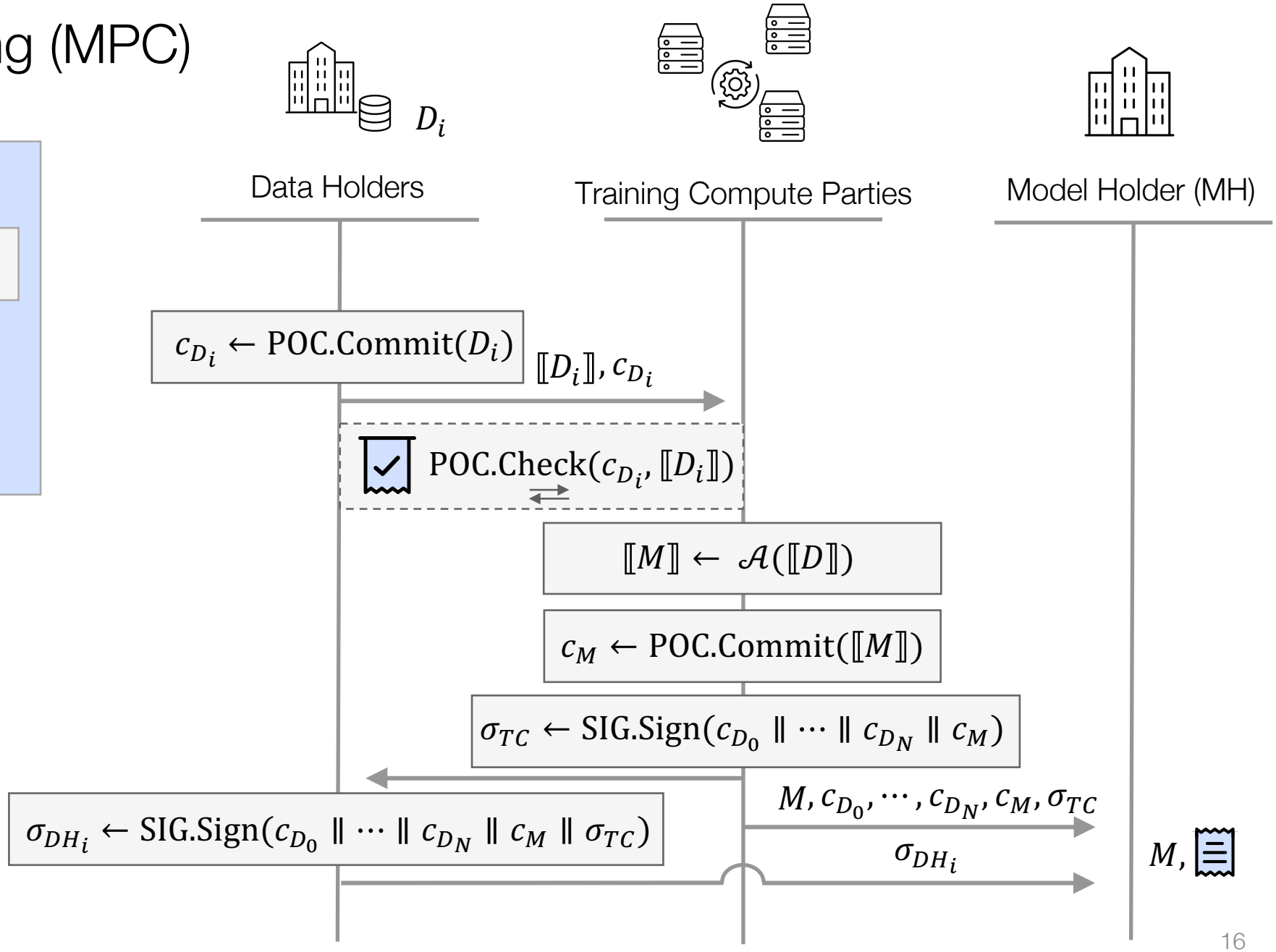
1. Commit to Input

PPML Training

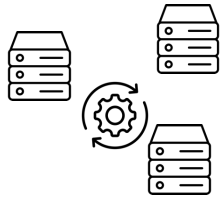
2. Commit to Result
3. Integrity of Computation
4. Integrity of Input

Training Receipt

- $c_{D_0}, \cdots, c_{D_N}, c_M$
- $\sigma_{DH_0}, \cdots, \sigma_{DH_N}, \sigma_{TC}$



Arc Protocol: Inference (MPC)



1. Commit to Input

PPML Inference

2. Commit to Result
3. Integrity of Computation
4. Integrity of Input

Inference Receipt

- $c_{D_0}, \cdots, c_{D_N}, c_M$
  $\sigma_{DH_0}, \cdots, \sigma_{DH_N}, \sigma_{TC}$
- c_x, c_y
  σ_{MH}, σ_{IC}

Model Holder (MH) | Computation Parties | Client

$[\![M]\!]$,

$c_x \leftarrow \text{POC.Commit}(x)$

$[\![x]\!], c_x$

$\text{POC.Check}(c_M, [\![M]\!])$

$\text{POC.Check}(c_x, [\![x]\!])$

$[\![y]\!] \leftarrow [\![M]\!]([\![x]\!])$

$c_y \leftarrow \text{POC.Commit}([\![y]\!])$

$\sigma_{IC} \leftarrow \text{SIG.Sign}(c_{D_0} \| \cdots \| c_{D_N} \| c_M \| c_x \| c_y)$

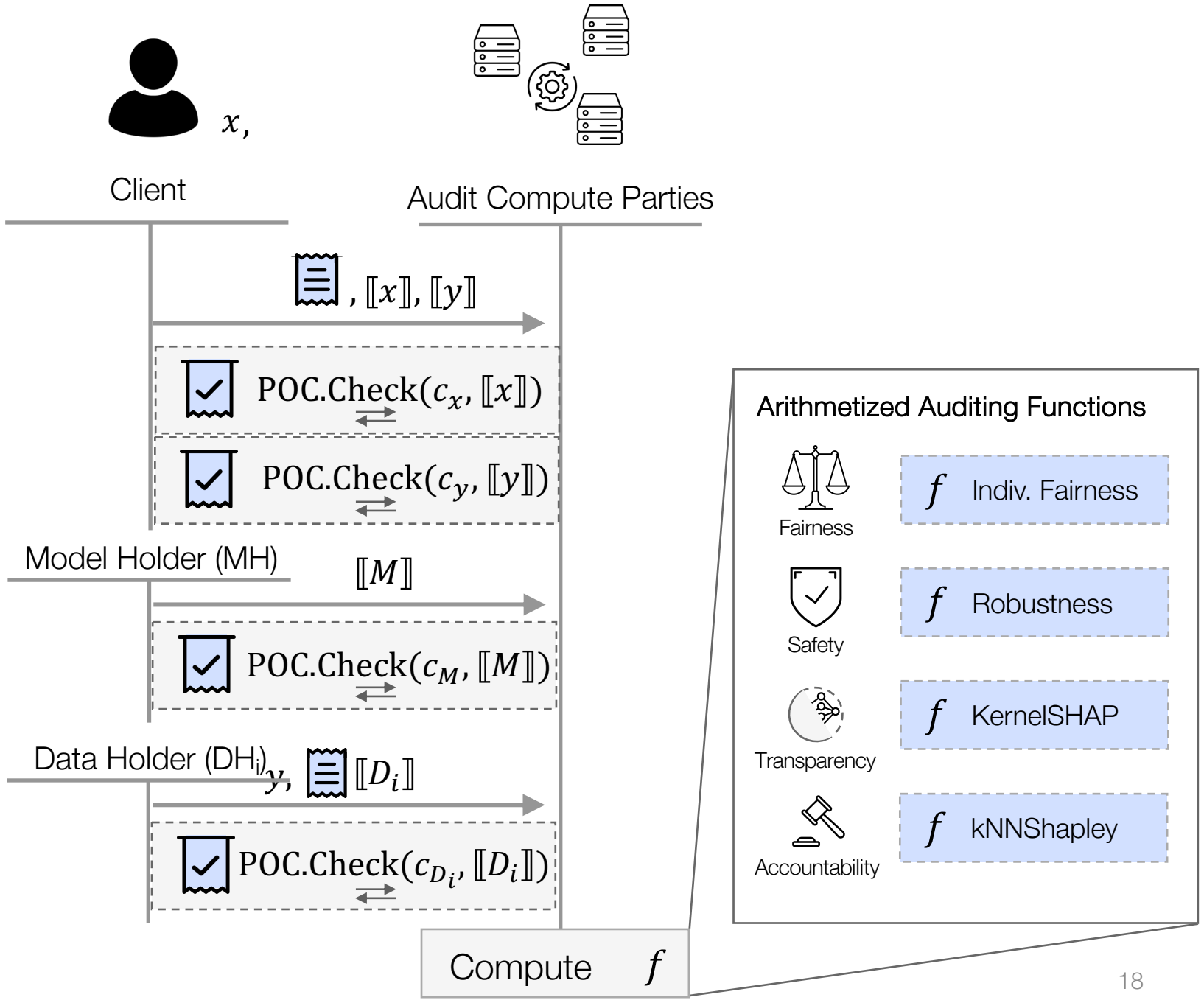
$y, c_{D_0}, \cdots, c_{D_N}, c_M, \sigma_{IC}$

y,

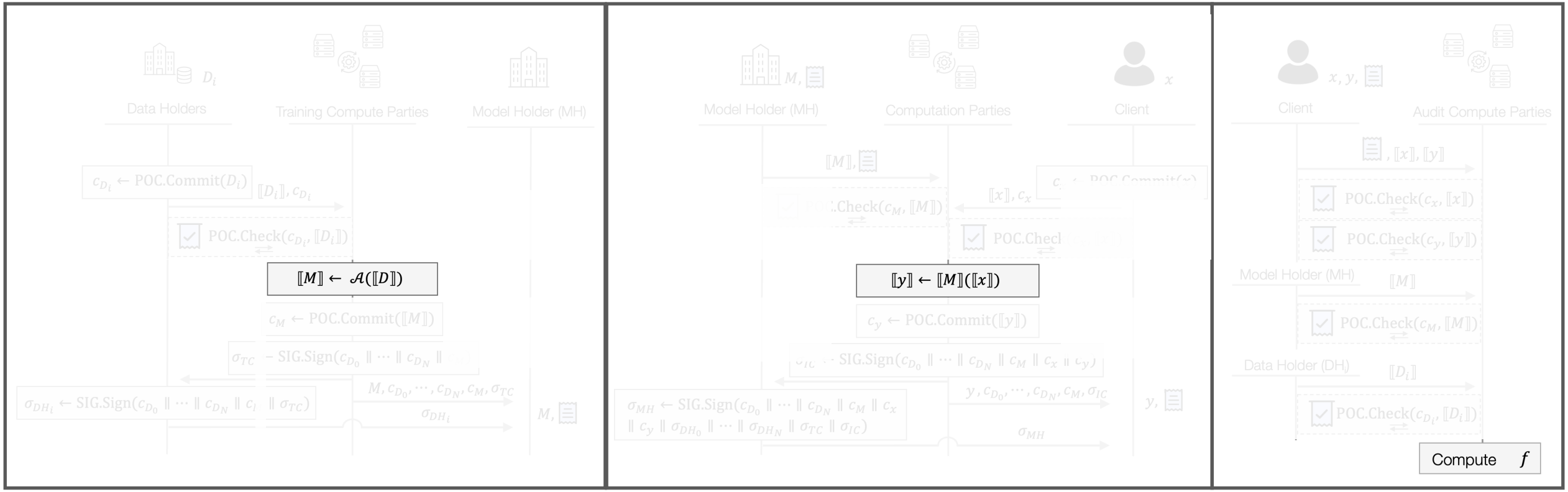
$\sigma_{MH} \leftarrow \text{SIG.Sign}(c_{D_0} \| \cdots \| c_{D_N} \| c_M \| c_x \| c_y \| \sigma_{DH_0} \| \cdots \| \sigma_{DH_N} \| \sigma_{TC} \| \sigma_{IC})$

σ_{MH}

Arc Protocol: Auditing

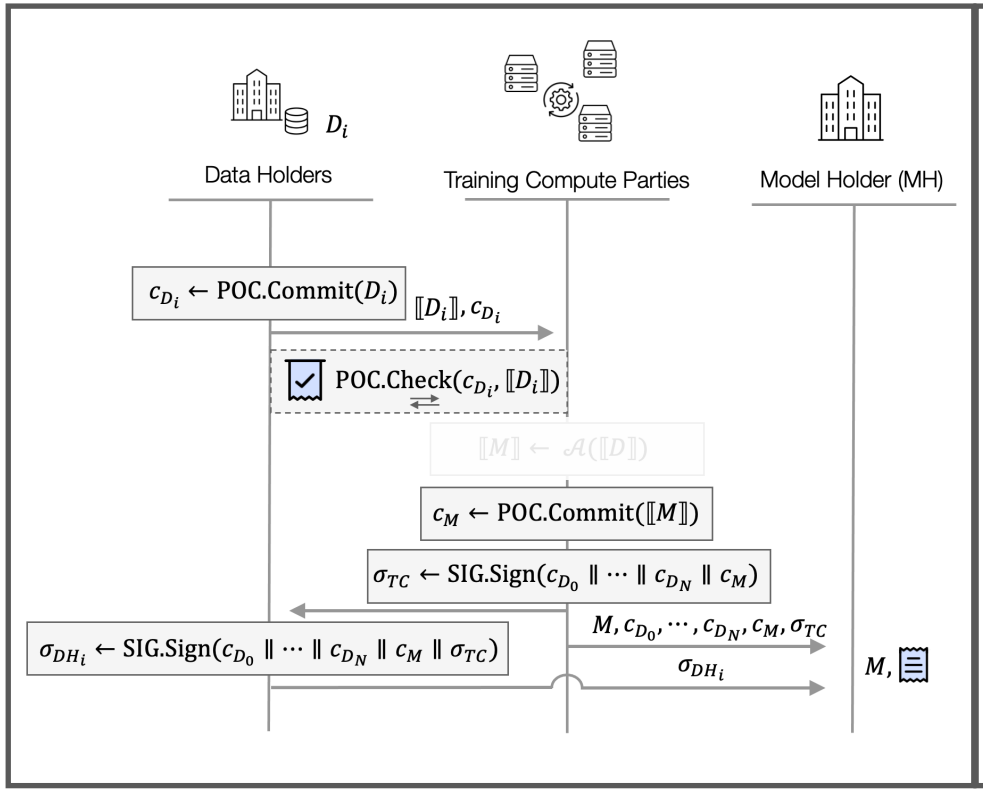


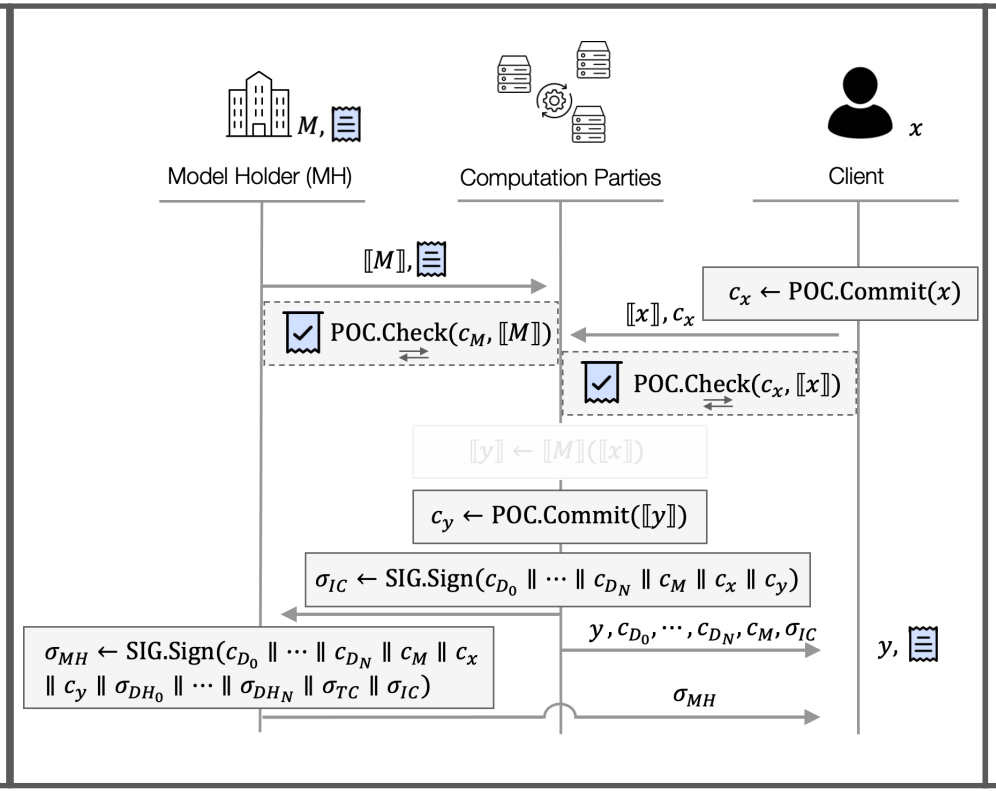
Arc Protocol

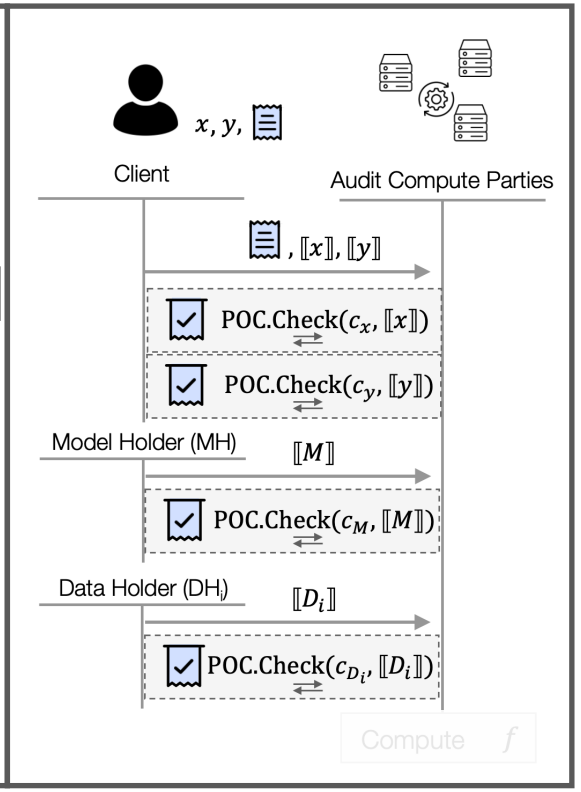


Training Inference Auditing

Arc Protocol





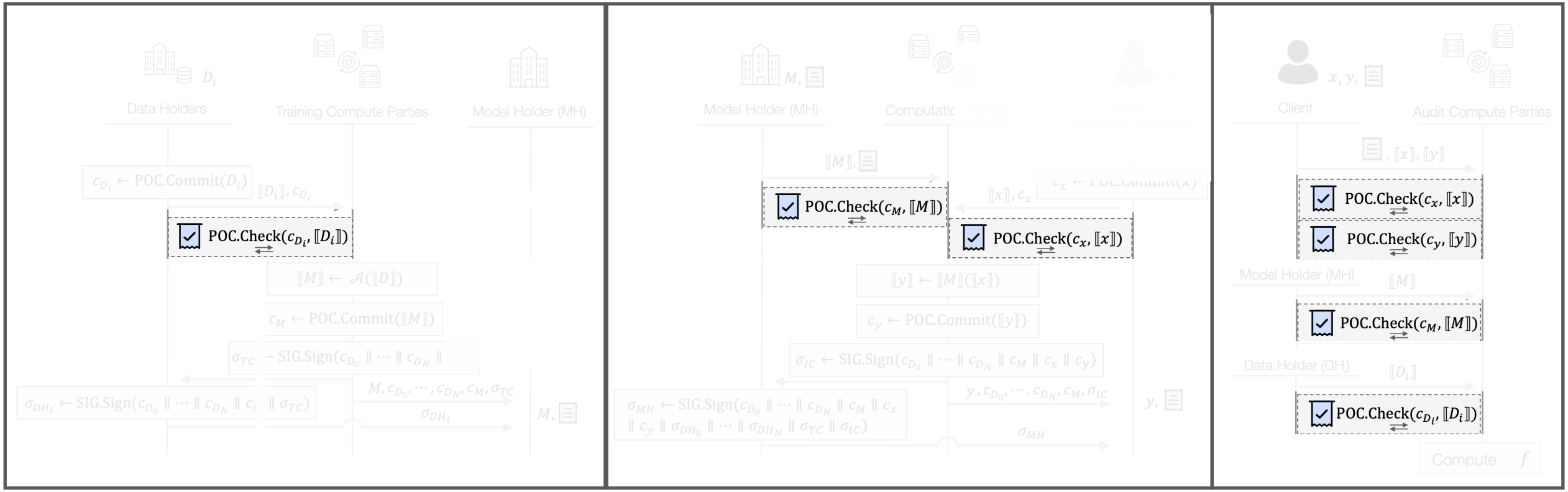


Training

Inference

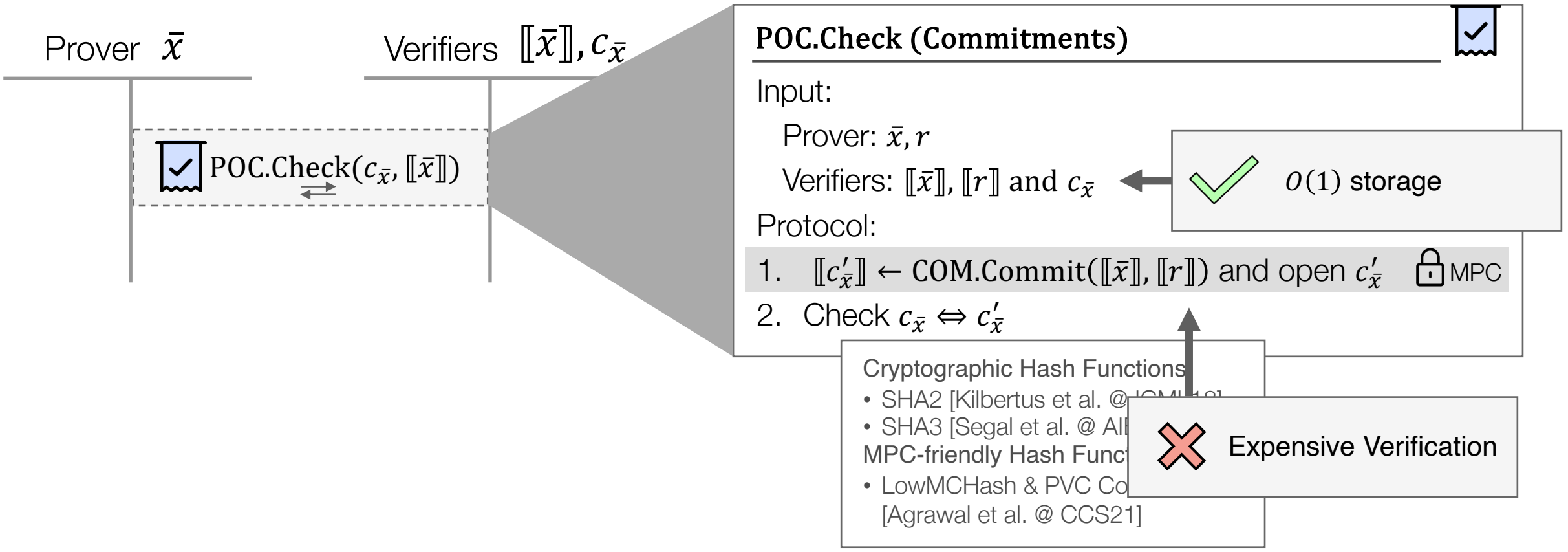
Auditing

Arc Protocol



Training Inference Auditing

Proof of Consistency



Proof of Consistency

POC.Check (Homomorphic Commitments)

Input:

Prover: $\bar{x} = [x_1 \quad \cdots \quad x_d], r = [r_1 \quad \cdots \quad r_d]$

Verifiers: $[\![\bar{x}]\!], [\![r]\!]$ and $c_{\bar{x}} = [c_{x_1} \quad \cdots \quad c_{x_d}]$

Protocol:

1. Sample random challenge $\beta \xleftarrow{\$} \mathbb{F}_p$
2. Compute $[\![\tilde{x}]\!] = \sum_{i=0} [\![x_{i+1}]\!] \cdot \beta^i$, $[\![\tilde{r}]\!] = \sum_{i=0} [\![r_{i+1}]\!] \cdot \beta^i$
3. Compute $[\![\tilde{c}']\!] \leftarrow \mathrm{PED.Com}([\![\tilde{x}]\!], [\![\tilde{r}]\!])$ and open $\tilde{c}'$
4. Check $\sum_{i=0} c_{x_{i+1}} \cdot \beta^i \Leftrightarrow \tilde{c}'$

MPC

$O(d)$ storage

$O(1)$ EC-MPC computation

$O(d)$ local EC computation

Cerebro [Zheng et al. @ USENIX Sec'21]

Our Proof of Consistency

POC.Check (Polynomial Commitments)

Input:

Prover: $\bar{x} = [x_1 \quad \dots \quad x_d], r$

Verifiers: $[\![\bar{x}]\!], [\![r]\!]$ and $c_{\bar{x}}$

Protocol:

+ masking

1. Sample random challenge $\beta \xleftarrow{\$} \mathbb{F}_p$
2. Compute $[\![\rho]\!] = \sum_{i=1} [\![x_i]\!] \cdot \beta^i$ and open ρ (MPC)
3. Prover computes $\pi \leftarrow \mathbf{PC.Prove}(g_{\bar{x}}, c_{\bar{x}}, \beta, \rho)$
4. Each Verifier checks $\mathbf{PC.Verify}(c_{\bar{x}}, \beta, \rho, \pi) \Leftrightarrow 1$

✓ Polynomial Commitment: Constant size

✓ $O(d)$ local field operations

✓ Verification $\ll O(d)$ and supports batching for multiple provers

Idea: Perform equality test using polynomial commitments

Evaluation

- End-to-end evaluation of Arc (integrated with MP-SPDZ), where POC is instantiated with:
  - **Ours:** Polynomial Commitments (KZG)
  - **SHA3:** Hash-based Commitments [Segal et al. @ AIES21]
  - **PED:** Pedersen Commitments [Cerebro, Zheng et al. @ USENIX Sec'21]

- Setting: 3PC semi-honest / malicious
- Auditing Functions: Indiv. Fairness, KernelSHAP, Robustness, kNNShapley

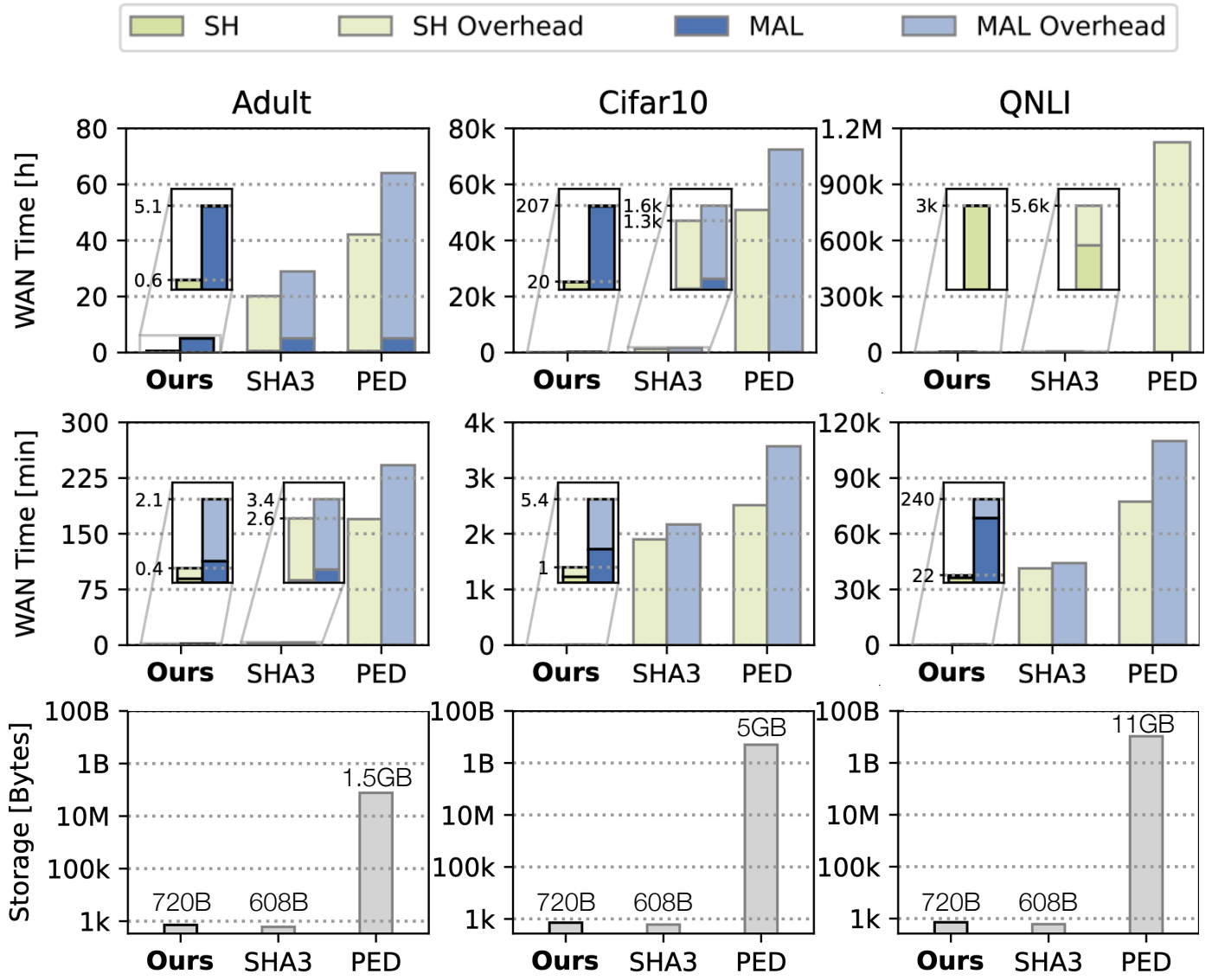
* The training overhead for malicious protocols and larger instances of the SHA3 and PED baselines are extrapolated

Evaluation: Training & Inference

Training (1 epoch)

Inference

Inference Receipt

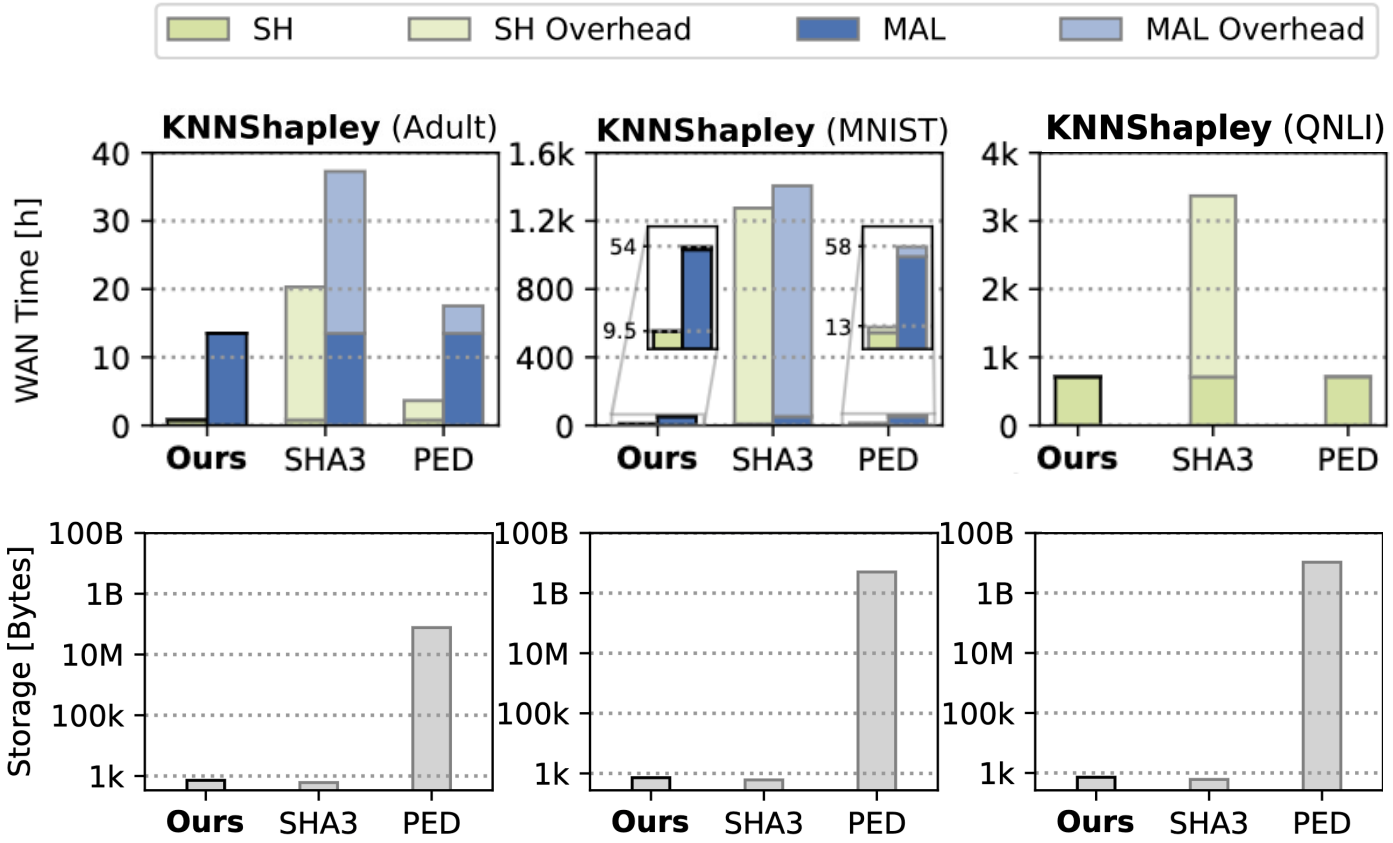


Ours: Polynomial Commitments
SHA3: Hash-based Commitments
PED: Homomorphic Commitments

Evaluation: Auditing

Auditing

Inference Receipt



Ours: Polynomial Commitments
SHA3: Hash-based Commitments
PED: Homomorphic Commitments

This Talk:
- Framework for Auditing PPML Pipelines
- New Protocol for Input Consistency in MPC

In the Paper:
- Protocol Extensions for Plaintext Training / Inference
- Arithmetic-to-Arithmetic Share Conversion
- MPC Arithmetizations of Auditing Functions

Future work:
- Robust Auditing Protocols
- Caching Auditing Inputs
- Verifiable Data Removal

pps-lab/arc

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

pps-lab.com/research/ml-sec