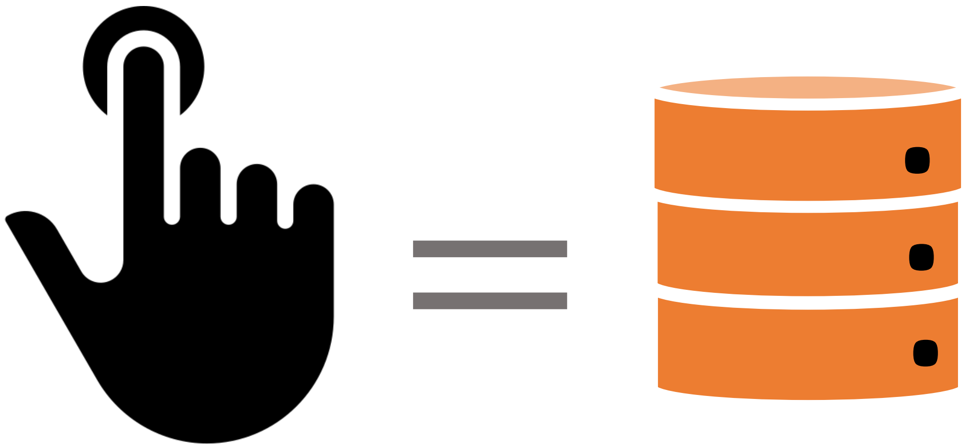
Data Privacy in the Decentralized Era

CS 388

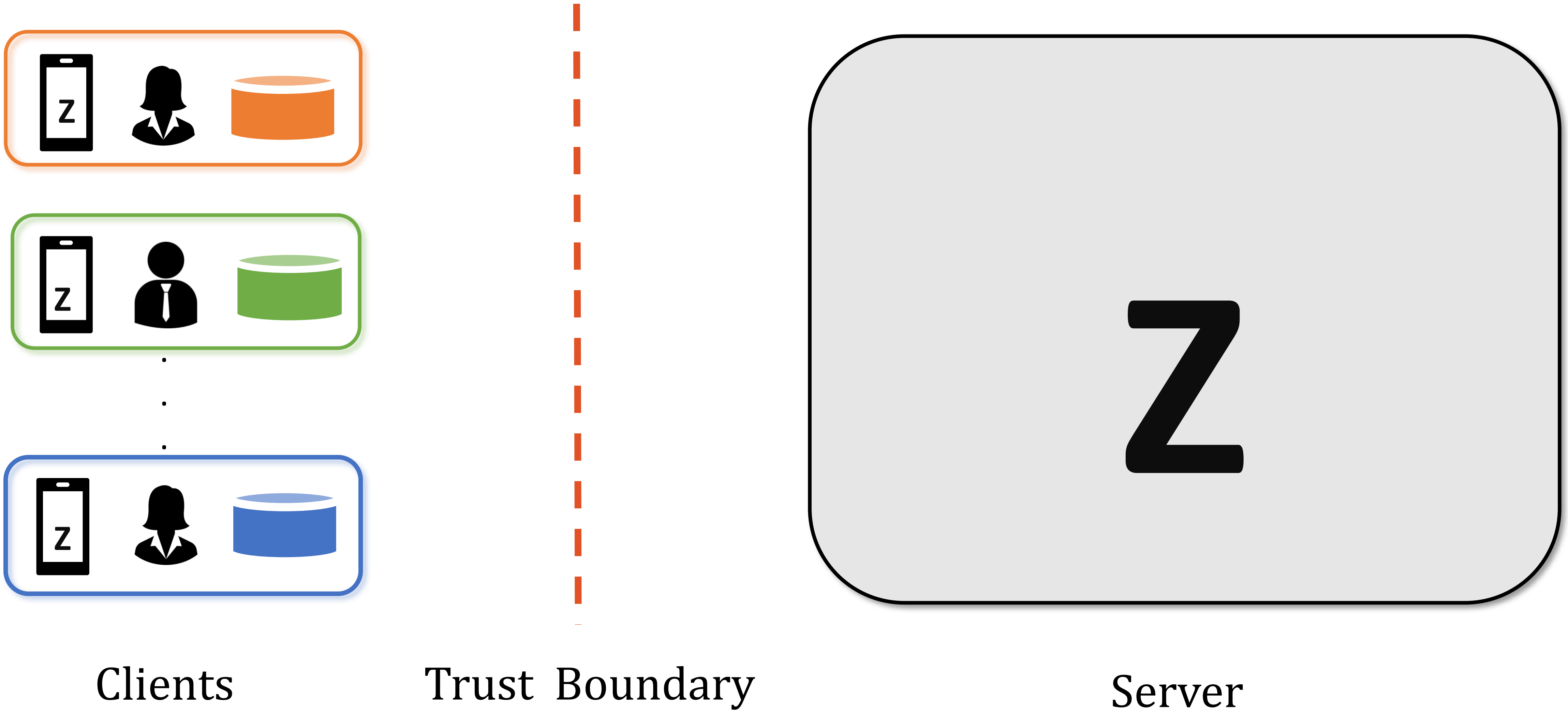
Amrita Roy Chowdhury

Data is born at the edge

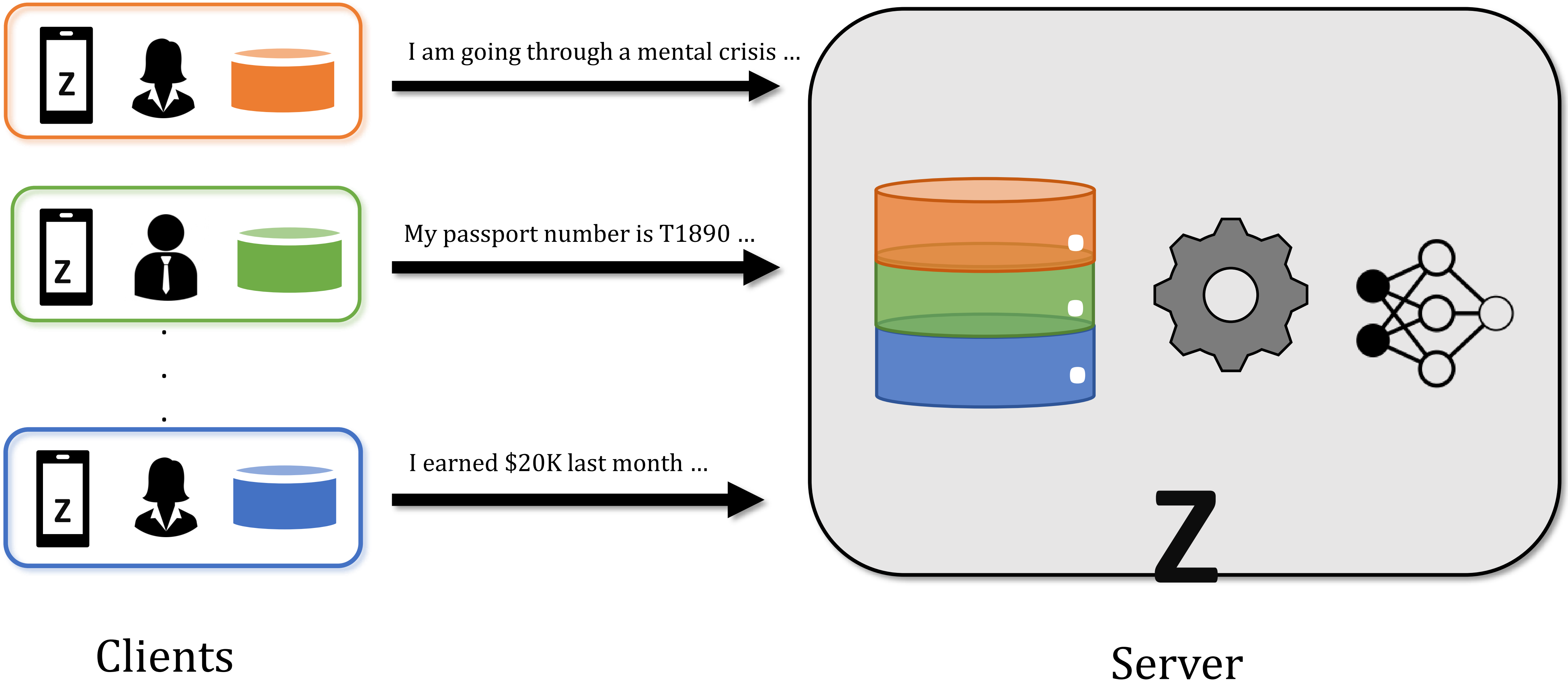
- Data is generated on smart devices
- Decentralized data ecosystem
  - Client-server architecture



Decentralized Data Ecosystem



Decentralized Data Ecosystem



Privacy Risks

- Generated data is **personal** and **sensitive**
  - Location history
  - Shopping history
  - Browsing history
  - Social media interactions …

- Snapshots of our daily lives

Facebook Gave Device Makers Deep Access to Data on Users and Friends

How Strangers Got My Email Address From ChatGPT's Model

'Mother of all breaches' data leak reveals 26 billion account records stolen from Twitter, LinkedIn, more

Oops, I Did It Again: Apple Faces Fourth iPhone Privacy Lawsuit After Gizmodo Story

Apple has been sued again—and again—because it collects data when its privacy settings promise not to, according to researchers' tests.


RIP
Privacy

Privacy Regulations

- Legal obligations due to **privacy regulations**
  - India – DPDPA
  - EU – GDPR, AI Act
  - Canada – PIPEDA, CPPA, AIDA
  - USA – Biden's Executive Order; Local state laws e.g. CCPA

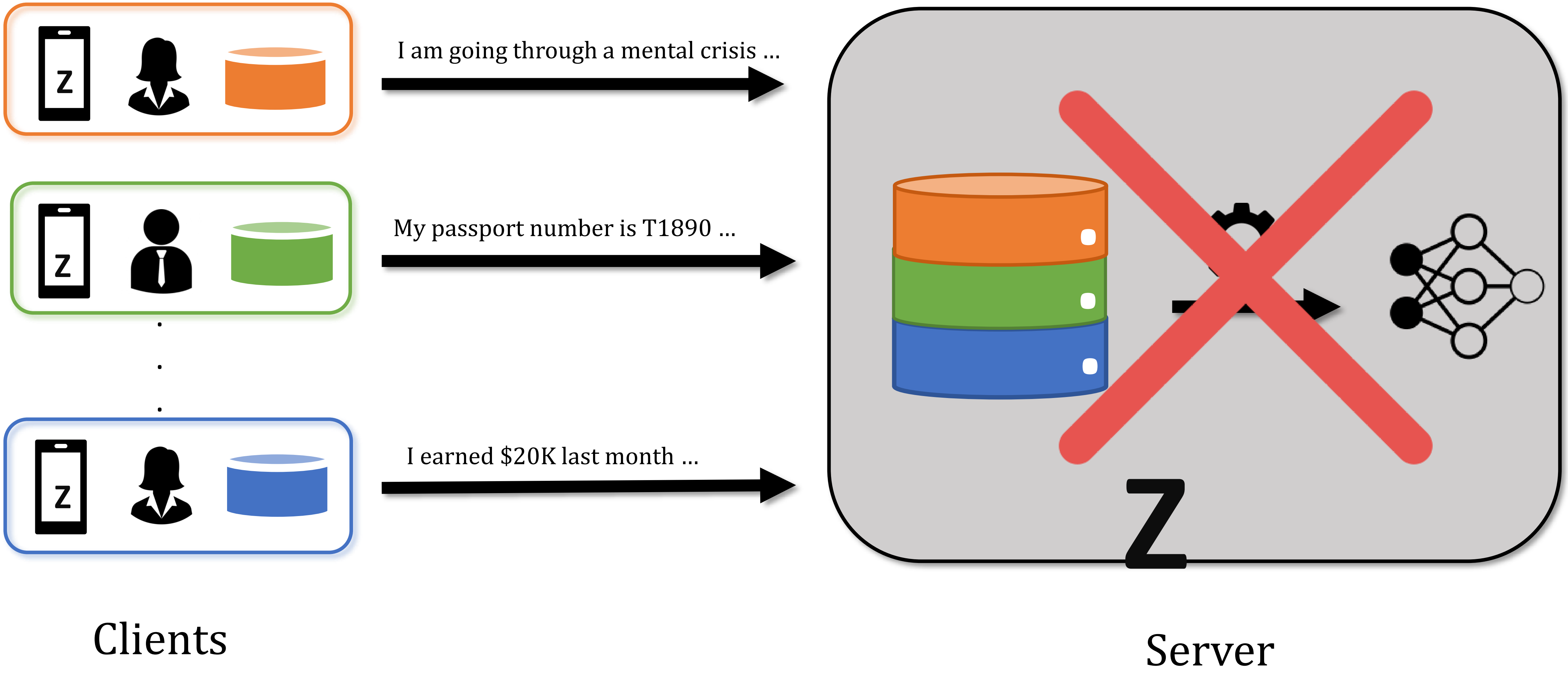
FACT SHEET: President Biden Issues Executive Order on Safe, Secure, and Trustworthy Artificial Intelligence

India's Digital Personal Data Protection Act (DPDPA) Demystified

California Consumer Privacy Act (CCPA): What you need to know to be compliant

What is GDPR? The summary guide to GDPR compliance

Decentralized Data Ecosystem



How to **privately** analyze data in a decentralized ecosystem with multiple **distrusting** entities?

My Research

End-to-end solutions for data privacy that

1. Provide **provable** guarantees
2. Preserve **data functionality**
3. Are **compatible** with real-world constraints

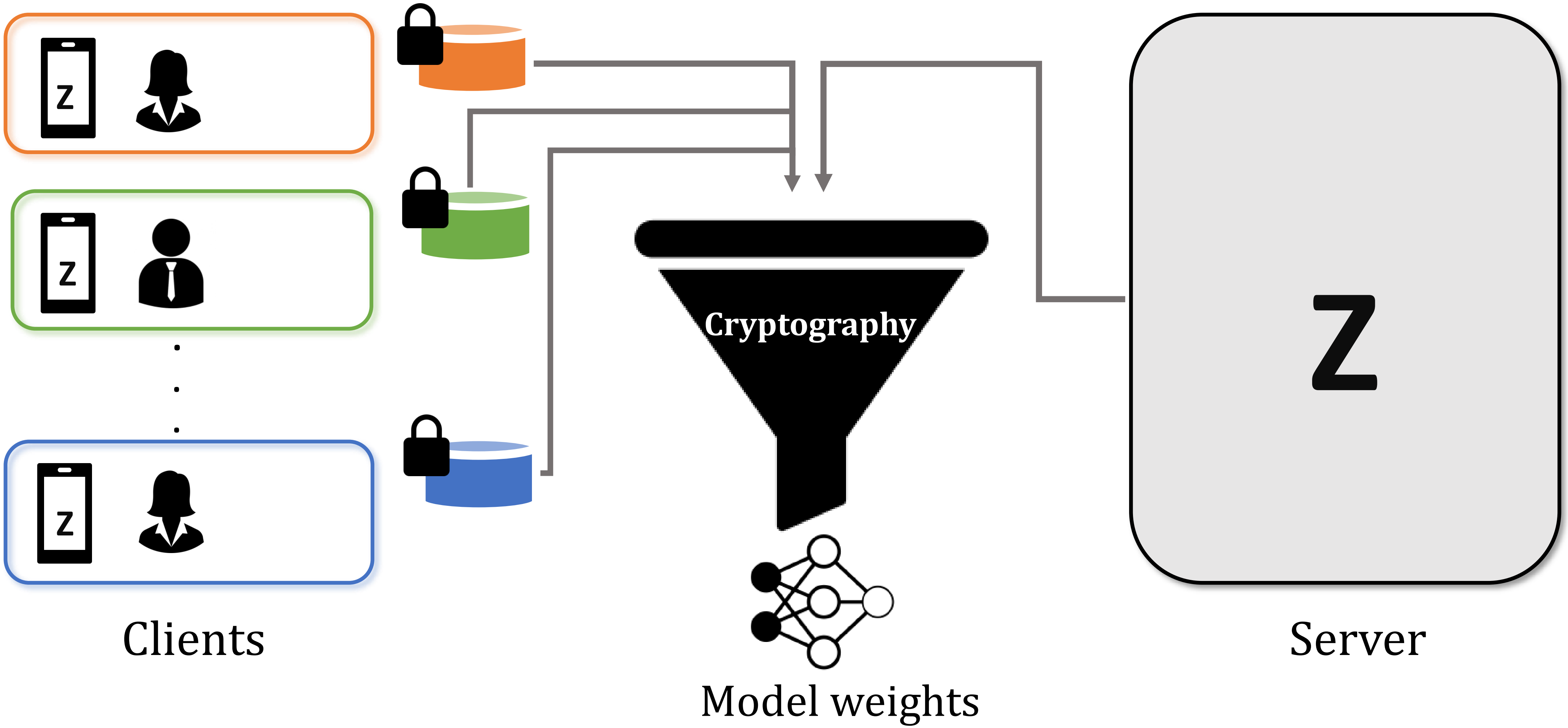
How to **privately** analyze data in a decentralized ecosystem with multiple ***distrusting*** entities?

Cryptography

Distrusting parties collaborate to

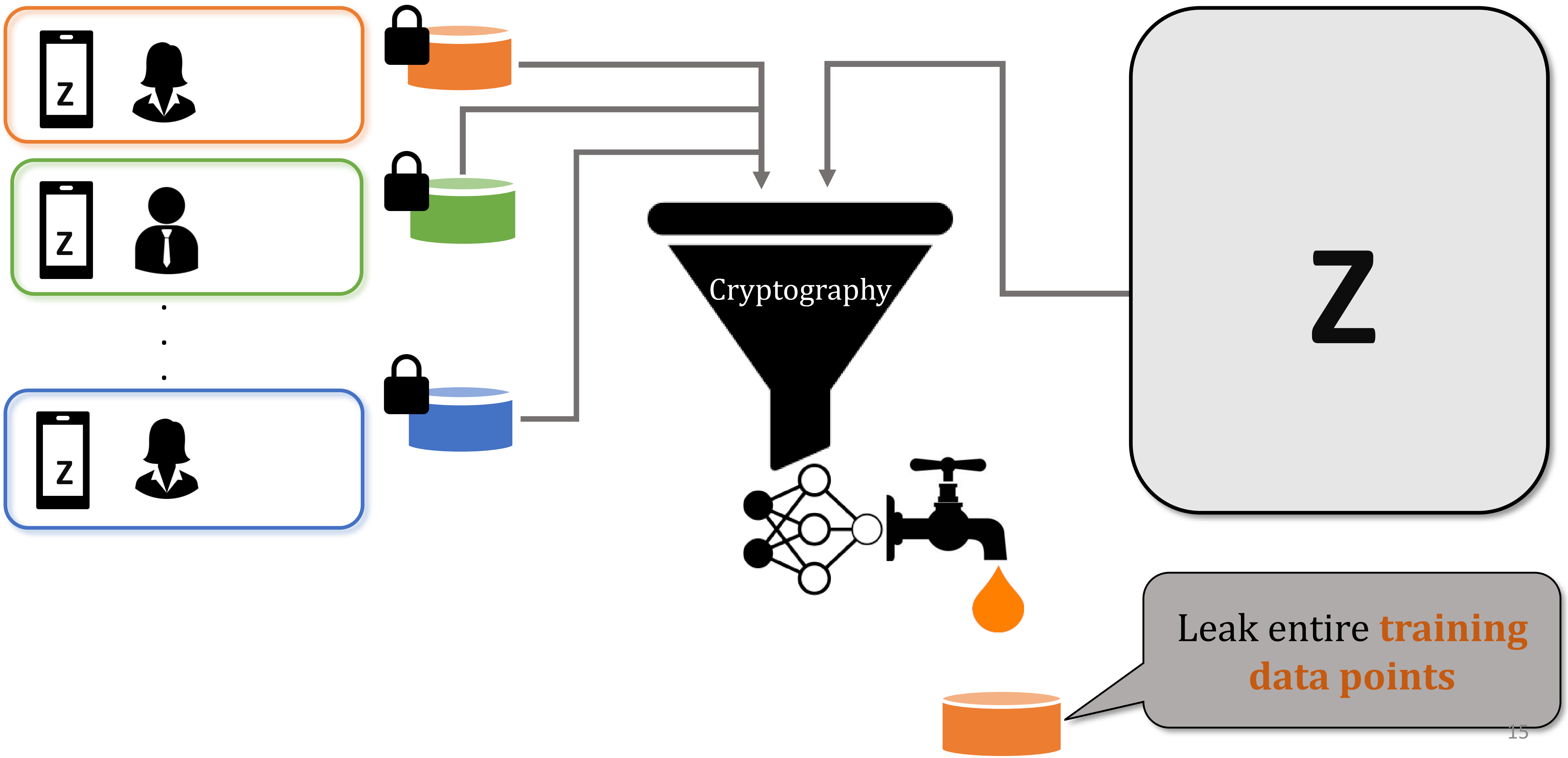
- Compute over joint dataset
- **Without seeing the data**

Cryptography



How to ***privately*** analyze data in a decentralized ecosystem with multiple **distrusting** entities?

Cryptography



- Cryptography protects only input privacy
- Output privacy is beyond scope

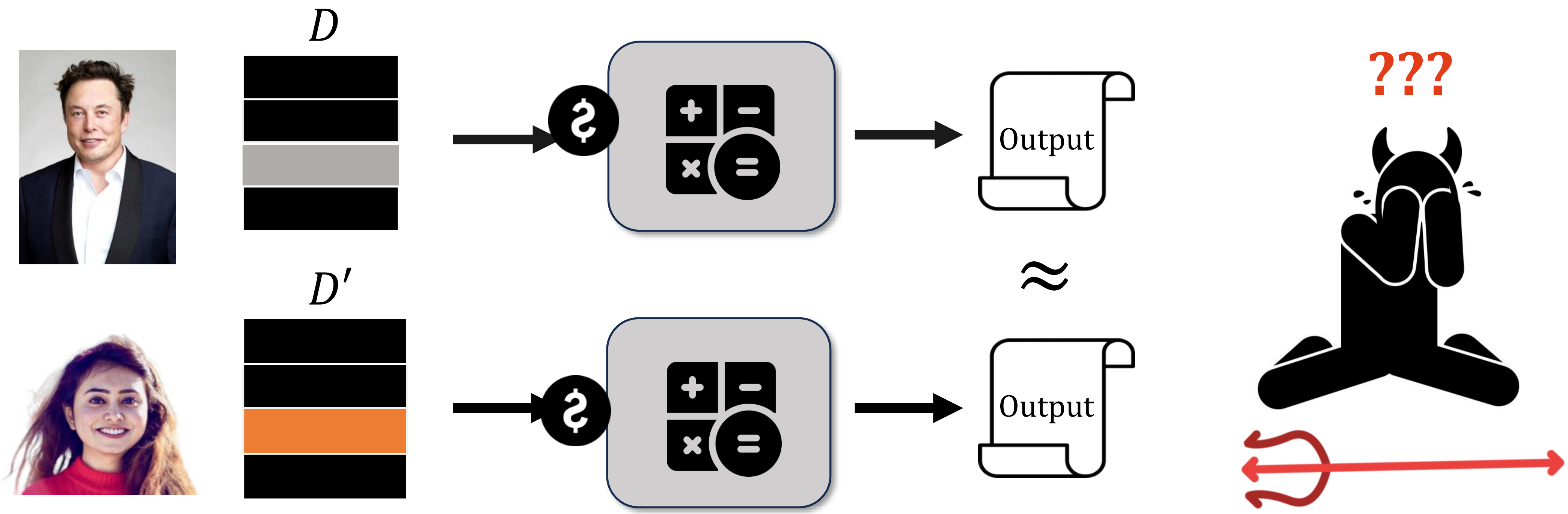
Differential Privacy (DP)

- Quantifies information **leakage from output**
- Imposes a formal bound on the **privacy risk to an individual**
- Add controlled noise to output

Differential Privacy (DP)

[Dwork06]

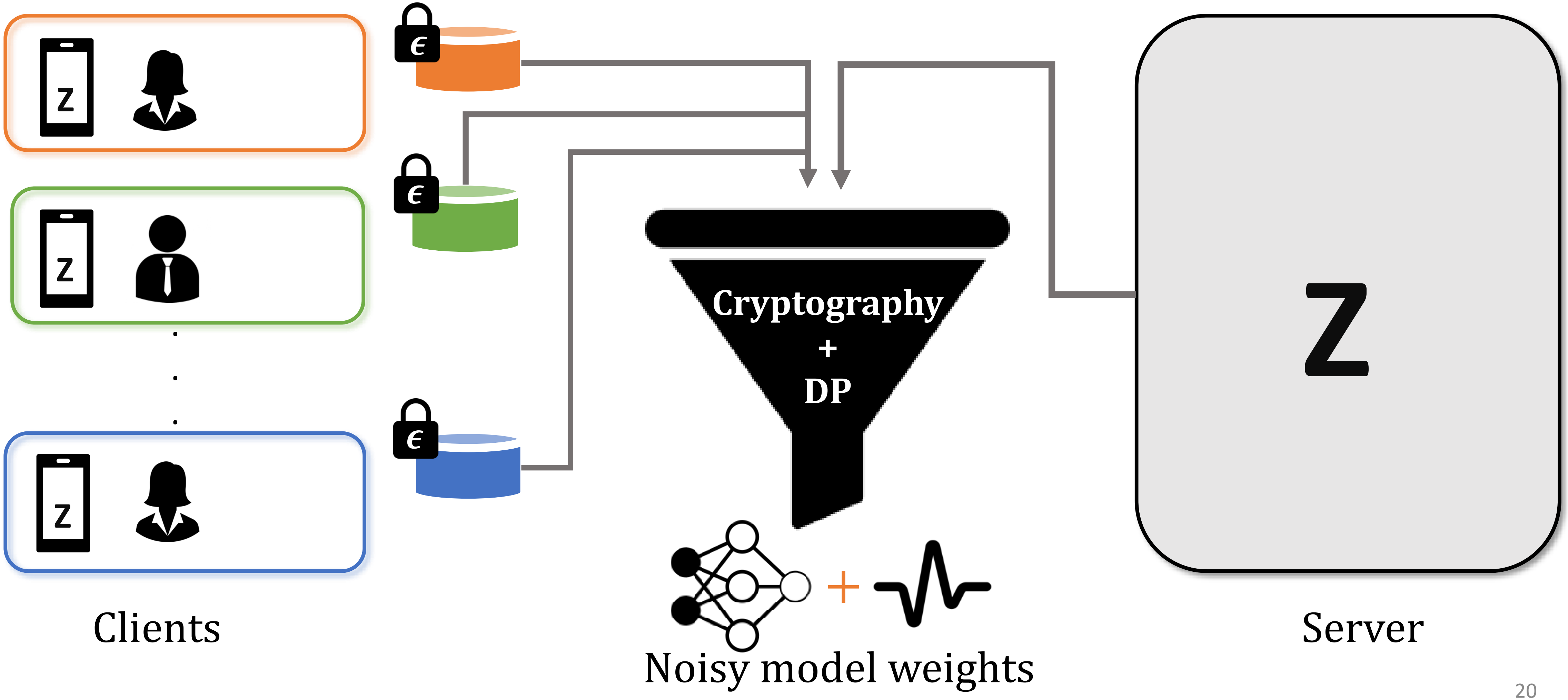
A randomized algorithm satisfies DP if its output is **insensitive** to changing **one** record in its input dataset



Differential Privacy (DP)

- Characterized by **privacy budget** or **parameter** ϵ
- **Smaller** $\epsilon \implies$ **Stronger** privacy

Cryptography + DP

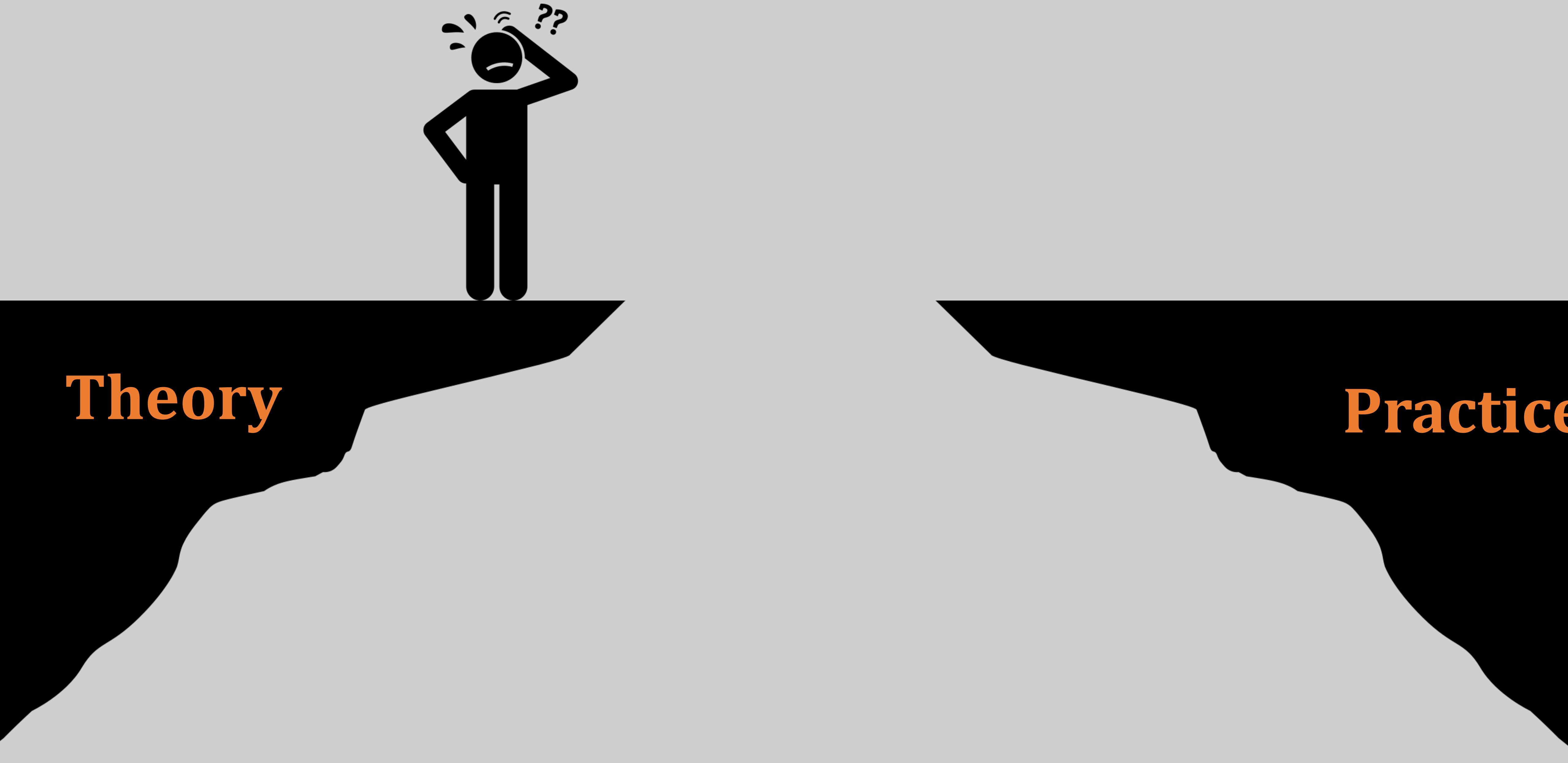


What about Practical Deployment?

Tools:

Cryptography – Over 35 years
Differential Privacy – Almost 20 years

Surprisingly, one of the first known real-world deployment happened **only in 2023**

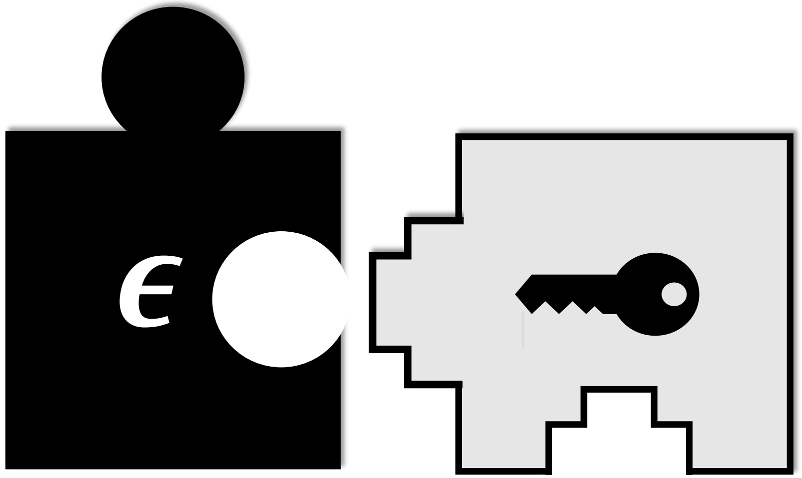

??
Theory
Practice

New Challenges

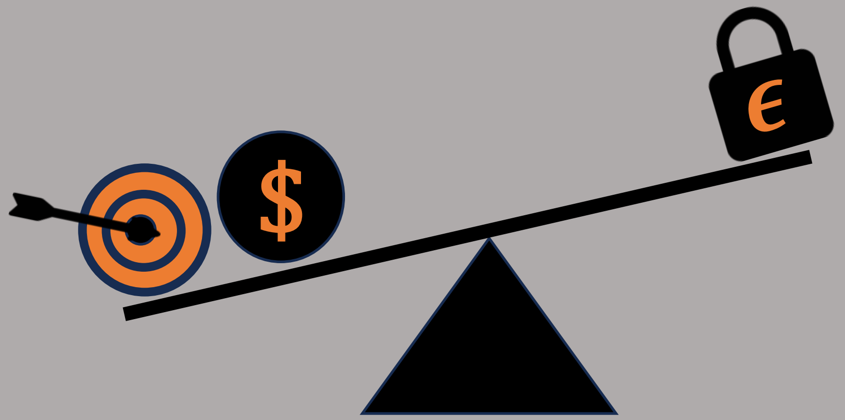
- Interaction between crypto and DP is **complex and context-dependent**

- Crypto works on masked data
  - Affects performance (computation/communication)
  - No impact on accuracy

- DP works on cleartext
  - Affects accuracy
  - No impact on performance



Gives rise to complex **privacy/performance/accuracy** trade-off

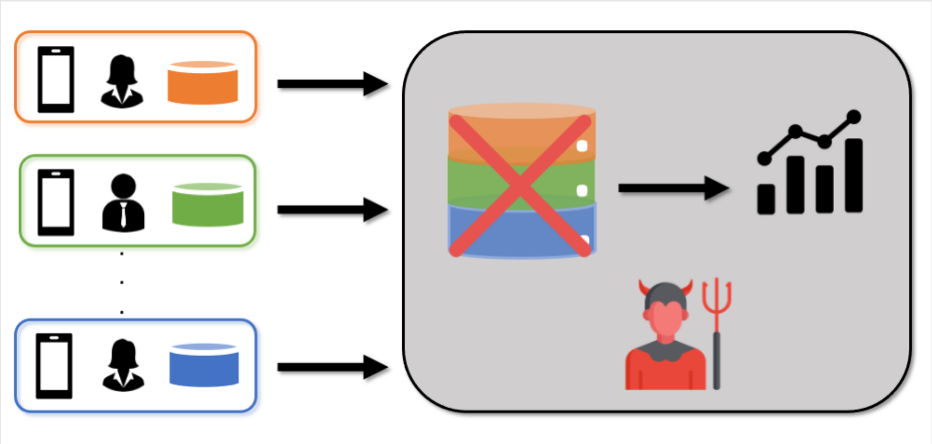


Solution Approach

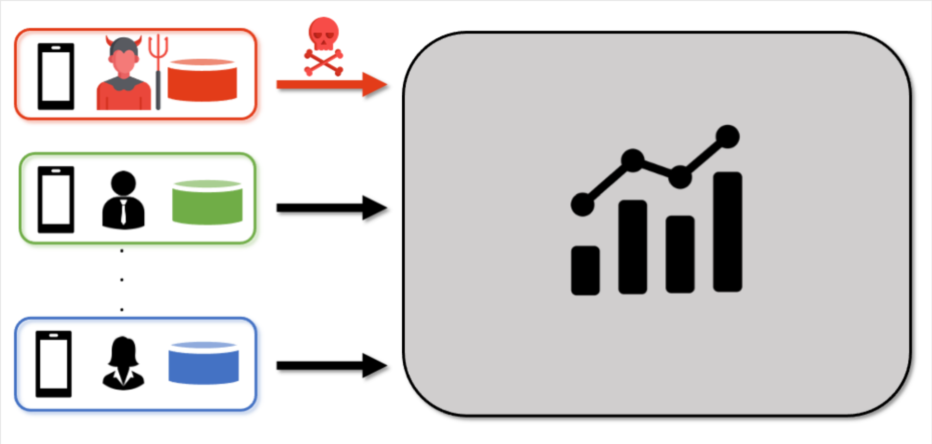
Principled exploration of the **synergy** between cryptography and DP for balancing the trade-off

Dual Threat

Untrusted Server

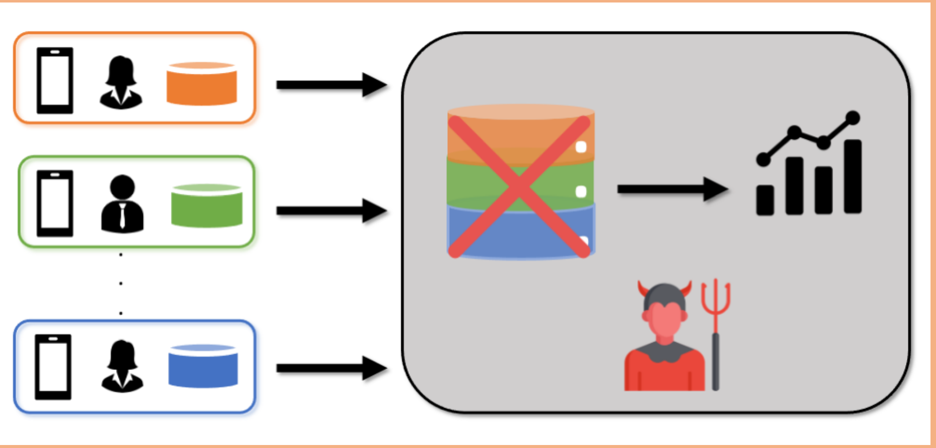


Untrusted Clients



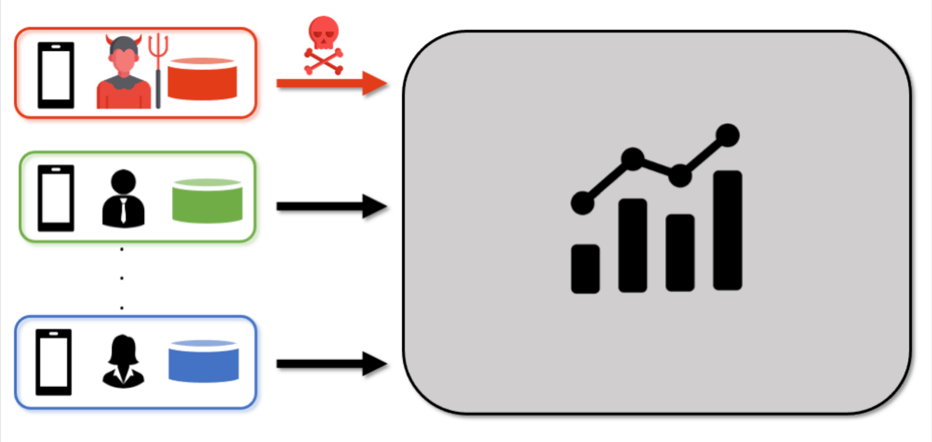
Dual Threat

Untrusted Server



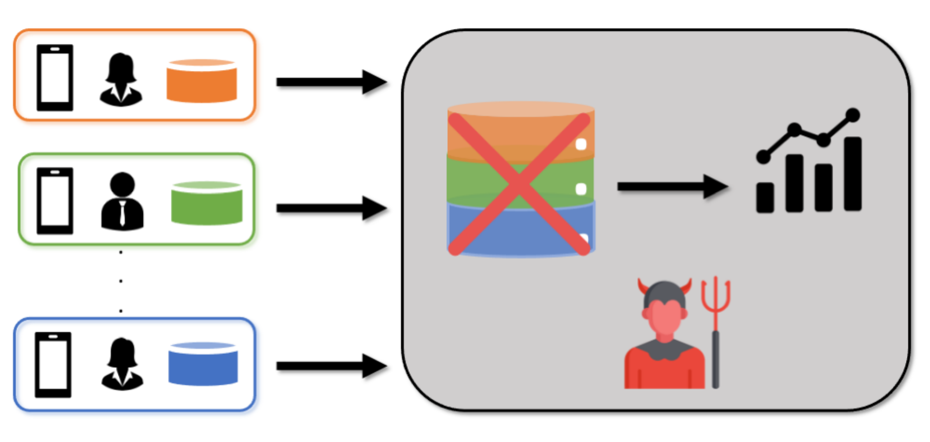
- Server wants to glean as much information as possible
  - Strongly incentivized to **violate client's privacy**

Untrusted Clients



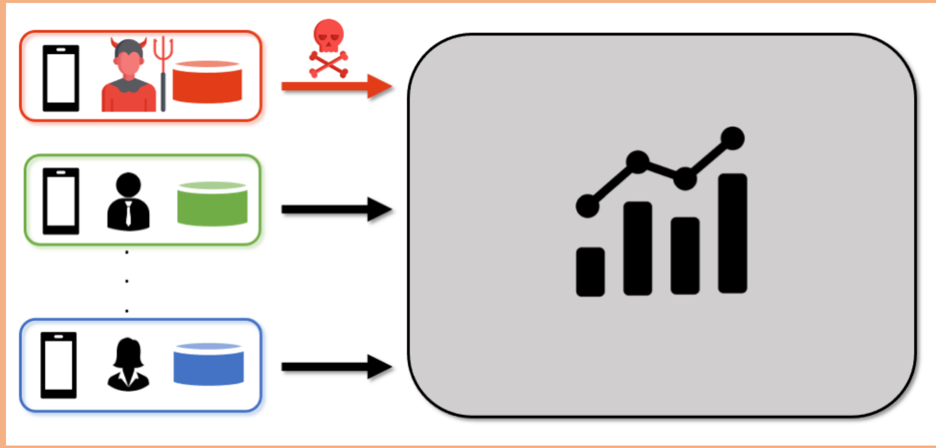
Dual Threat

Untrusted Server



- Server wants to glean as much information as possible
  - Strongly incentivized to **violate client's privacy**

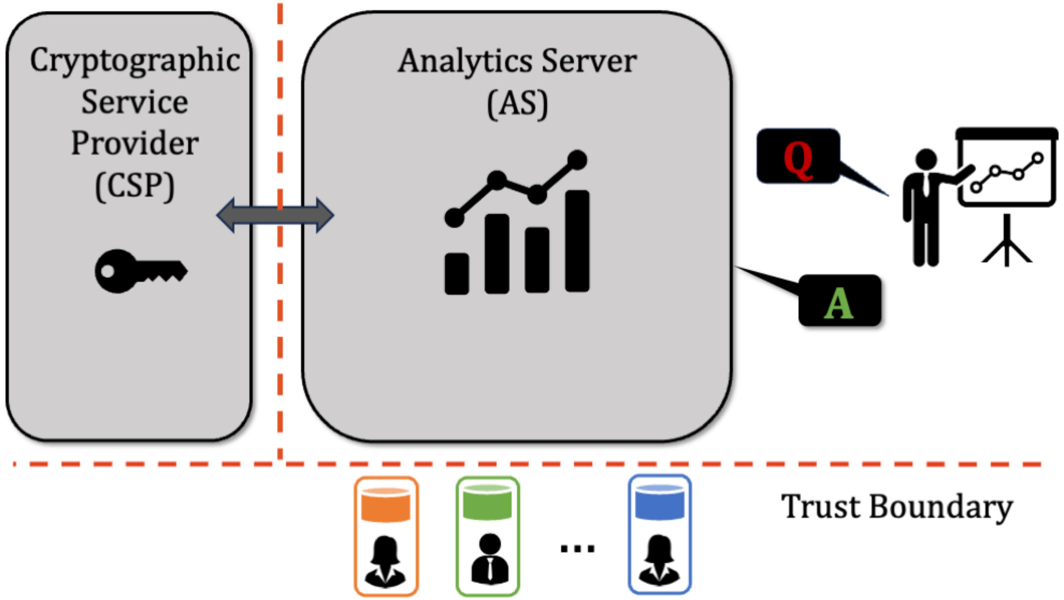
Untrusted Clients



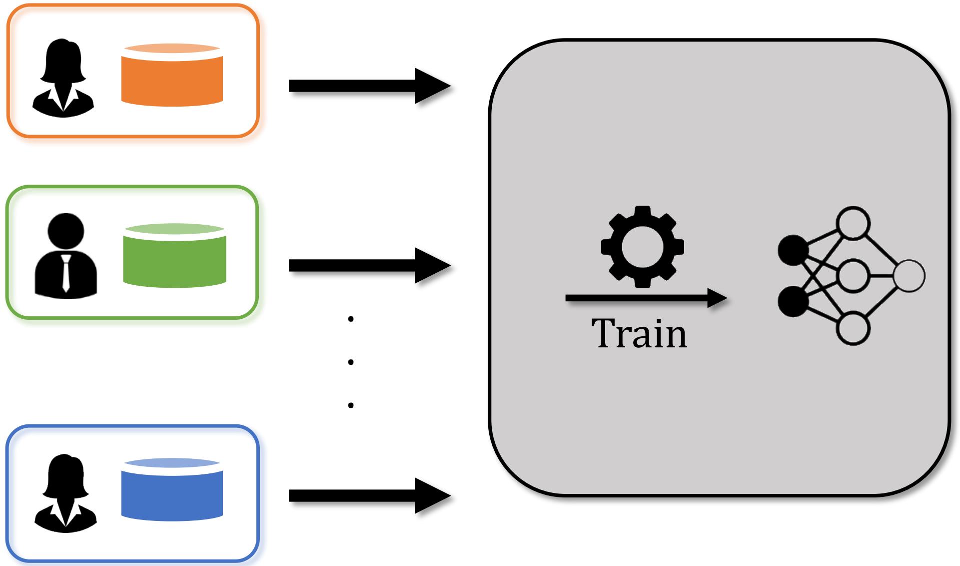
- Clients stage **poisoning attacks**
  - Submit **malformed** data
  - Adversely affect server's analytics

Untrusted Server

Untrusted Client



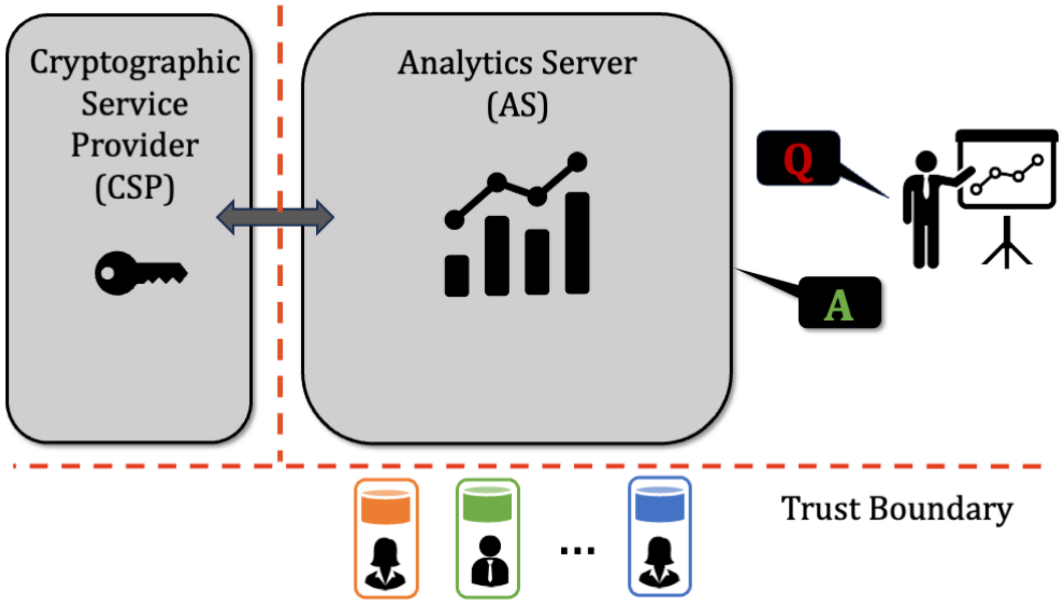
Cryptϵ [**RC**WHMJ'20]



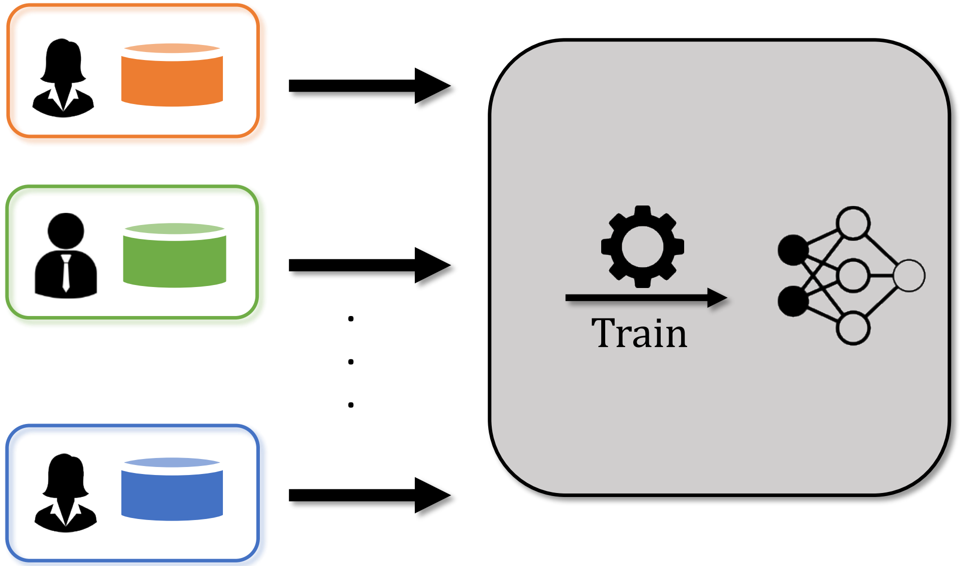
EIFFeL [**RC**GJM'22]

Untrusted Server

Untrusted Client



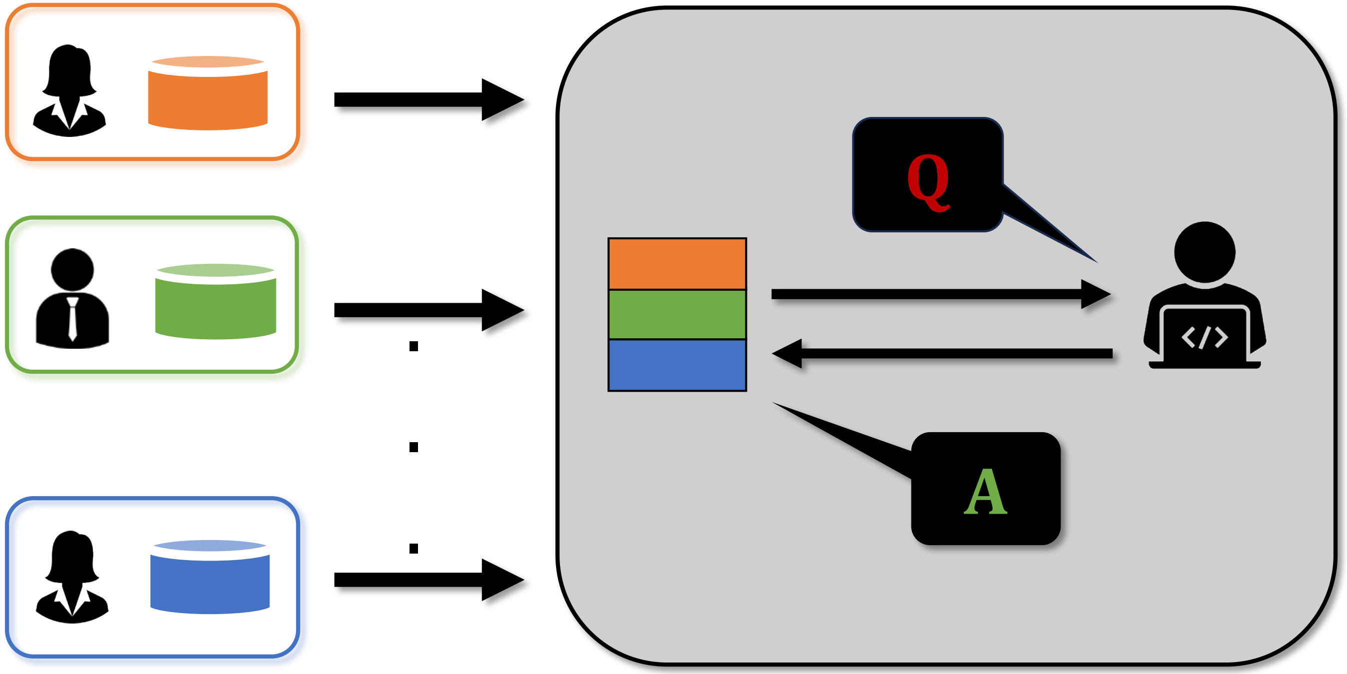
Cryptε [RCWHMJ'20]



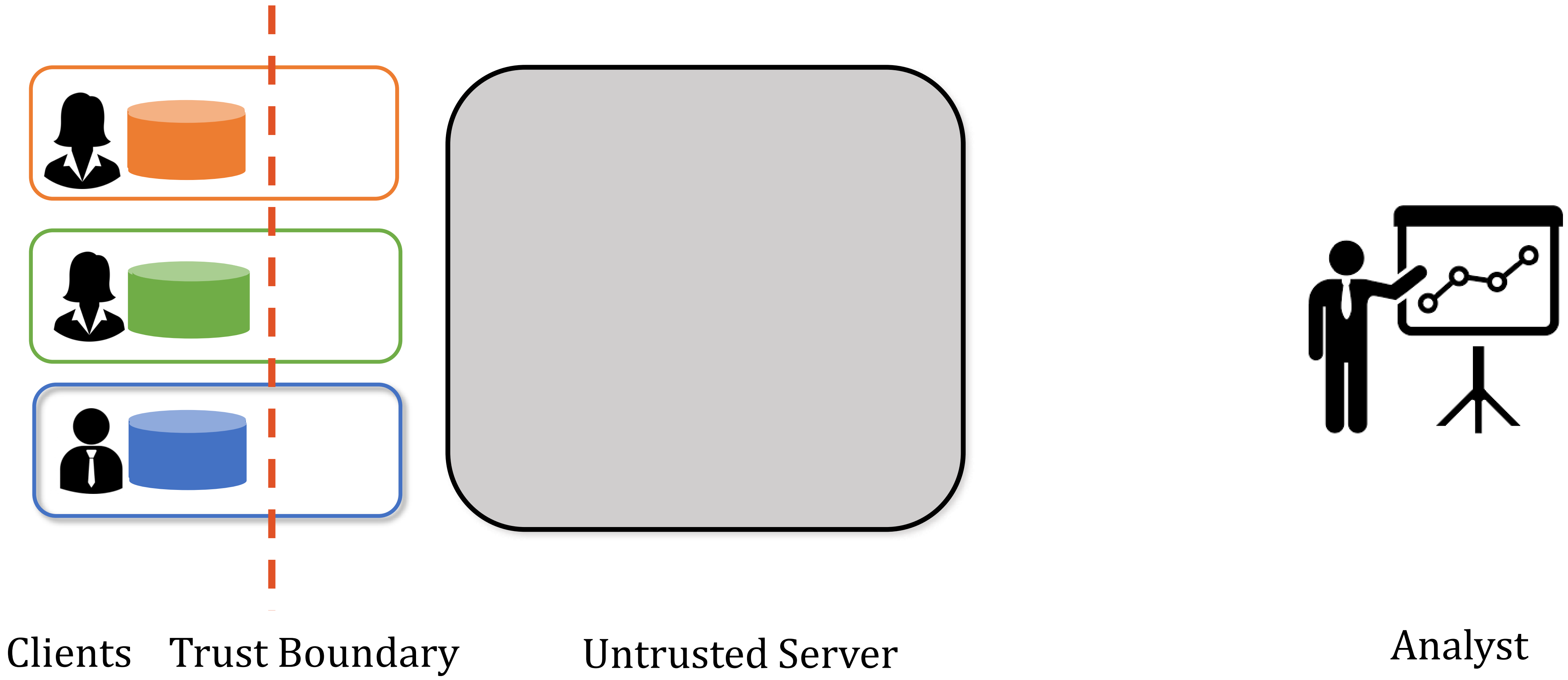
EIFFeL [RCGJM'22]

Cryptϵ: Crypto-Assisted Differential Privacy

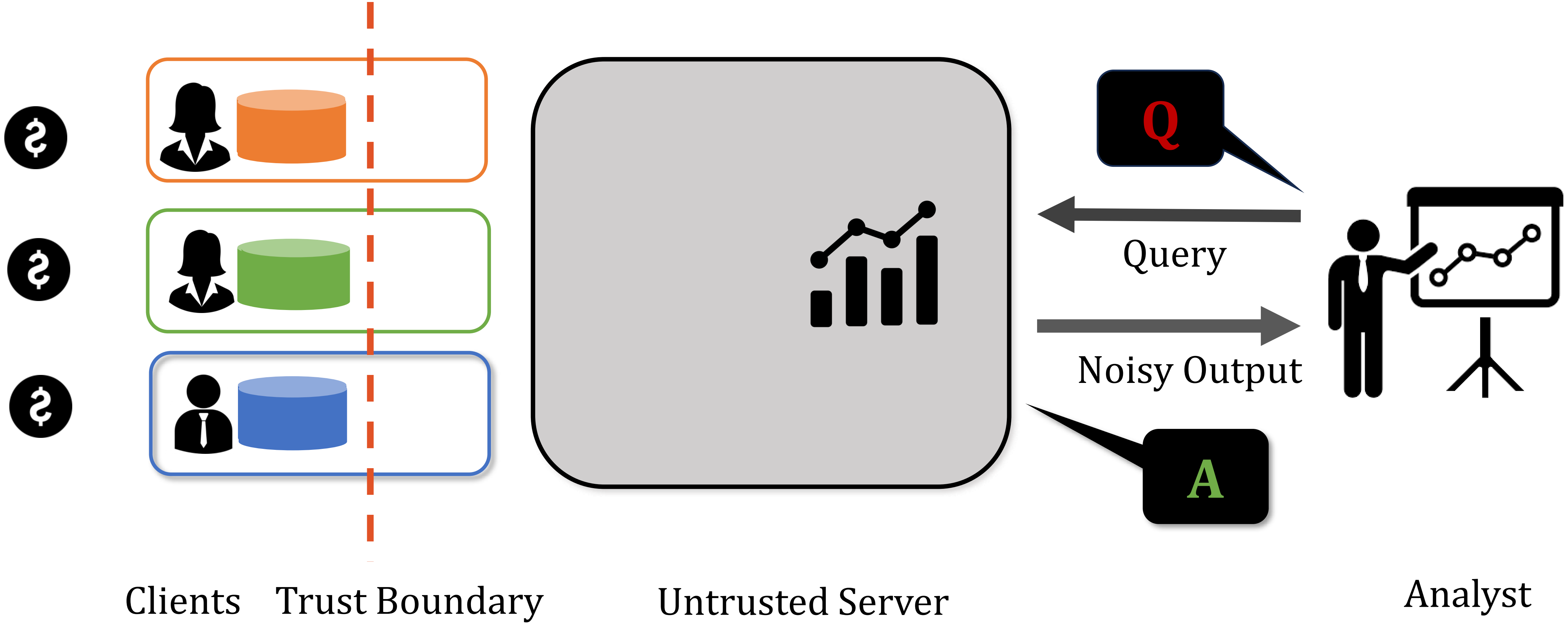
- **Task** – Analytical queries over the joint dataset
  - E.g. query – Count of records of male employees w/ age 45
- We want a DP guarantee on the query responses



Local Differential Privacy



Local Differential Privacy

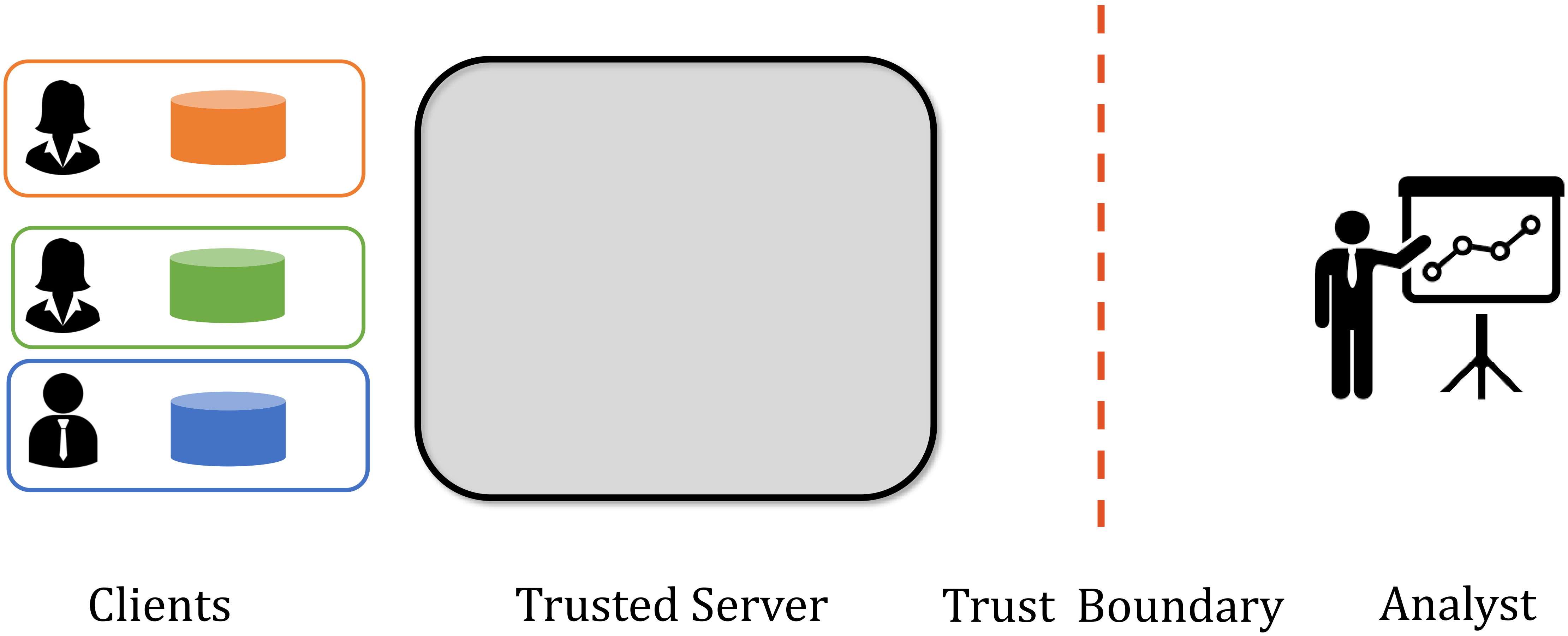


Local Differential Privacy

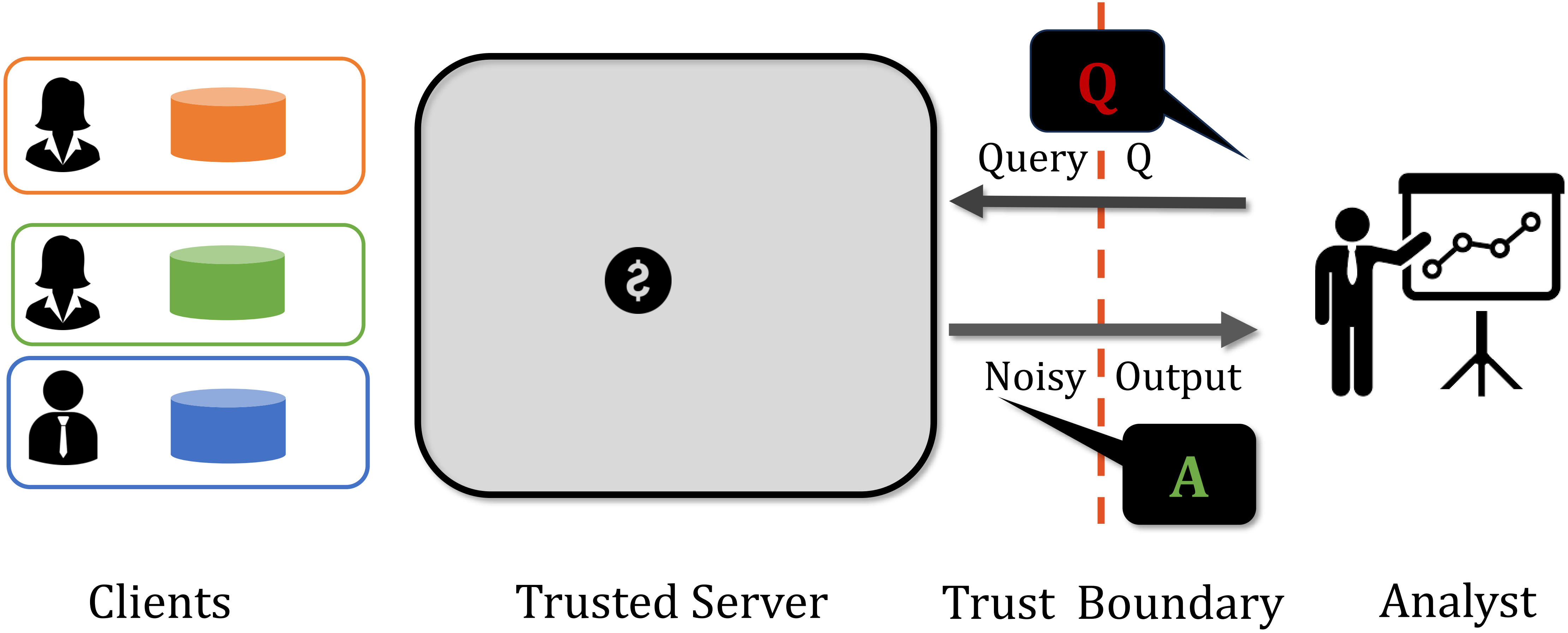
- No trusted server
- Every client adds noise – Server stores **noisy data**
- Low accuracy
  - $\Omega(\sqrt{n}/\epsilon)$ for statistical counting queries, where n is the number of clients

Real-world Deployment

Central Differential Privacy



Central Differential Privacy



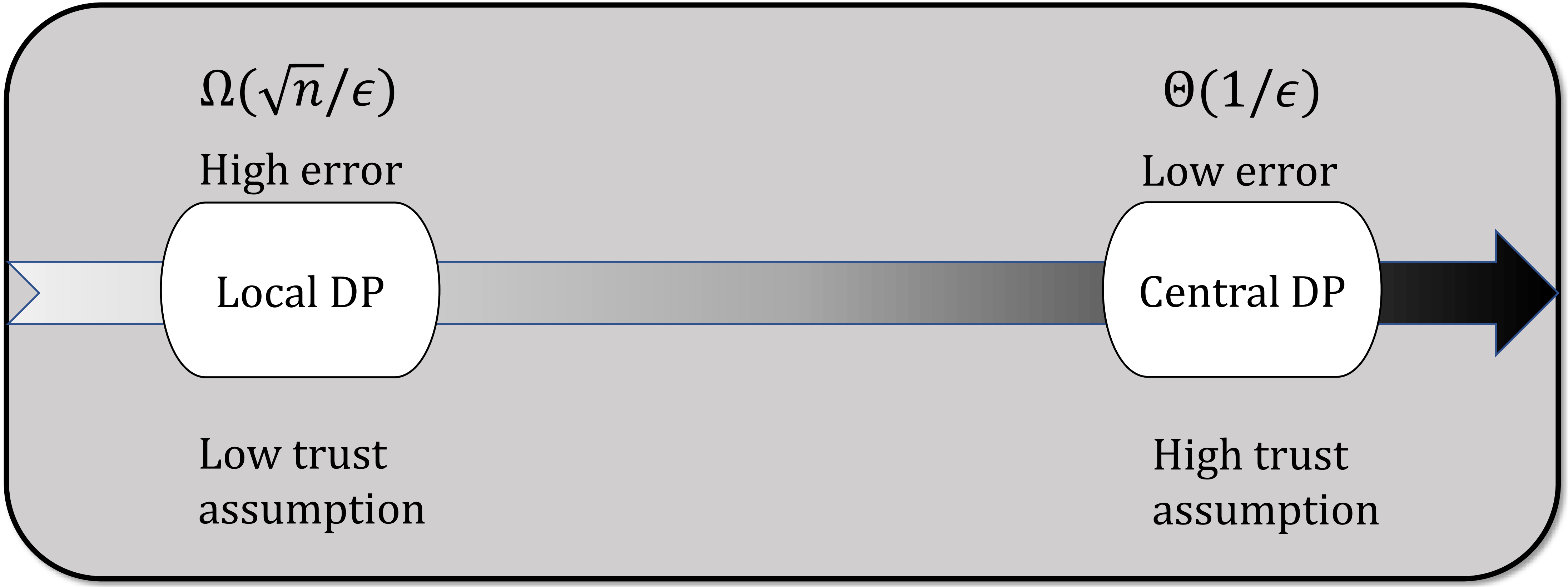
Central Differential Privacy

- The central server
  - is trusted
  - stores the data **in the clear**
  - adds a **single** instance of noise
- Expected error is $\Theta(1/\epsilon)$

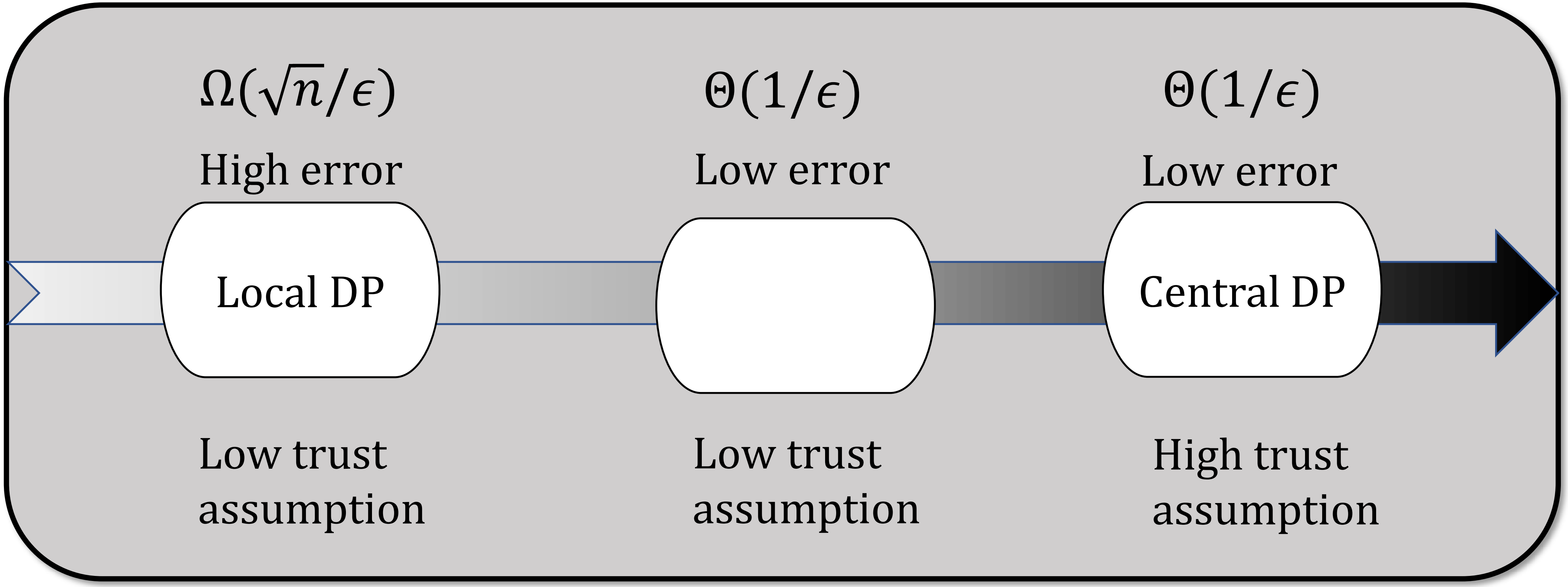
Real-world Deployment

United States® Census 2020

Trust – Accuracy Gap

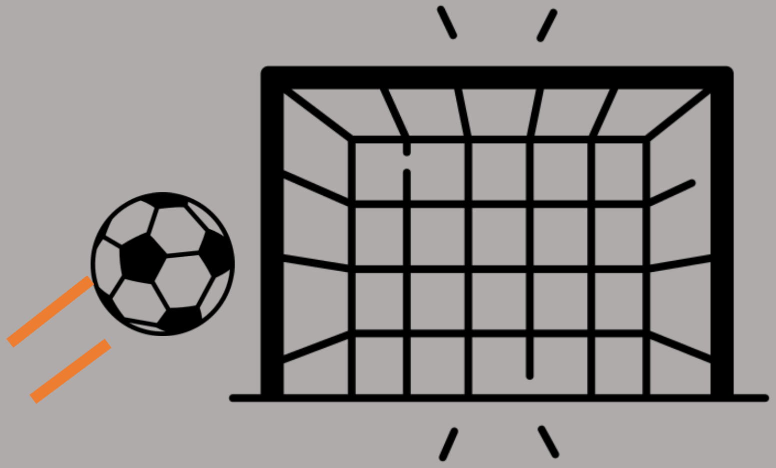


Ideal Goal

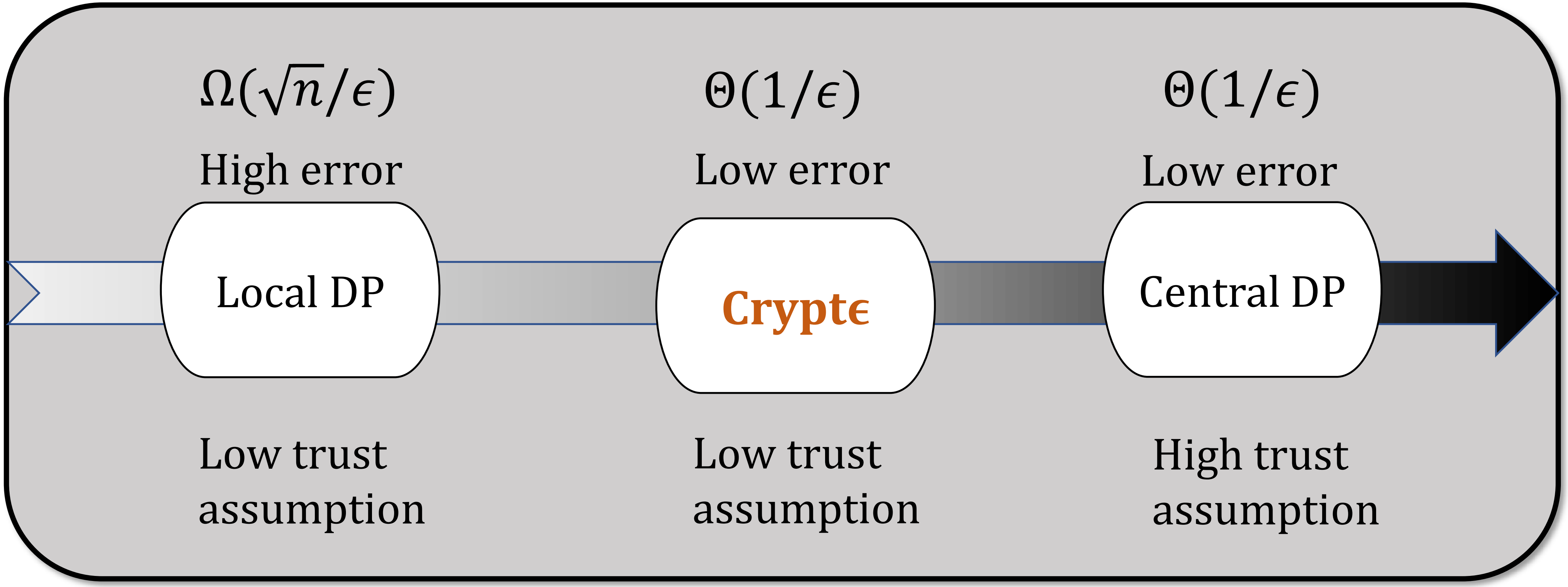

$\Omega(\sqrt{n}/\epsilon)$
High error
Local DP
Low trust assumption
$\Theta(1/\epsilon)$
Low error
Low trust assumption
$\Theta(1/\epsilon)$
Low error
Central DP
High trust assumption

The ideal goal is to achieve

- the same accuracy of central DP
- without any "trusted server" like local DP



Ideal Goal



$\Omega(\sqrt{n}/\epsilon)$
High error
Local DP
Low trust assumption
$\Theta(1/\epsilon)$
Low error
Cryptϵ
Low trust assumption
$\Theta(1/\epsilon)$
Low error
Central DP
High trust assumption

- Cryptϵ achieves this with the help of **cryptographic primitives**
- **First** work to bridge the gap

Challenges in Practice

- **Clients need to be online** – Actively participate in a cryptographic protocol

In theory, the problem can be solved using **off-the-shelf** multi-party computation (MPC) tools

- MPC allows untrusting parties to jointly compute over their secret data

Challenges in Practice

- Clients need to be online
- **Ad-hoc implementation and optimization**

Challenges in Practice

Performance optimization is non-trivial

Accurate Histogram Publication [ZCXMX14]

For $i \in [1, L]$: $c_i' \leftarrow [c_i] + [\eta_i]$, where $\eta_i \sim Lap(\frac{1}{\epsilon_1})$

$([c_{i_1}'], \ldots, [c_{i_L}']) \leftarrow Sort(Threshold([c_1'], \ldots, [c_L']))$

$C \leftarrow Cluster([c_{i_1}'], \ldots, [c_{i_L}'])$

For $C_i \in C$: $[C_i] \leftarrow \Sigma_{c_j \in C_i}[c_i']/|C_i|$

For $i \in [1, L]$:$[c_L''] \leftarrow [C_i] + [\eta_i]/|C_i|$, where $\eta_i \sim Lap\left(\frac{1}{\epsilon_2}\right)$

Release $(c_i'', \ldots, c_L'')$

Private

Clear

Challenges in Practice

Performance optimization is non-trivial

- EMP toolkit takes 13 min for a dataset of size 33K (compute all steps securely)
- Optimized version requires **3.4X less** time (compute only red steps securely)

Accurate Histogram Publication [ZCXMX14]

For $i \in [1, L]$: $c_i' \leftarrow [c_i] + [\eta_i]$, where $\eta_i \sim Lap(\frac{1}{\epsilon_1})$

$([c'_{i_1}], \ldots, [c'_{i_L}]) \leftarrow Sort(Threshold([c'_1], \ldots, [c'_L]))$

$C \leftarrow Cluster([c'_{i_1}], \ldots, [c'_{i_L}])$

For $C_i \in C$: $[C_i] \leftarrow \Sigma_{c_j \in C_i}[c'_i]/|C_i|$

For $i \in [1, L]$:$[c''_L] \leftarrow [C_i] + [\eta_i]/|C_i|$, where $\eta_i \sim Lap\left(\frac{1}{\epsilon_2}\right)$

Release $(c''_i, \ldots, c''_L)$

Private

Clear

Figuring out what operations can be done in the clear is subtle

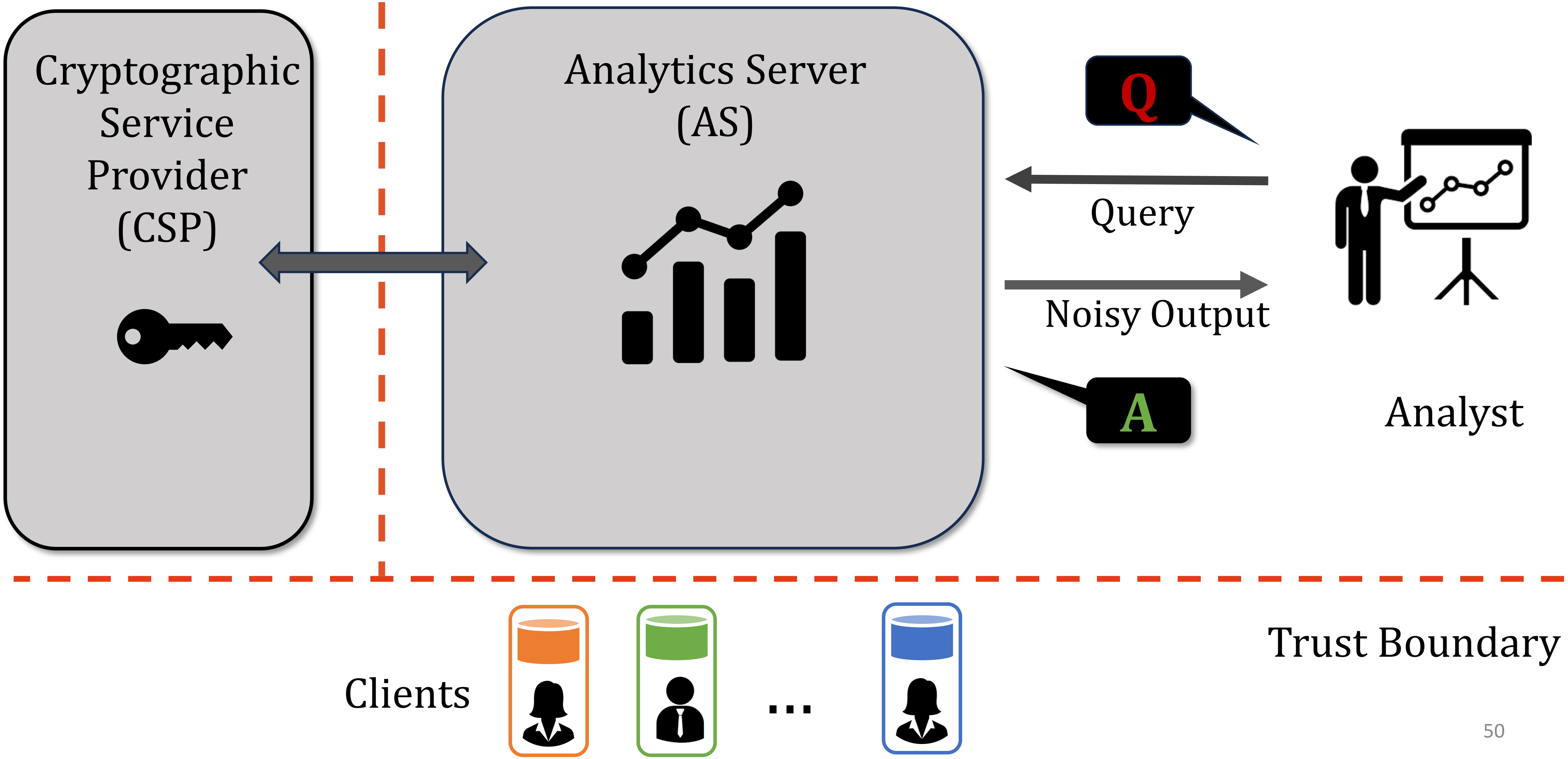
Challenges in Practice

- Clients need to be online
- Ad-hoc implementation and optimization
- **Tricky privacy proofs** – Hybrid solutions are error-prone

Cryptϵ Design Highlights

- **Rethink computation model** – Split trust into **2-server** model

Cryptϵ



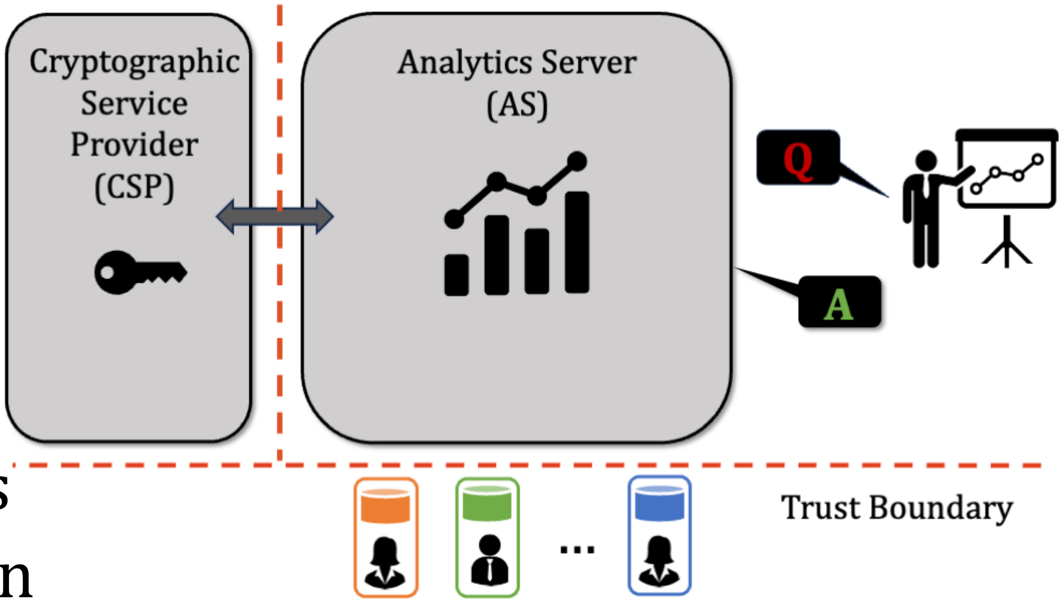
Cryptϵ Threat Model

- **Semi-honest** threat model
  - Every party follows protocol, but
  - Tries to learn **more information** from protocol transcript
- AS and CSP are **non-colluding**

Cryptϵ Computation Model

- AS is an extension of the analyst
  - E.g. AS is Company Z
- CSP is a third-party entity
  - E.g. CSP is Symantec/Divvi up

- AS collects **encrypted data** from clients
- DP program execution happens between AS and CSP only



Challenges in Practice

- **Clients need to be online**
- Ad-hoc implementation and optimization
- Tricky privacy proofs

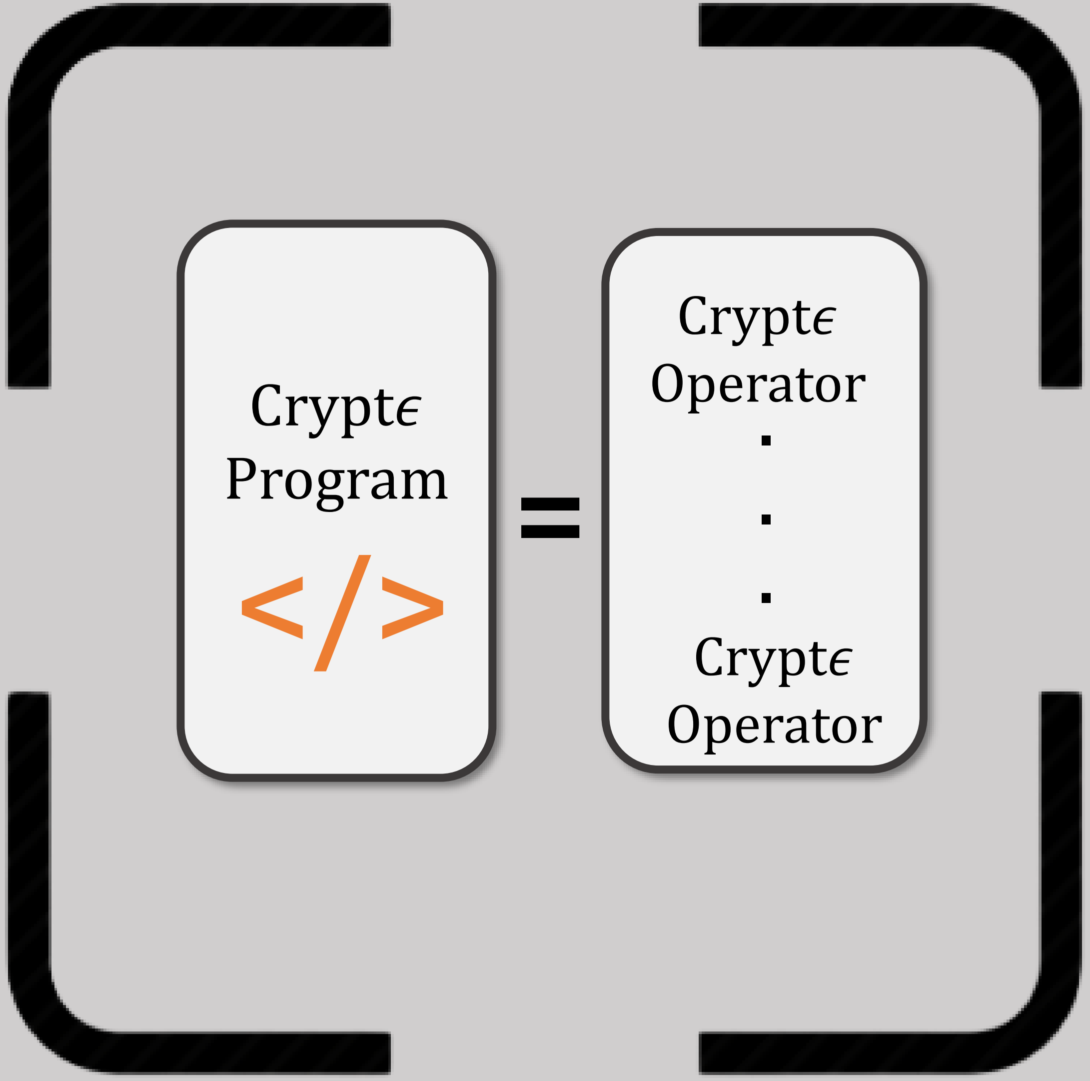
Cryptϵ Design Highlights

- Rethink computation model
- **Programming framework** – Author logical DP programs using high-level operators

Cryptϵ Programming Framework

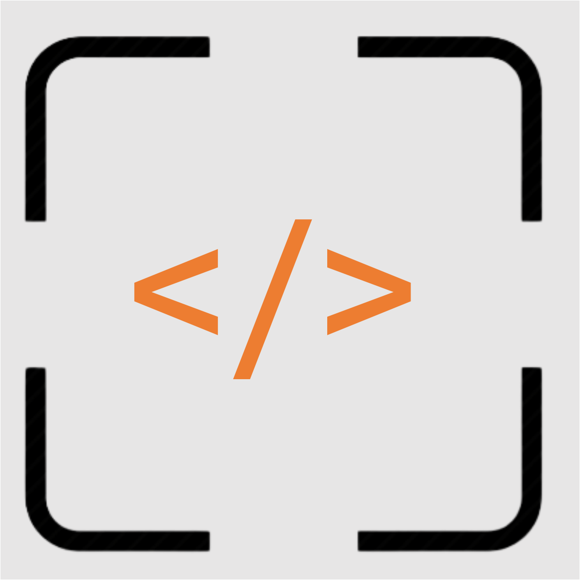
Supports a fixed set of operators

Cryptϵ Programming Framework



Cryptϵ Programming Framework

- Operators inspired by relational algebra, e.g.:
  - Filter $\sigma_{\phi}(\,)$
  - GroupBy $\gamma_{A}^{count}(\,)$
- Optimized implementation for every program
  - Clear categorization of operators by **type**
  - **Built-in optimizations** for each operators



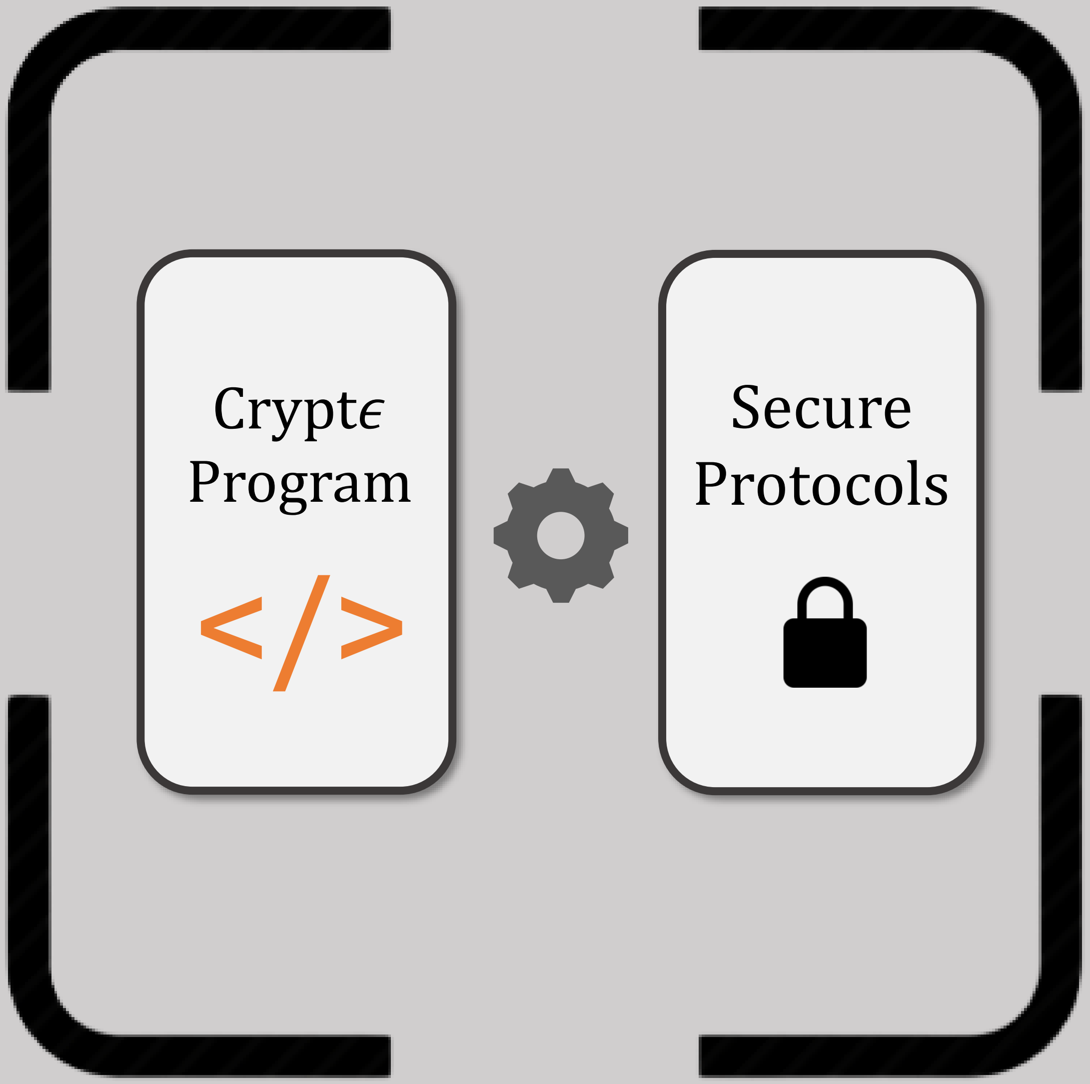
Challenges in Practice

- Clients need to be online ✓
- **Ad-hoc implementation and optimization** ✓
- Tricky privacy proofs

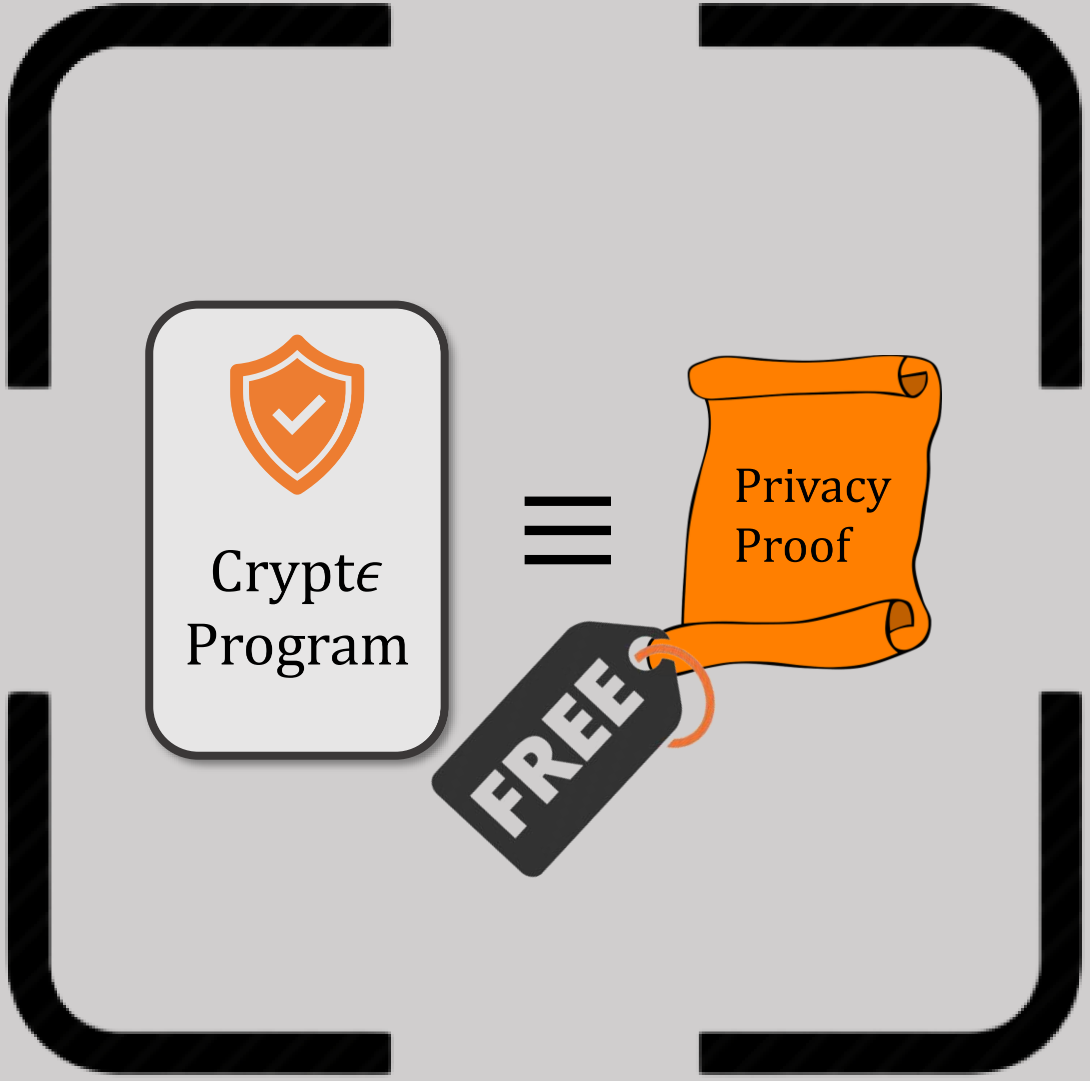
Cryptϵ Design Highlights

- Rethink computation model
- Programming framework
- **Separation** of logical and physical layers

Cryptϵ Programming Framework



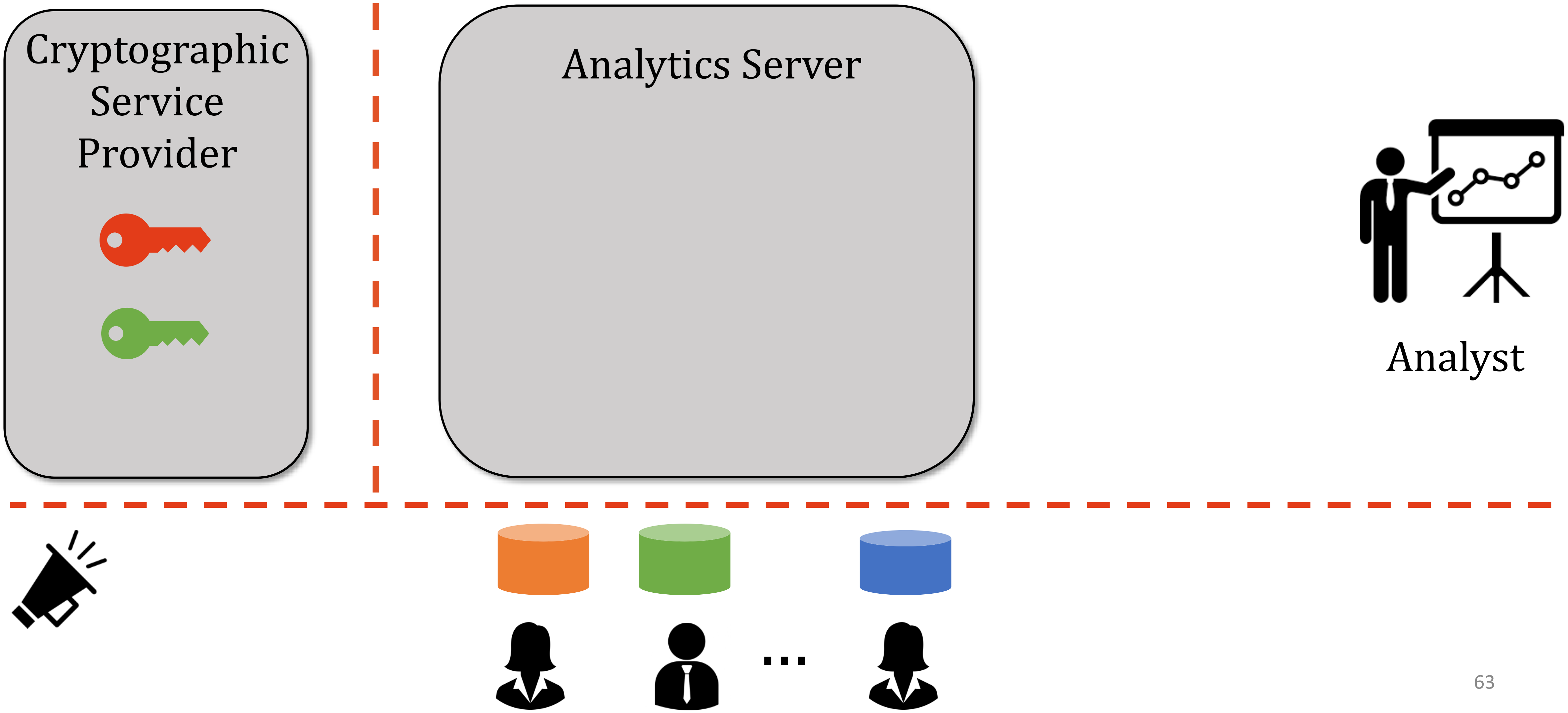
Cryptϵ Programming Framework



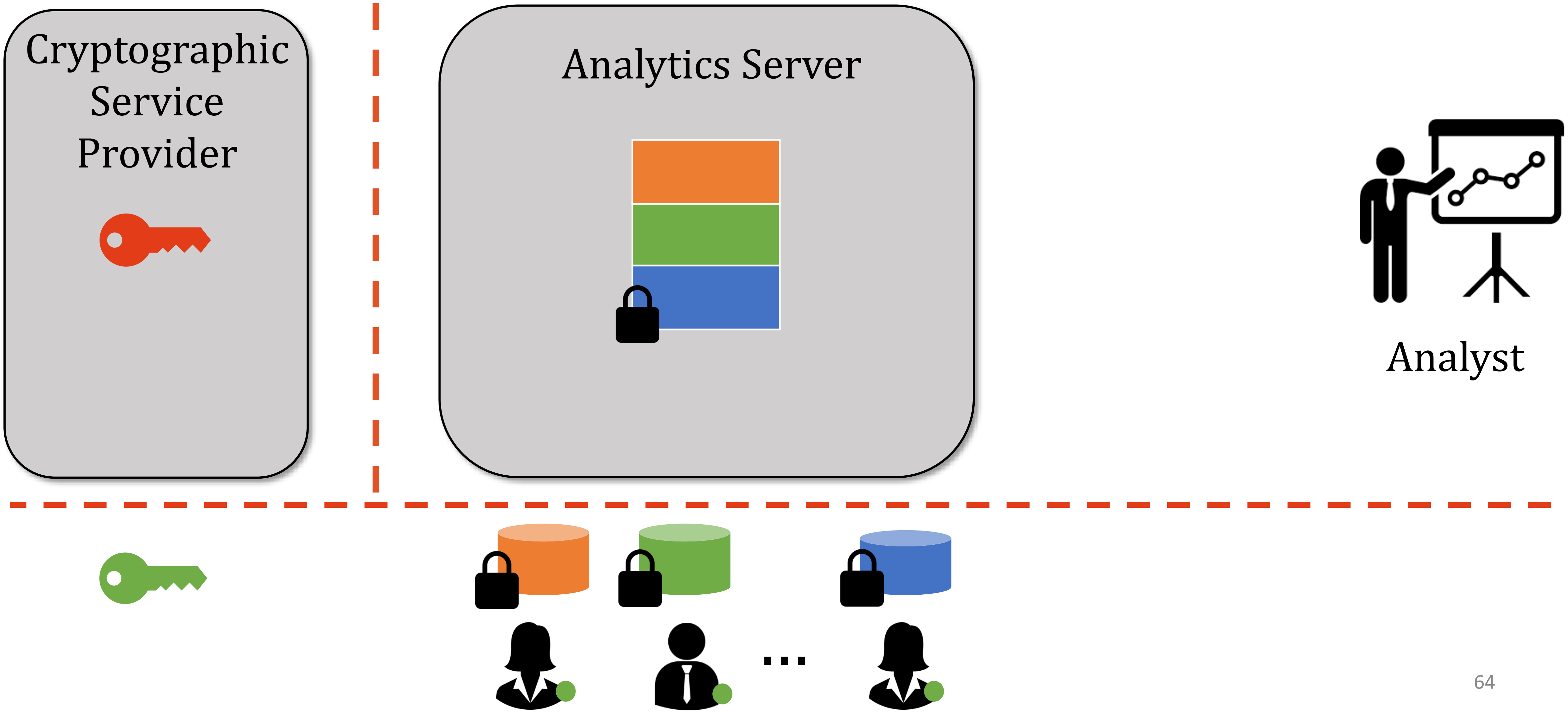
Challenges in Practice

- Clients need to be online
- Ad-hoc implementation and optimization
- **Tricky privacy proofs**

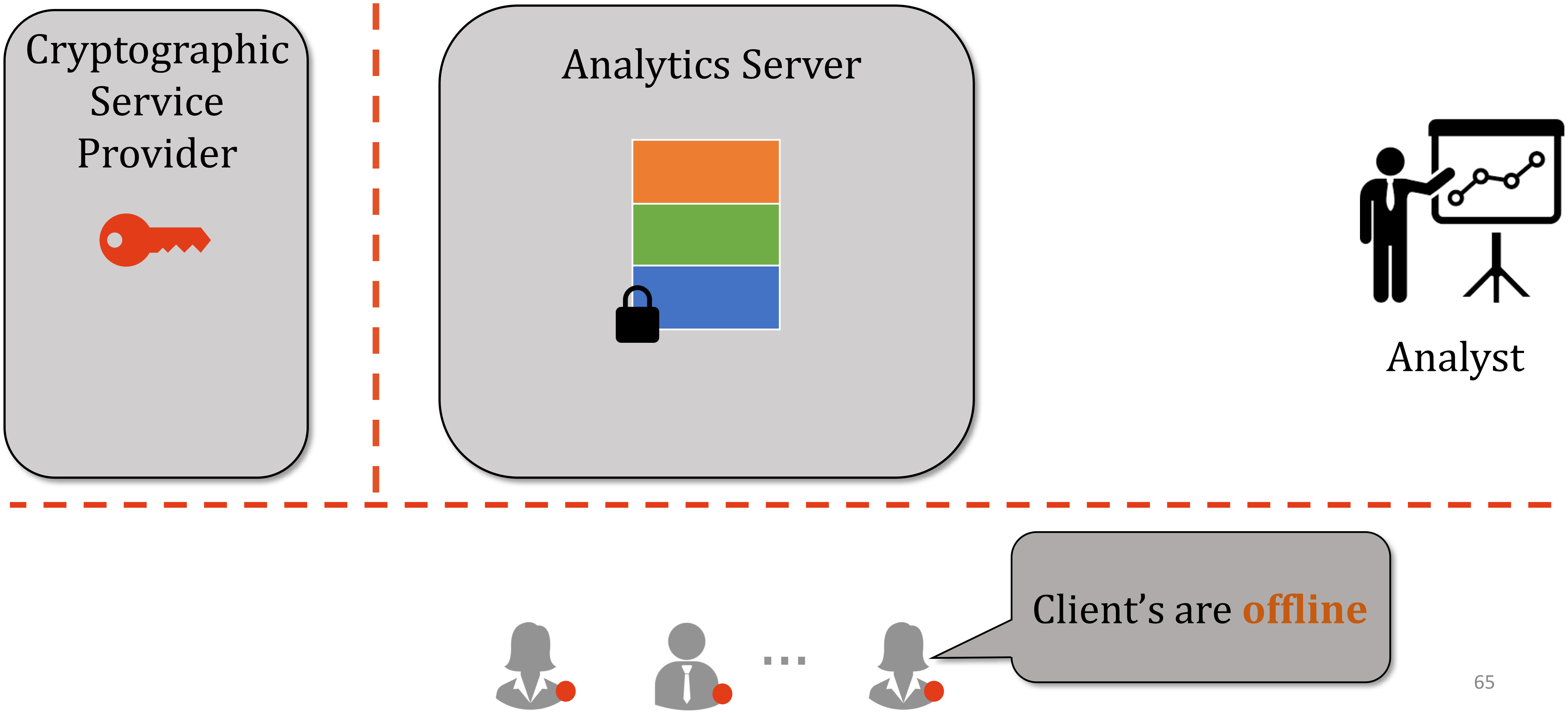
Cryptϵ – Bird's Eye View



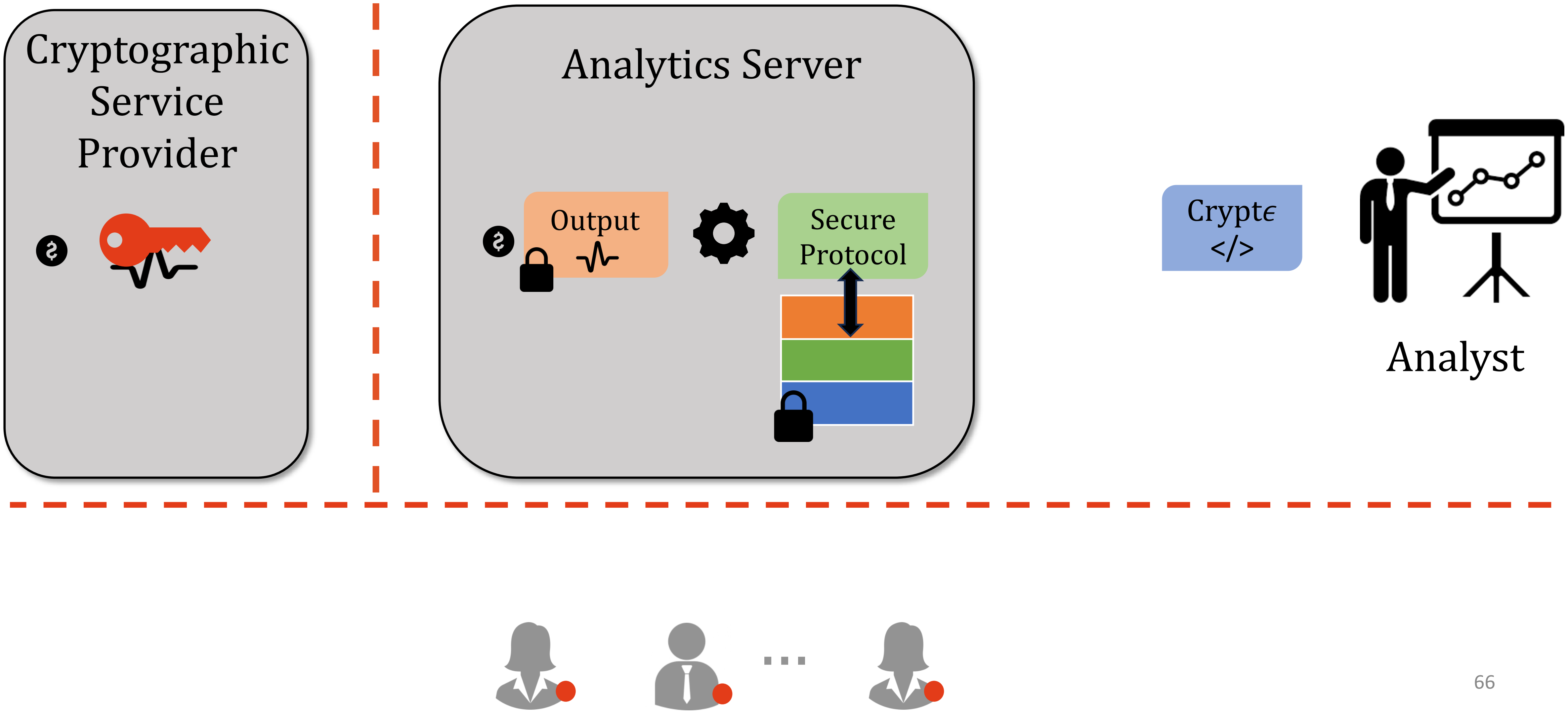
Cryptϵ – Bird's Eye View



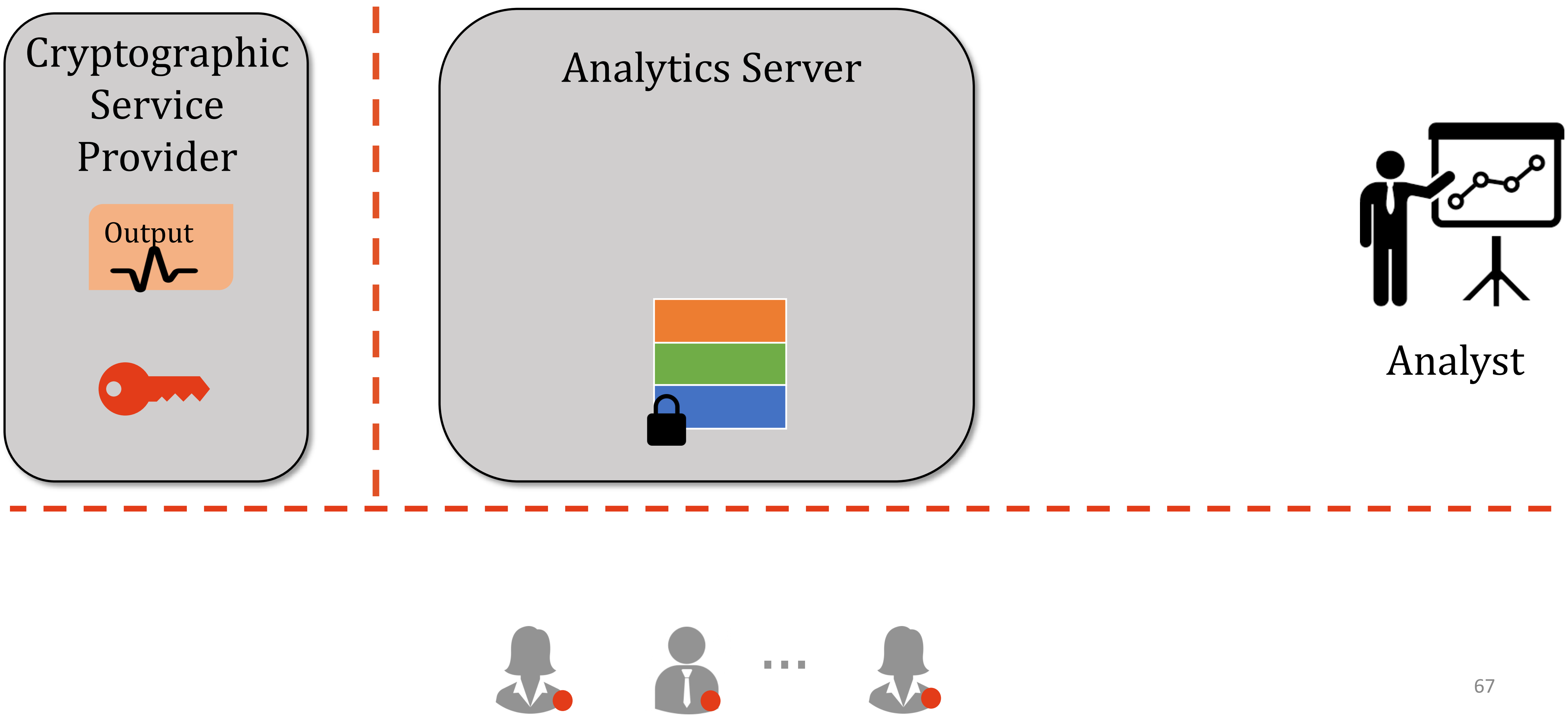
Cryptϵ – Bird's Eye View



Cryptϵ – Bird's Eye View



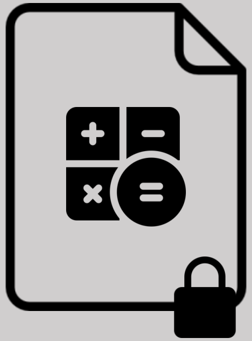
Cryptϵ – Bird's Eye View



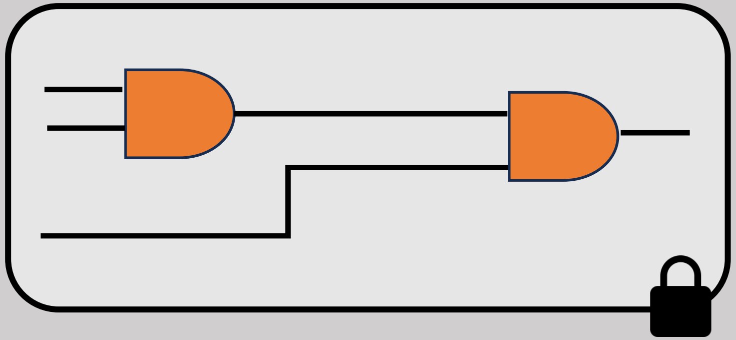
- Cryptϵ adds **two instances** of noise independent of the data size
- Cryptϵ achieves **constant** error bounds like central DP

Implementation

- Labeled homomorphic encryption [BCF17]
  - Linear operations and 1 multiplication directly on encrypted data
  - **New cryptographic tool** – Extension to support n-way multiplications



- Garbled circuits
  - Boolean operations directly on encrypted data



Privacy Guarantee

Thm. (Informal) Every Cryptϵ program satisfies ϵ − SIM-CDP [MPRV09]

- View of the AS (CSP) is **computationally indistinguishable** from that of executing the program in the central DP model

Cryptϵ Extension

- Malicious AS
  - Message authentication code (MAC)
  - Designated verifier non-interactive ZKP
- Malicious CSP
  - Trusted execution environment (TEE)

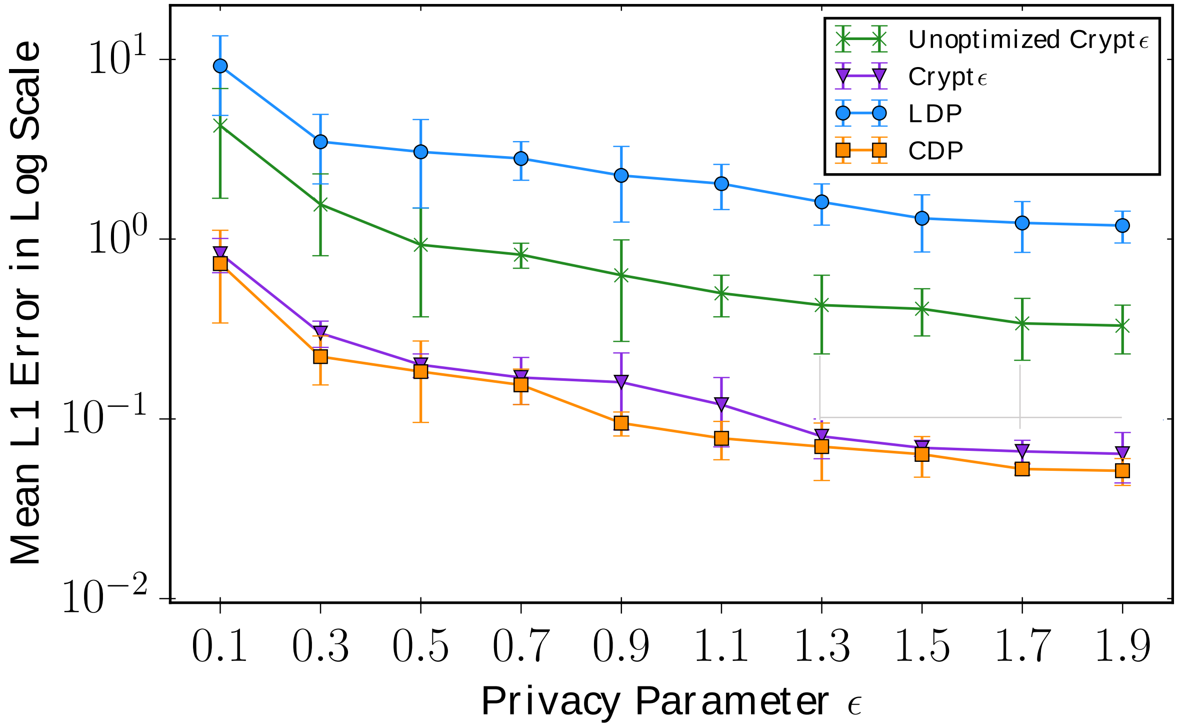
Evaluated Cryptϵ Programs

- **Schema** – $Age, Sex, Country, Race$

- **P1** – Outputs the c.d.f of Age with domain [1,100]
- **P2** – Outputs the marginal over attributes $Race$ and Sex
- **P3** – Counts the number of Age values w/ $\geq$200 records
- **P4** – Counts the female employees of Canada w/ Age 30
  - $\hat{c} \leftarrow Lap_{\epsilon,\Delta=1}\left(count\left(\sigma_{A=30 \wedge S=F \wedge C=Canada}\left(\pi_{A,S,N}\left(\tilde{D}\right)\right)\right)\right)$

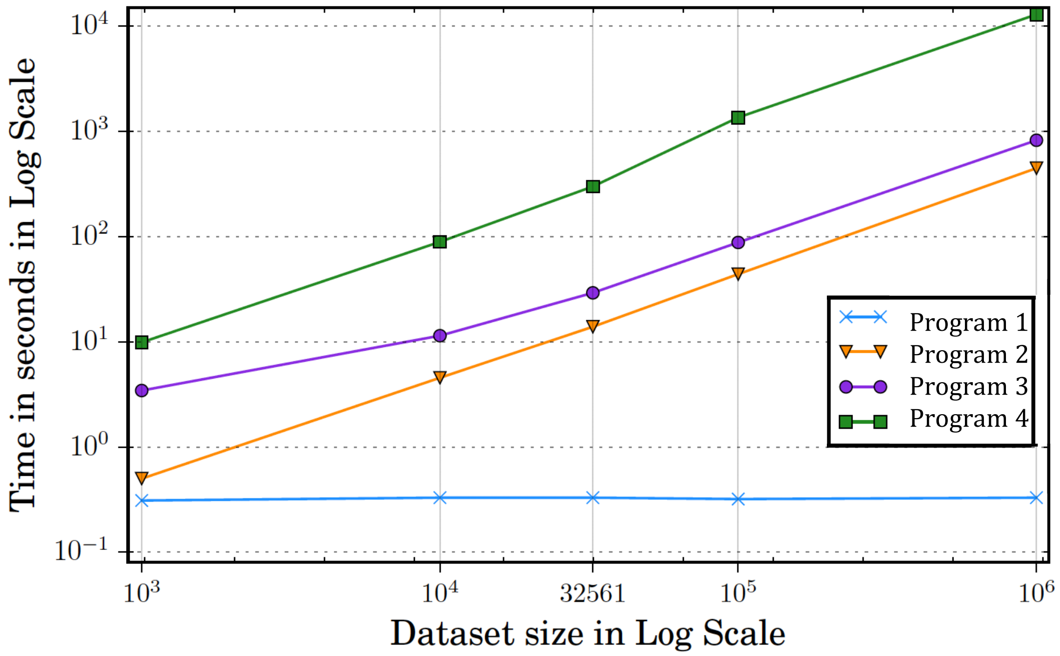
Accuracy

- **Same order of magnitude of error** as that of central DP



Performance

- A large class of programs run <**3.5 hrs** for dataset size **1M**
- Cost **scales linearly** with dataset size



Performance

- A large class of programs run <3.5 hrs for dataset size **1M**
- Cost **scales linearly** with dataset size

- Novel **crypto-engineering** optimizations improve the performance by **41X to 5667X**

Time (s)	Program			
	1	2	3	4
Unoptimized Cryptϵ	1757	13653	1201	30698
Cryptϵ	0.31	14	29	300
Speedup	5667×	982×	41×	102×

Performance

- A large class of programs run <**3.5 hrs** for dataset size **1M**
- Cost **scales linearly** with dataset size

- **Novel DP index optimization** improves the performance by **41X**

Time (s)	Program			
	1	**2**	**3**	**4**
Unoptimized Cryptϵ	1757	13653	1201	30698
Cryptϵ	0.31	14	29	300
Speedup	5667×	982×	41×	102×

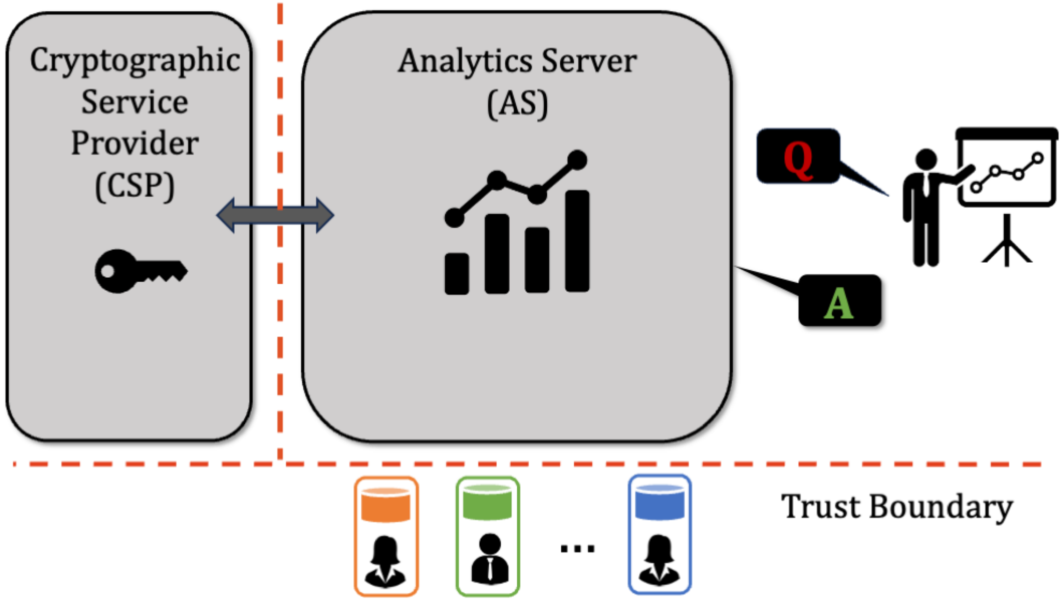
Key Takeaway

Can **non-experts** perform **high accuracy** differentially private query analytics in the decentralized setting?

Yes, Cryptε enables this w/ cryptographic primitives!

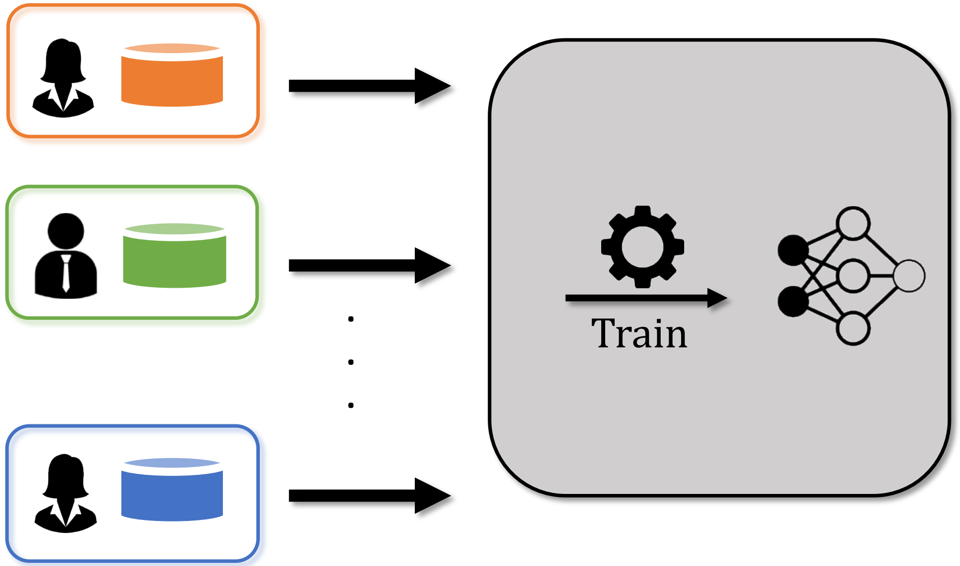
Challenge	Solution
Online clients	2-Server model
Ad-hoc per query optimized implementation	• Logical programming framework • New cryptographic and DP primitives
Tricky privacy proofs	Provably private by construction

Untrusted Server



Cryptϵ [RCWHMJ'20]

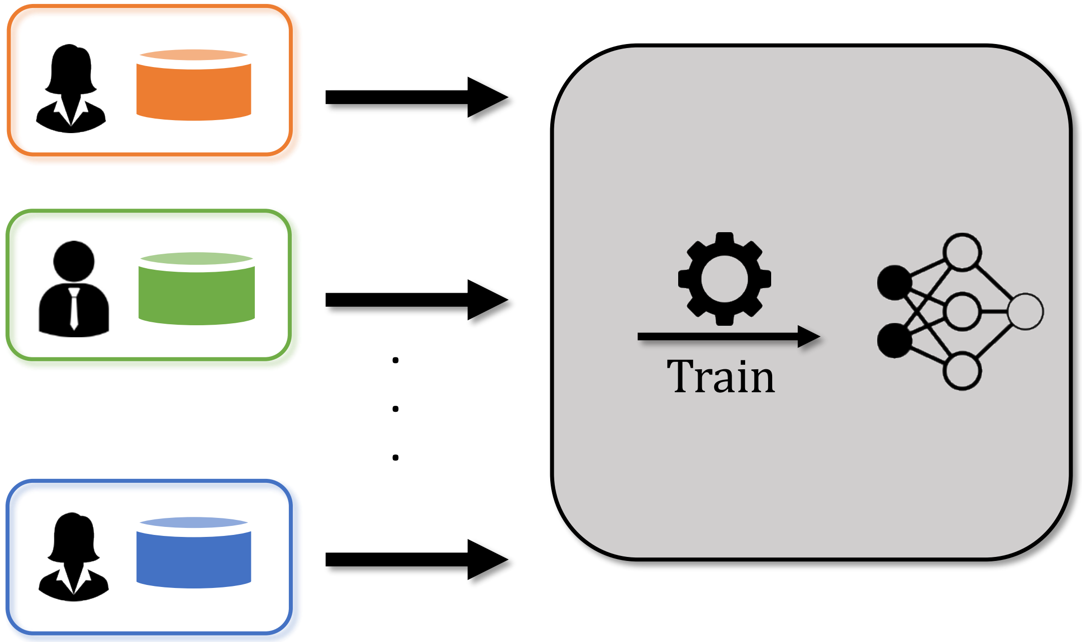
Untrusted Client



EIFFeL [RCGJM'22]

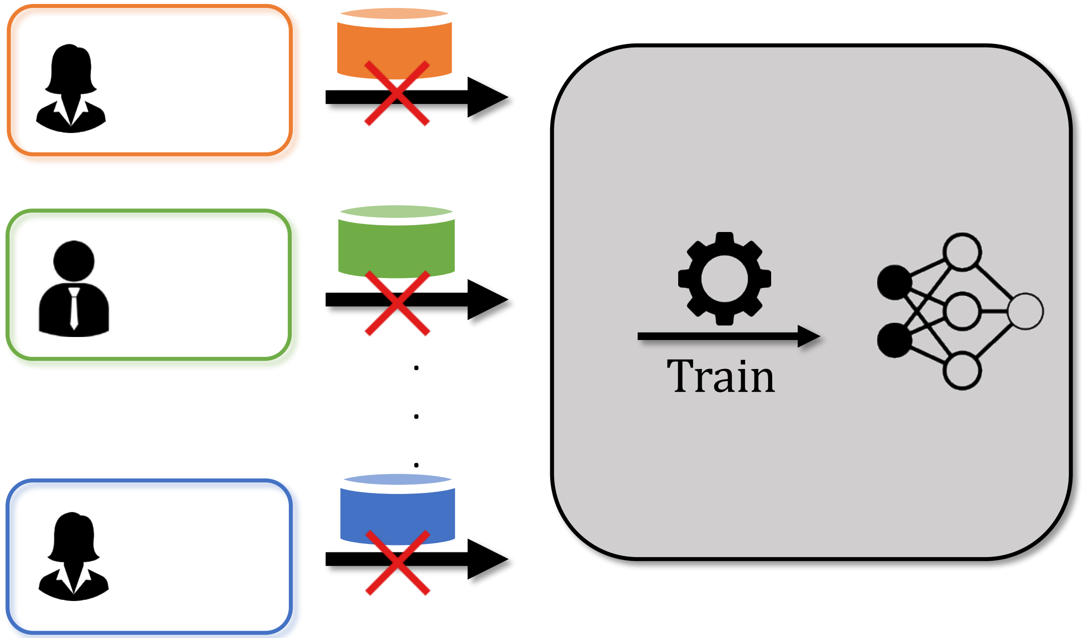
EIFFeL – Ensuring Integrity for Federated Learning

- **Task** – Train a ML model
- Called federated learning
- Decentralized learning
  - Multiple clients
  - Coordinated by a server
- Collaboratively train a model over joint dataset



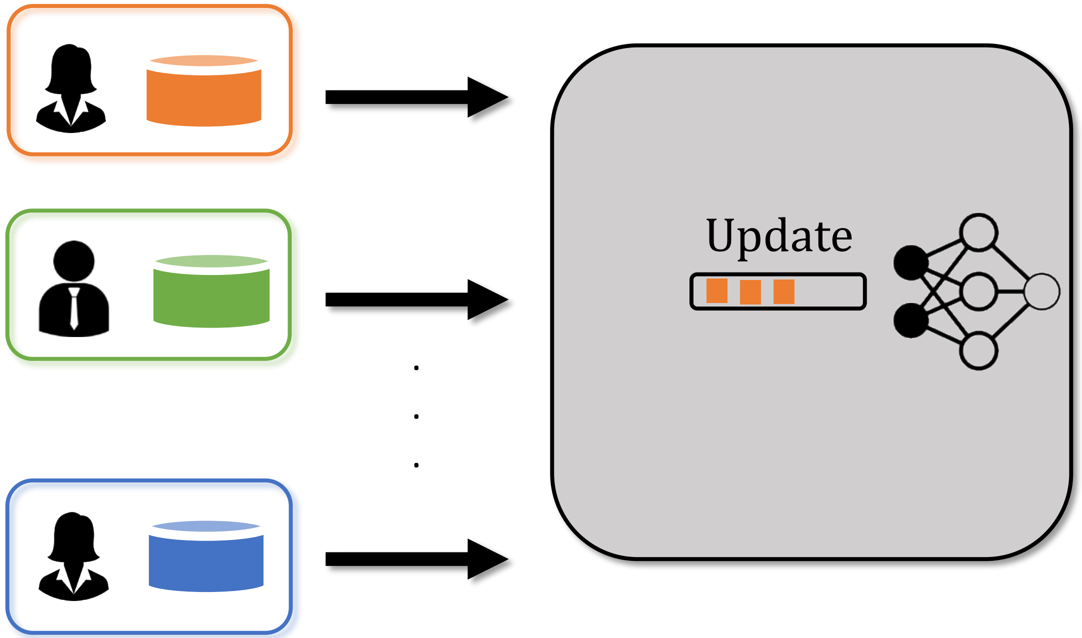
Federated Learning

- Client's raw data is stored **locally**
  - No centralized pooling
- Focused updates (gradients) for immediate aggregation
- Aggregate is used to train a global model
- Iterative training process



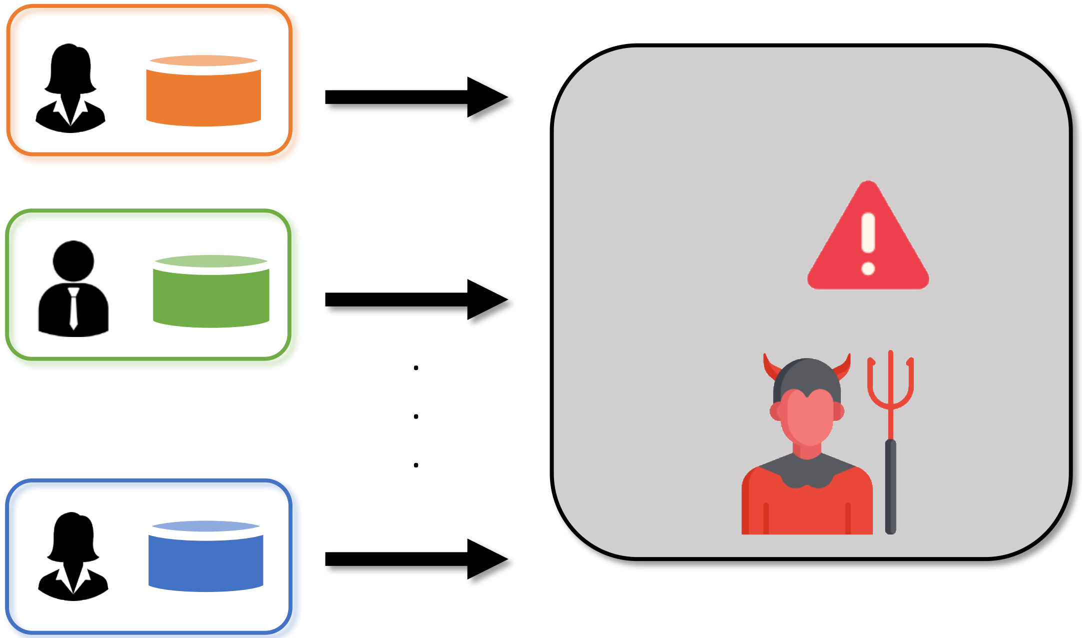
Federated Learning

- Client's raw data is stored **locally**
  - No centralized pooling
- Focused updates (gradients) for immediate **aggregation**
- Aggregate is used to train a global model
- Iterative training process



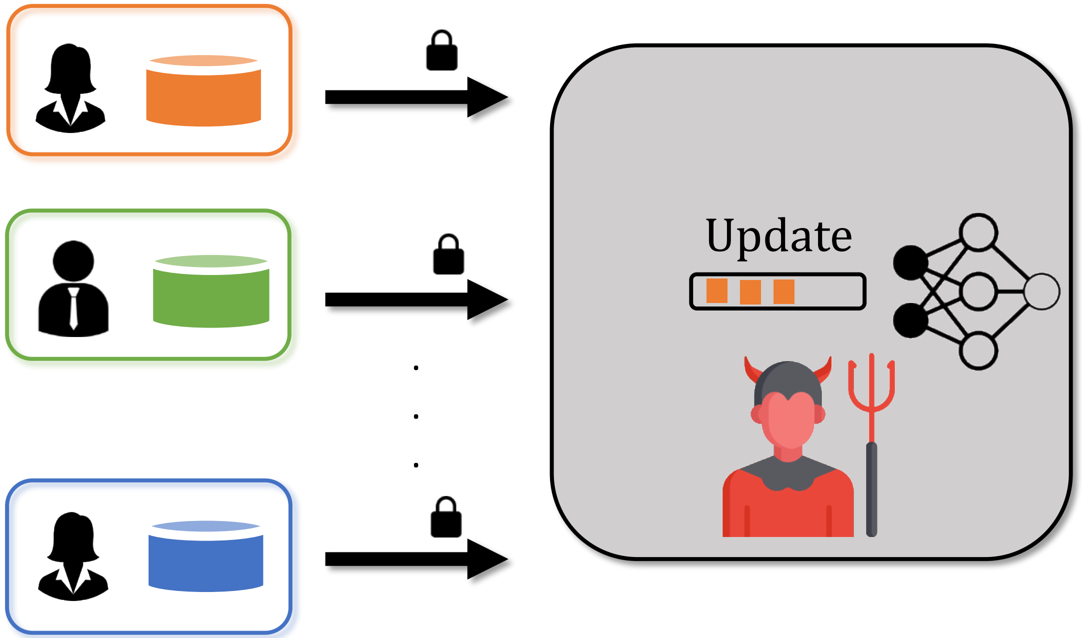
Threat Model – Server

- Server is untrusted
- Decoupling training from data pooling is not enough for privacy
- Inference attacks on just updates
  [BDFKR18][MSCS19][ZLH19] [NSH19] [YMVAKM21]



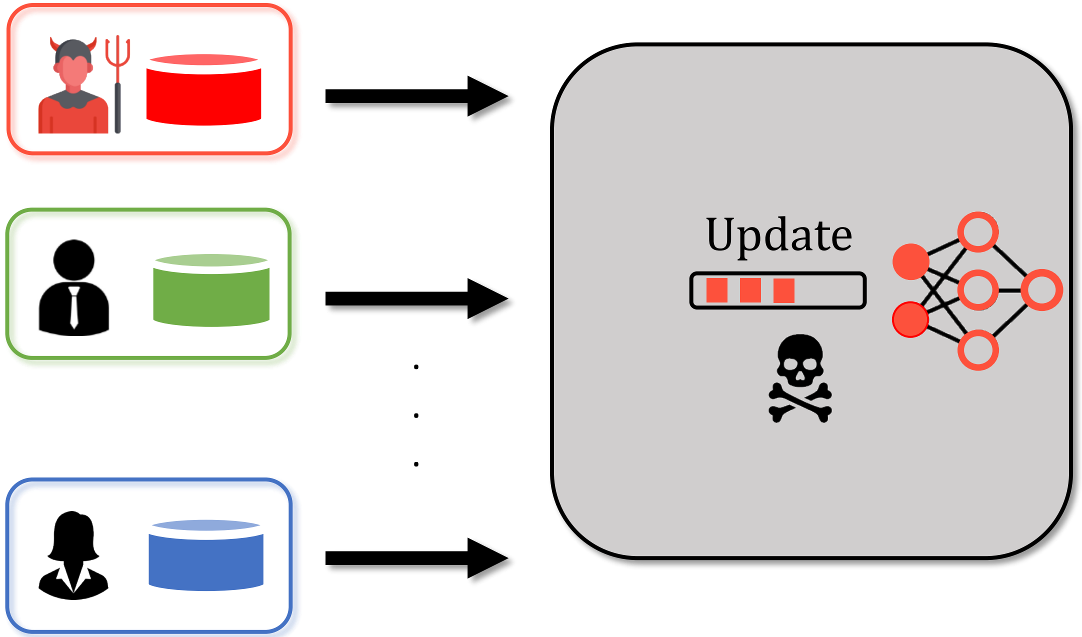
Threat Model – Server

- Ensure input privacy
- Secure aggregation of updates
- Cryptographic protocol
  - Aggregates (sums) **masked** updates
  - Only **final aggregate** revealed in the clear



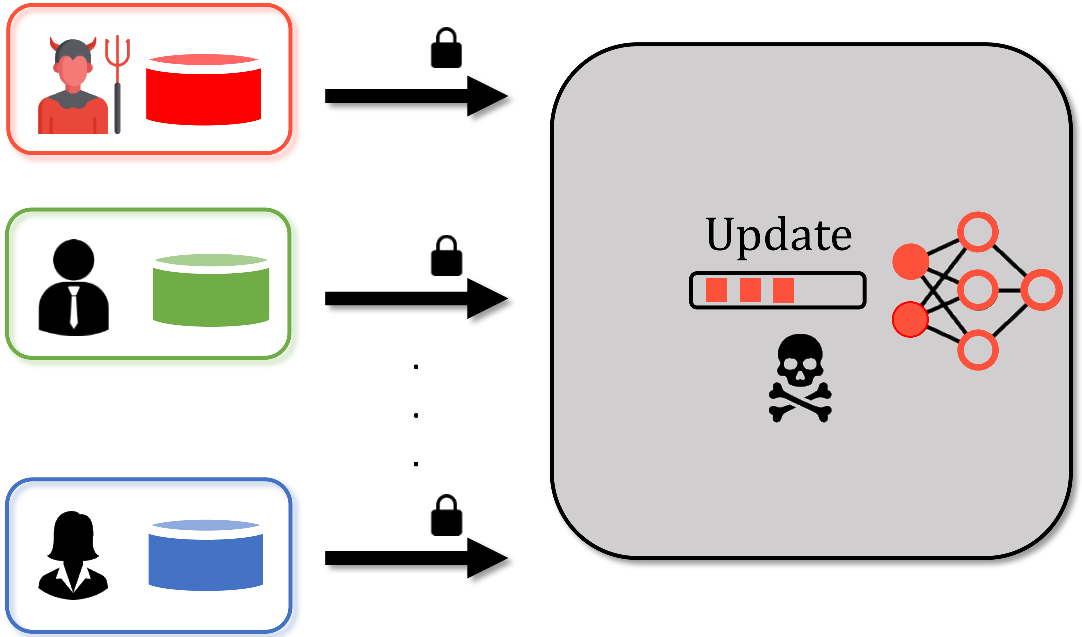
Threat Model – Client

- Distributed nature
  - Vulnerable to **poisoning attacks**
- Malformed update adversely affect model accuracy
- Compromises integrity of training



Threat Model – Client

- Ensure updates are well-formed – **Input integrity**
- Detect syntactic inaccuracies
  - E.g.: Inputs are norm-bounded
- Challenge – Inputs are masked due to secure aggregation



How to ensure integrity **without** violating privacy?

EIFFeL – **First** general framework for **both** privacy and integrity

Security Goal

- **Malicious** threat model
- Adversary
  - m malicious clients
  - Server
- Can behave arbitrarily
- Could collude together

Challenges in Practice

- **Detection vs removal** of malformed inputs – Just detection is easier, but vulnerable to **DoS** attacks

Challenges in Practice

- Detection vs removal of malformed inputs
- **Flexibility** in integrity checks – Off-the-shelf tools (zero knowledge proof) are expensive

Challenges in Practice

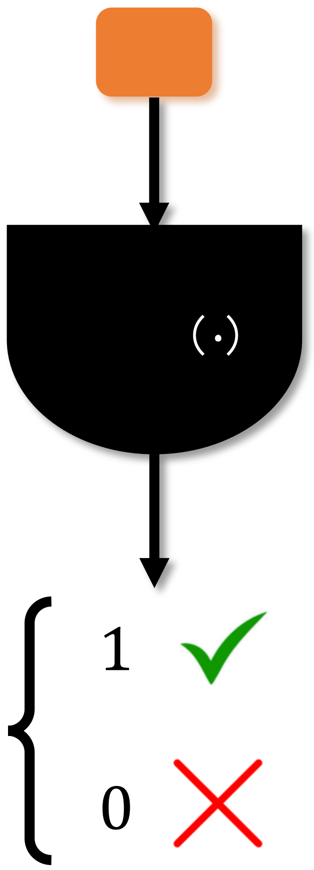
- Detection vs removal of malformed inputs
- Flexibility in integrity checks
- **Compatibility** with real-world deployment – Single server

EIFFeL Design Highlights

- **New protocol for FL** – Secure aggregation with verified inputs

Secure Aggregation with Verified Inputs

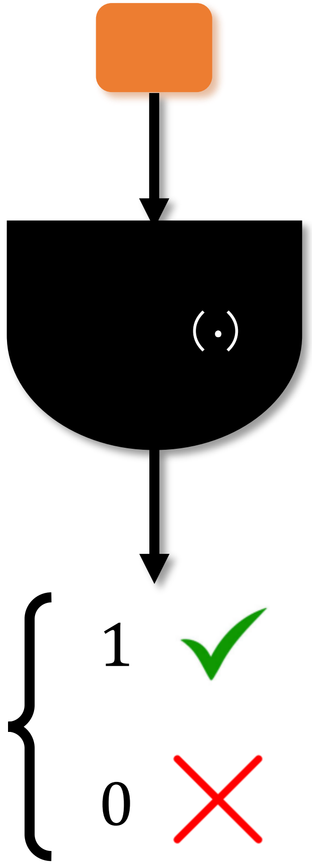
- A public **validation predicate**, $Valid(\cdot)$
- An input u is valid, i.e., passes the integrity check if $Valid(u) = 1$
- Defines a **syntax** for the inputs



Secure Aggregation with Verified Inputs

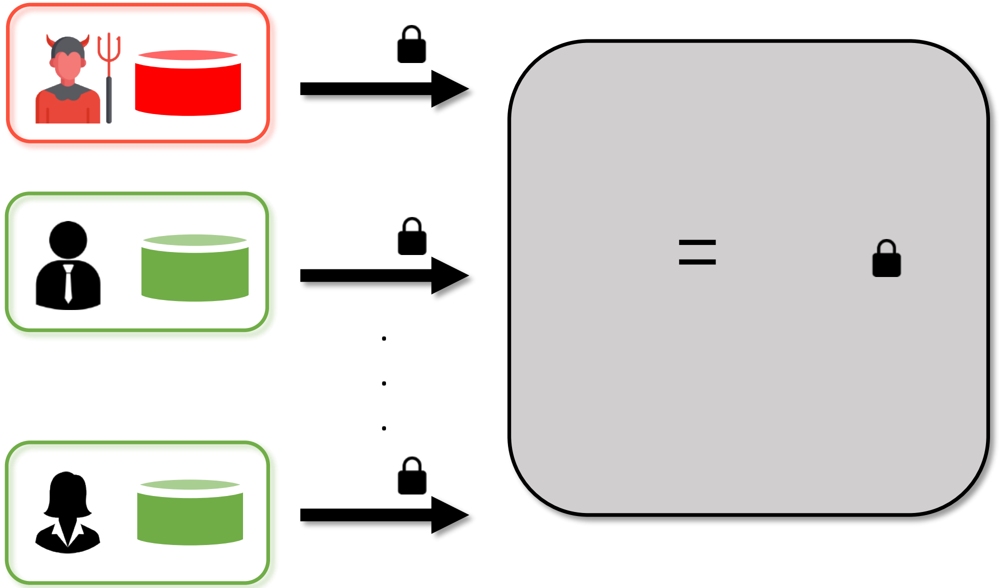
- Any per-input robustness check from ML literature can be a candidate [SWKL17] [BVHES18] [DMGPT18] [SKSM19] [LCWLC20] [XKG19][X21][SH21] [CFLG21]

- E.g.: Norm bound
  $Valid(u) = \mathbb{I}[||u||_2 < \rho]$



Secure Aggregation with Verified Inputs

- **Integrity**
  - Securely verifies the integrity of each input
  - Aggregates **only** well-formed inputs
- **Privacy**
  - Releases only the **final aggregate** in the **clear**

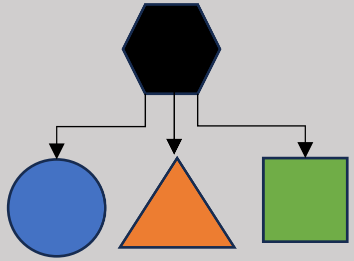


Challenges in Practice

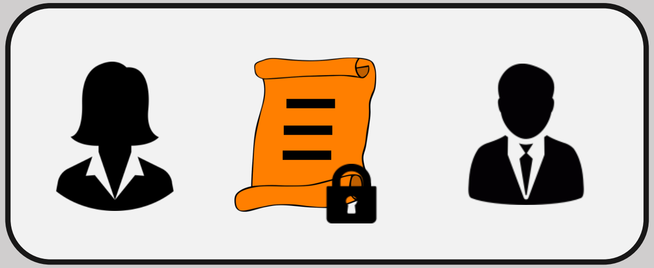
- **Detect vs remove malformed inputs**
- Inflexible integrity checks
- Incompatible with real-world deployment

Cryptographic Primitives

- **Privacy –** Secret Sharing Scheme
  - Allows multiple untrusting parties to securely compute on their secret inputs



- **Integrity –** Zero-Knowledge Proof
  - Allows secure proofs of statements
  - **New cryptographic tool** – Extend SNIP [CB17] to malicious threat model

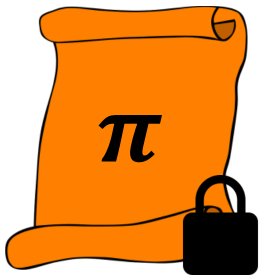


Zero Knowledge Proof

Prover

Verifier

- **Statement:** Is the input u is well-formed, i.e., $Valid(u) = 1$?
- **Claim:** Yes

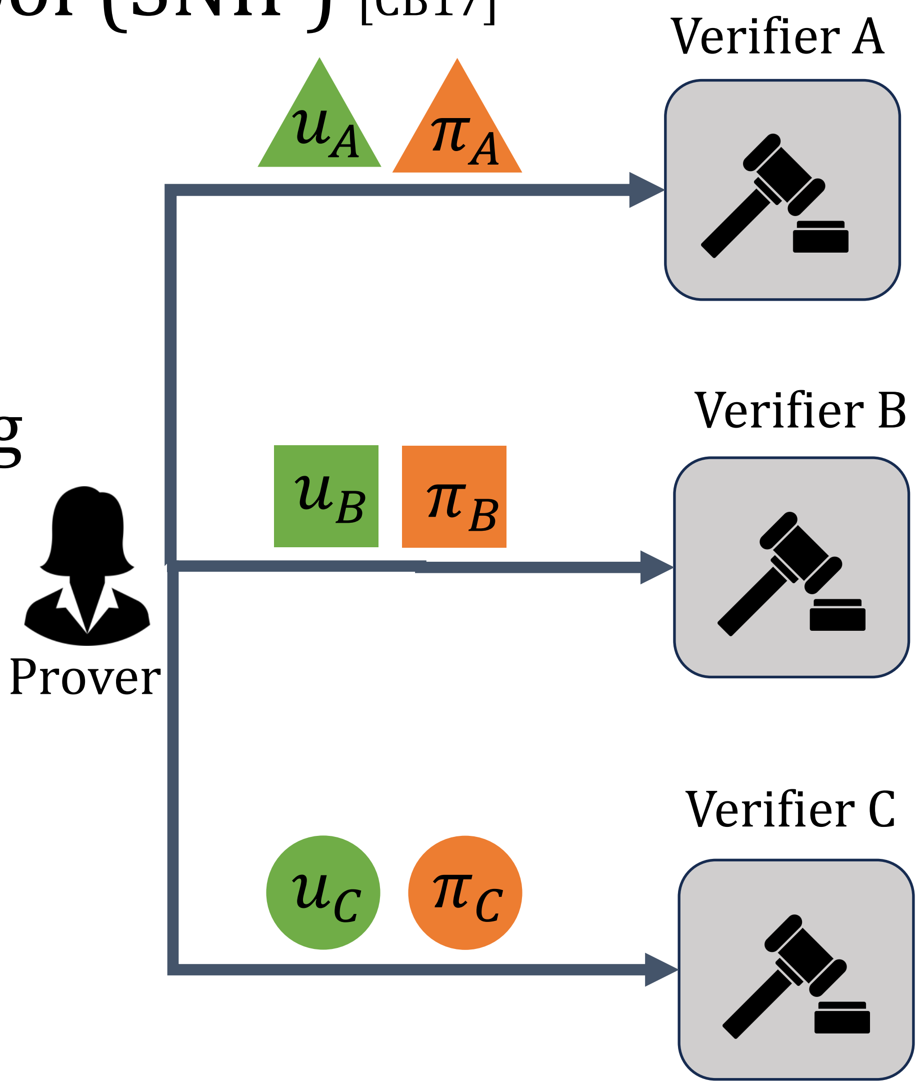


In our setting, client is prover; server is verifier

- Verifier is convinced of the proof iff the prover’s **claim is true**
- Verifier learns **nothing else**

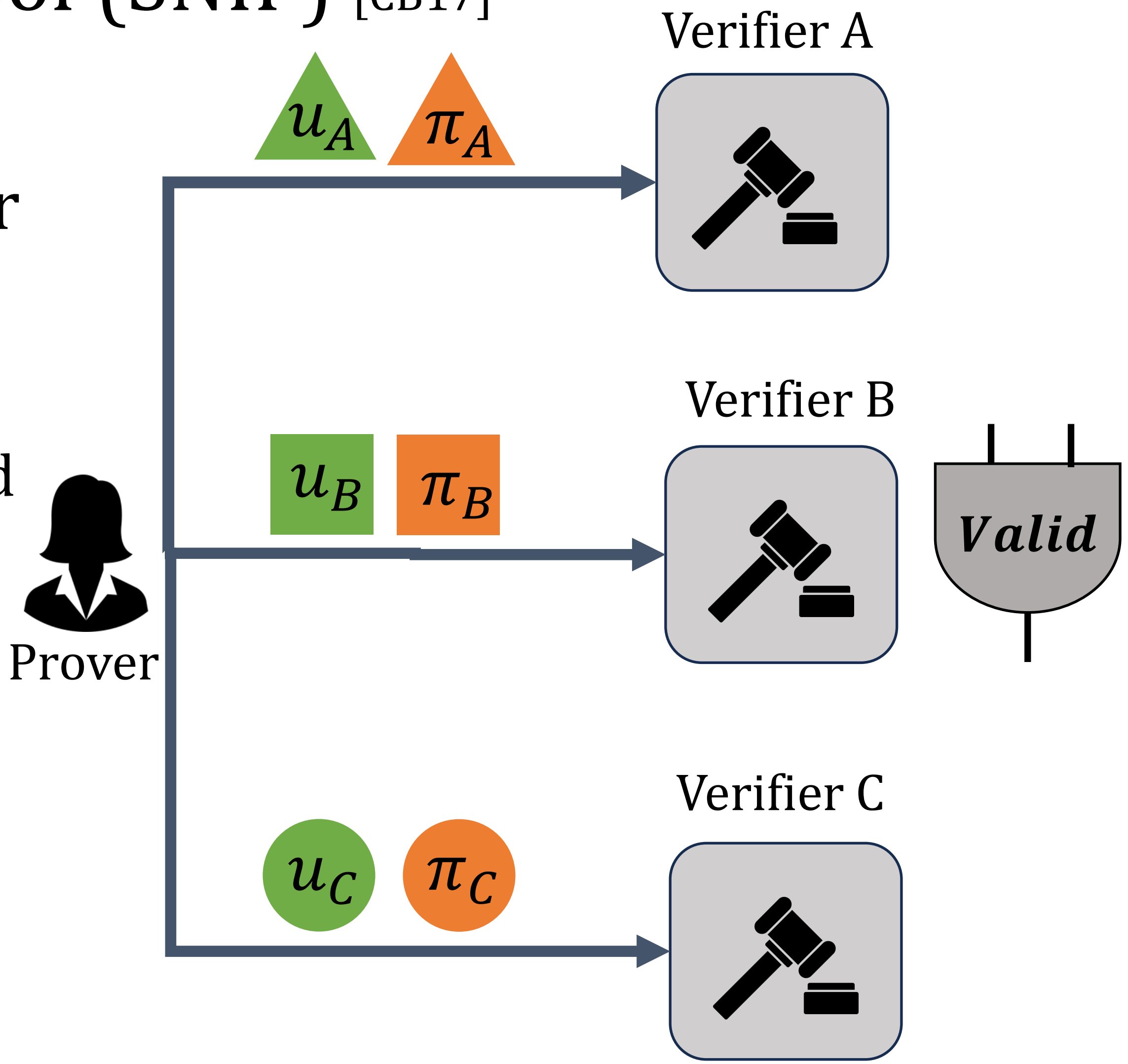
Secret-Shared Non-Interactive Proof (SNIP) [CB17]

- Light-weight ZKP
- Optimized for the client-server setting
  - Client is prover
  - Server is verifier
- Requires **multiple honest** verifiers, i.e., servers



Secret-Shared Non-Interactive Proof (SNIP) [CB17]

- Prover uses secret-sharing scheme for masking input u
- Verifiers
  - Hold a public predicate $Valid(\ )$ expressed as an arithmetic circuit
  - Want to test if $Valid(u) = 1$, without learning u



EIFFeL supports **arbitrary** $Valid()$ expressed as an arithmetic circuit with public parameters

Challenges in Practice

- Detect vs remove malformed inputs
- **Inflexible integrity checks**
- Incompatible with real-world deployment

Challenges in Practice

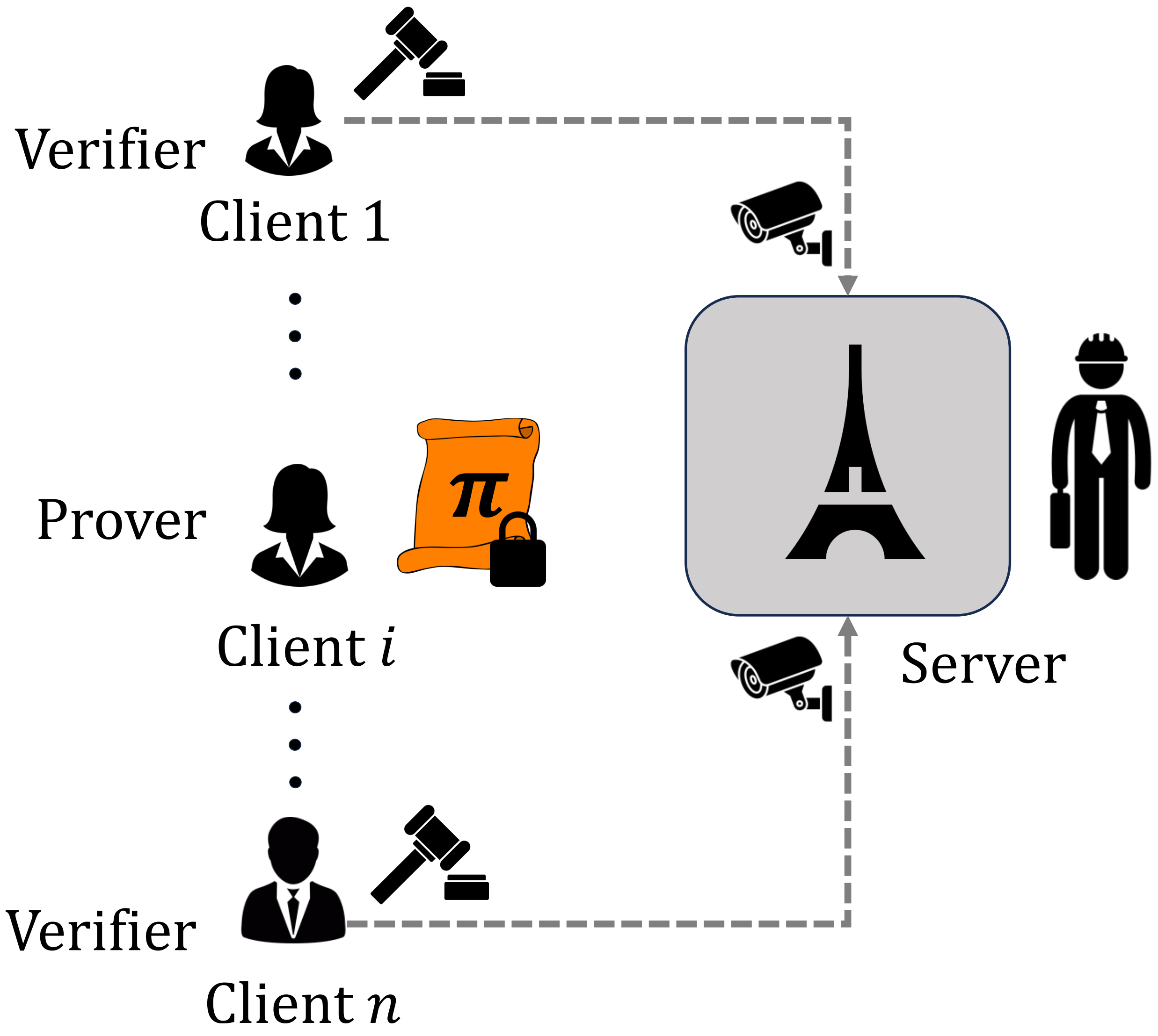
- Detect vs remove malformed inputs
- Inflexible integrity checks
- **Incompatible with real-world deployment**

EIFFeL Design Highlights

- New protocol for FL
- **Multiple verifiers** – Clients act as the verifiers under server supervision

EIFFeL

- Client i is the prover
- All other Client $j, j \neq i$ act as verifiers
- Server coordinates the verification process



Challenges in Practice

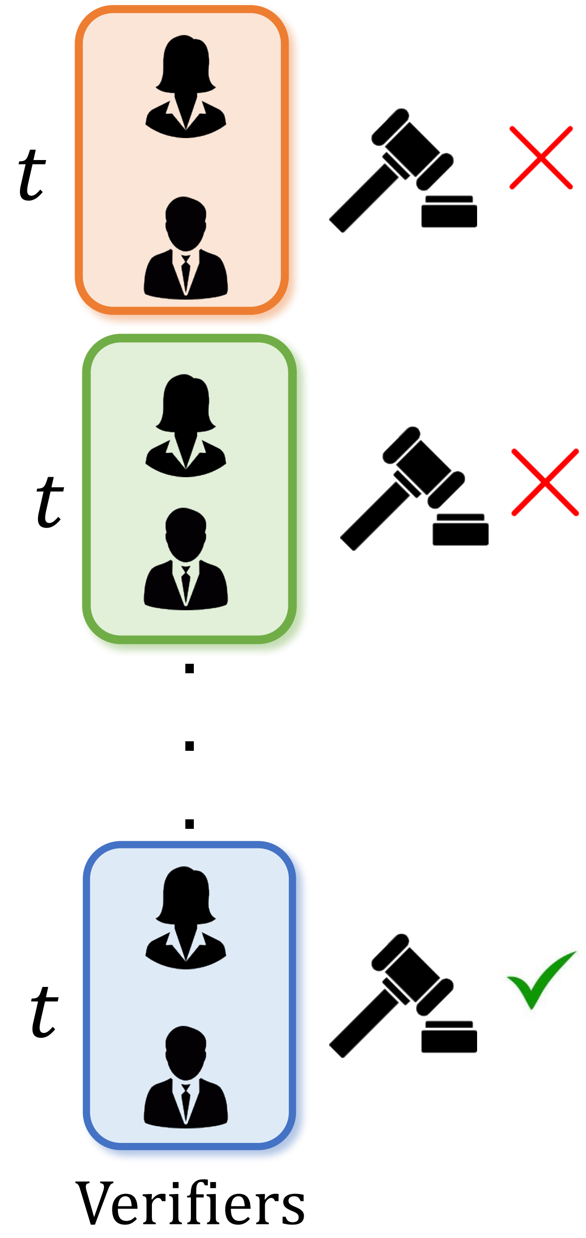
- Detect vs remove malformed inputs
- Inflexible integrity checks
- **Incompatible with real-world deployment**

EIFFeL Design Highlights

- New protocol for FL
- Clients act as verifiers
- **Malicious threat model** – Introduce redundancy

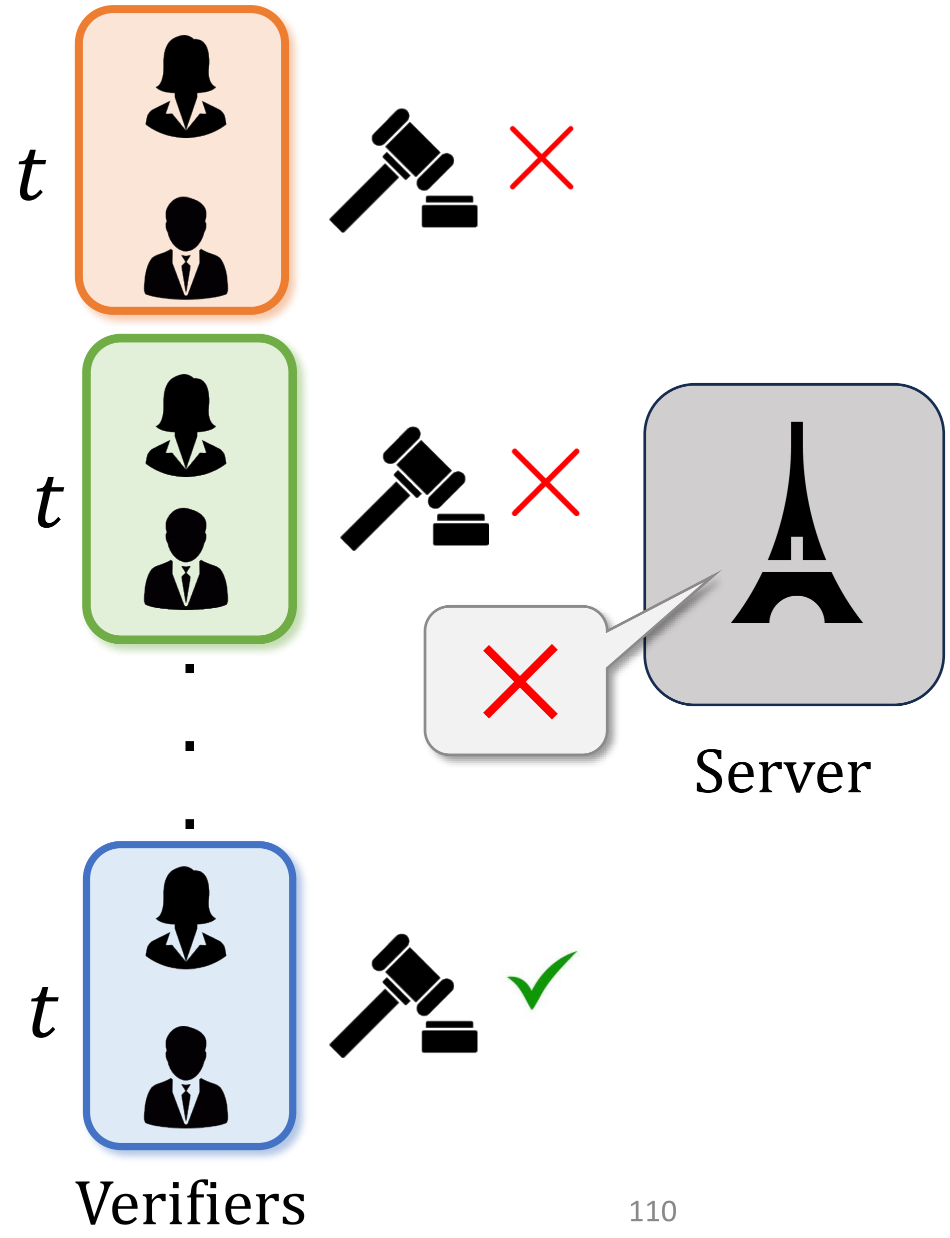
EIFFeL

- **Threshold secret sharing** for masking inputs – Parameter ***t***
- Any set of ***t*** clients can emulate a SNIP verification – **Redundancy**
- Server uses this redundancy for robust verification



EIFFeL

- **Threshold secret sharing** for masking inputs – Parameter ***t***
- Any set of ***t*** clients can emulate a SNIP verification – **Redundancy**
- Server uses this redundancy for robust verification



Under the Hood

- Principled approach
  - Reed Solomon codes
  - Verifiable secret shares

- **New cryptographic tool** – Efficient extension of SNIP to
  - a **fully malicious** threat model
  - in a **single server** setting

Challenges in Practice

- Detect vs remove malformed inputs
- Inflexible integrity checks
- **Incompatible with real-world deployment**

Privacy Guarantee

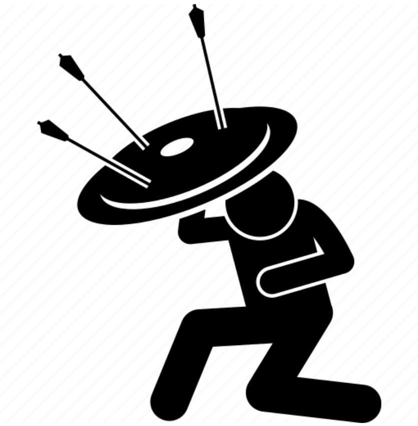
Thm. (Informal) EIFFeL instantiates a secure aggregation w/ verified inputs protocol

- Input privacy – Set threshold $t = m + 1$ where $m =$ #malicious clients
- Input integrity – $m < {}^{n}/_{3}$ where $n =$ #total clients in a single round

Evaluated Attacks and Defenses

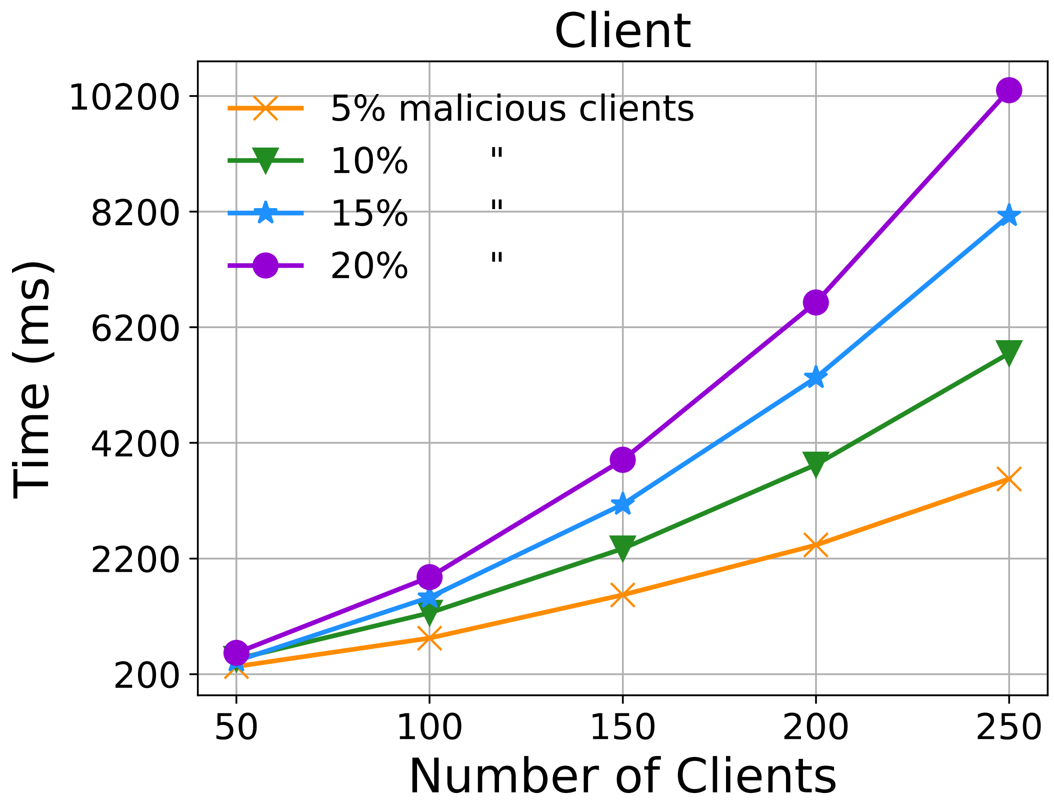
- 6 poisoning attacks
  - Label flipping
  - Min-max / Min-sum attack
  - Backdoor attacks

- 4 defenses ($Valid(\,)$ predicates)
  - Norm Bound – $Valid(u) = \mathbb{I}[||u||_2 < \rho]$
  - Cosine Similarity – $Valid(u) = \mathbb{I}[\frac{<u,u'>}{||u||_2||u'||_2} < \rho]$



Performance

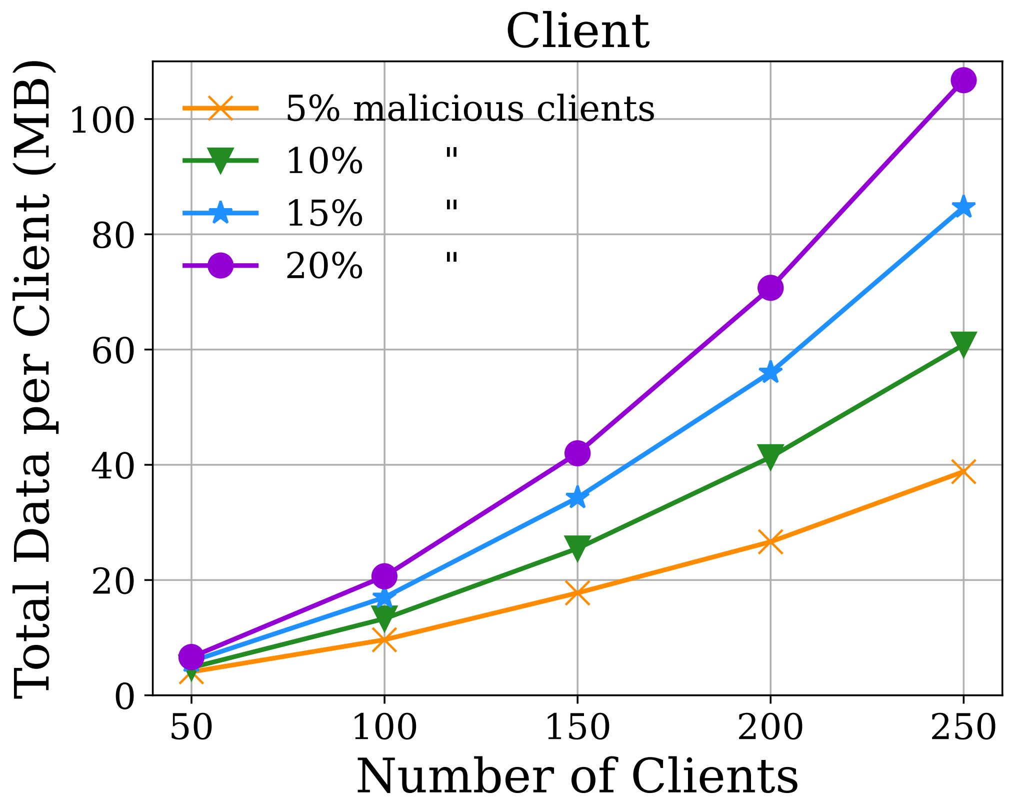
- EIFFeL has low computation cost
  - E.g: With **100** clients and **10%** poisoning, a model takes **2.4s** per iteration per client for MNIST



Performance

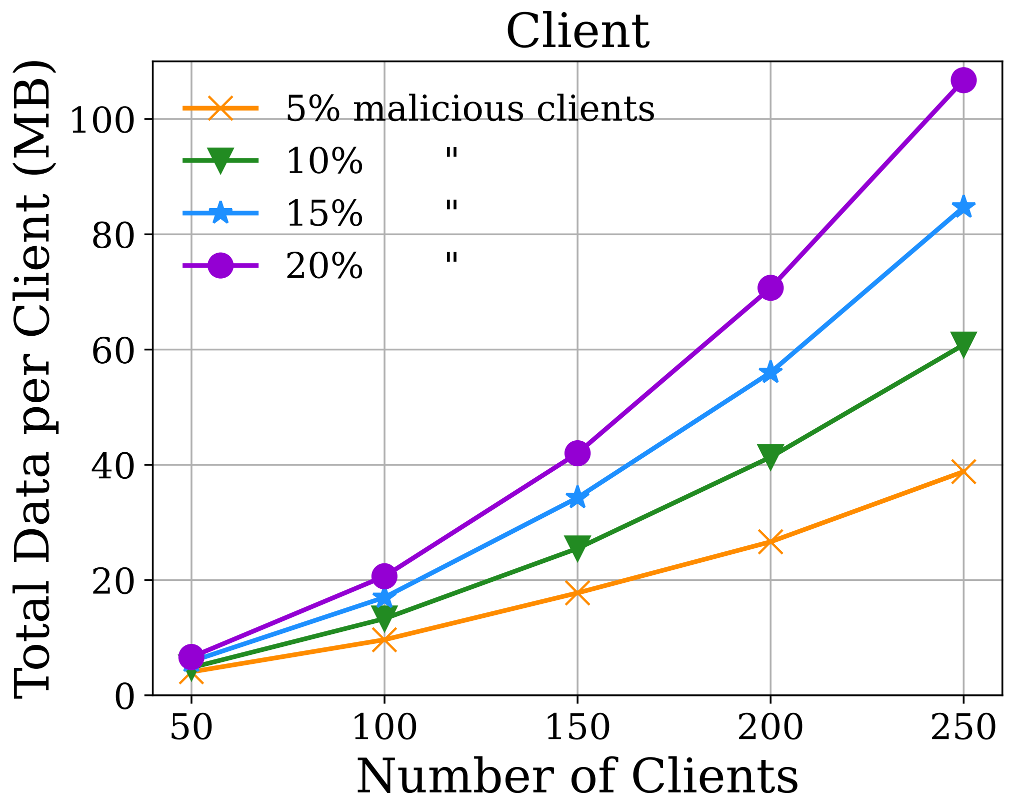
- EIFFeL has low computation cost
  - E.g: With **100** clients and **10%** poisoning, a model takes **2.4s** per iteration per client for MNIST

- EIFFeL has low communication cost
  - E.g.: Client communication cost is **13.3MB**



Performance

- EIFFeL has low computation cost
  - E.g: With **100** clients and **10%** poisoning, a model takes **2.4s** per iteration per client for MNIST
- EIFFeL has low communication cost
  - E.g.: Client communication cost is **13.3MB**
- Optimizations – **2.3X** improvement



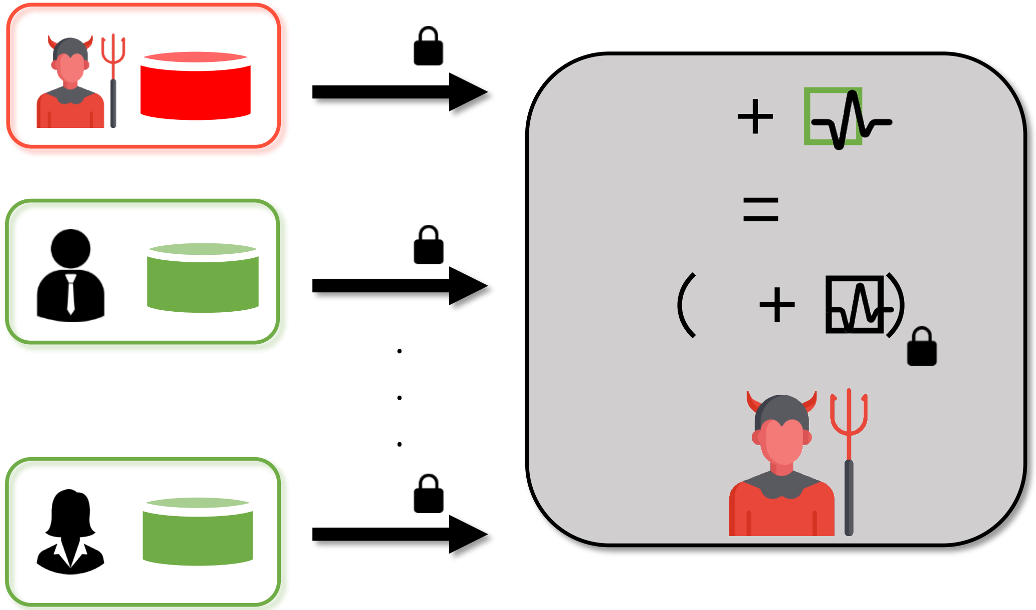
Extension to DP

- Integrate with differential privacy
  - Output noisy aggregate

Threat Model – Server

- Aggregate can reveal information [BDSSSP21] [DFA22][SAGJA23]
  - Server can leverage aggregates across multiple iterations

- **Add noise to the aggregate to ensure DP**



Extension to DP

- Integrate with differential privacy
  - Output noisy aggregate
- Prior work on adding DP noise w/ secure aggregation
  - Seamless adoption in EIFFeL

Key Takeaway

Can we thwart poisoning attacks from clients **w/o** violating privacy in federated learning?

Yes, with EIFFeL!

Challenge	Solution
Detection and removal of malformed inputs	New type of aggregation protocol
Flexible and efficient integrity checks	Use light-weight ZKP
Single server	New cryptographic tool • Clients acts as verifiers • Redundancy in the process

TL;DR

- Decentralized data ecosystems
  - Dual threat
- End-to-end privacy guarantee needs both cryptography and DP
- Gives rise to complex trade-offs requiring careful design