EPFL

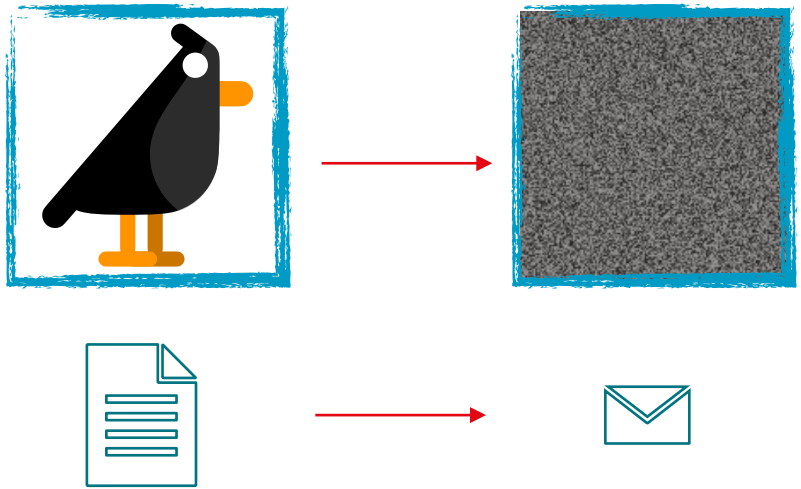
Thwarting Malicious Adversaries in Homomorphic Encryption Pipelines

Sylvain Chatel
CISPA

Saarbrücken, DE

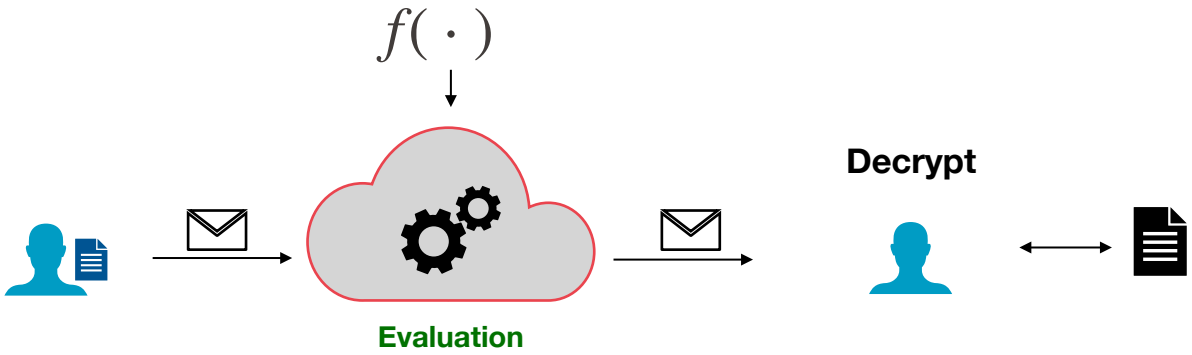
Bilkent - Mar 28th 2024

Encryption

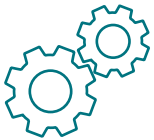


- Protect data:
  - At rest
  - In transit

Homomorphic Encryption

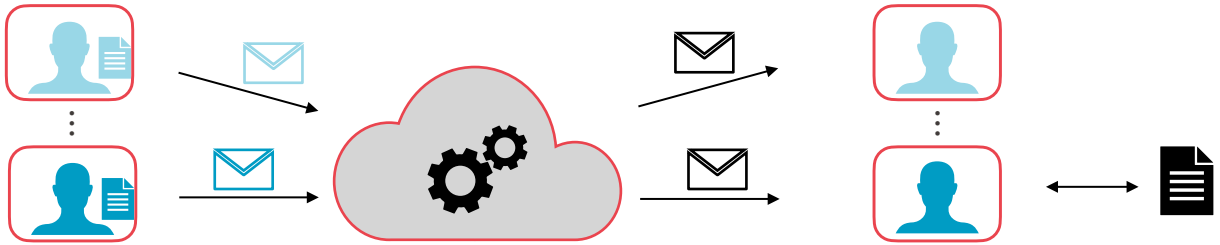


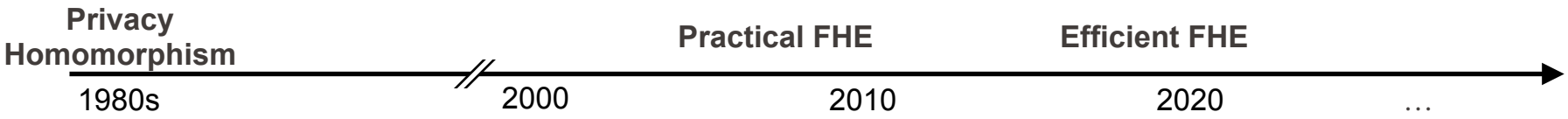
- At rest
- In computation
- In transit


Passive Adversary

Honest but curious (i.e., passive semi-honest)





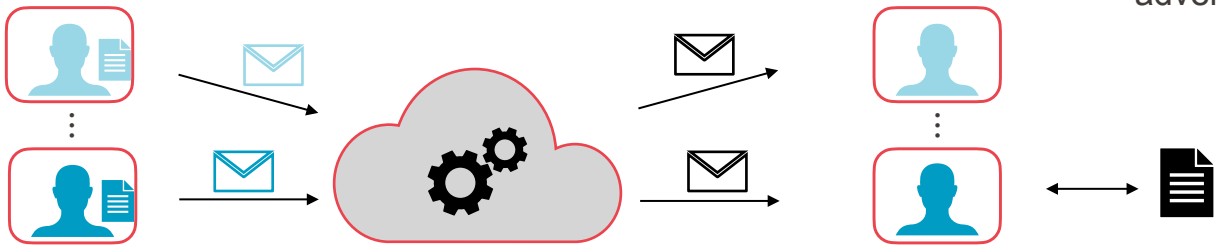
- Correctness
- Semantic security

Multiparty FHE

Active Adversary

~~Passive adversary (i.e., Honest but curious)~~

Active adversary



Evaluation

Privacy Homomorphism — 1980s

Practical FHE — 2000

Efficient FHE — 2010

2020 …

Multiparty FHE

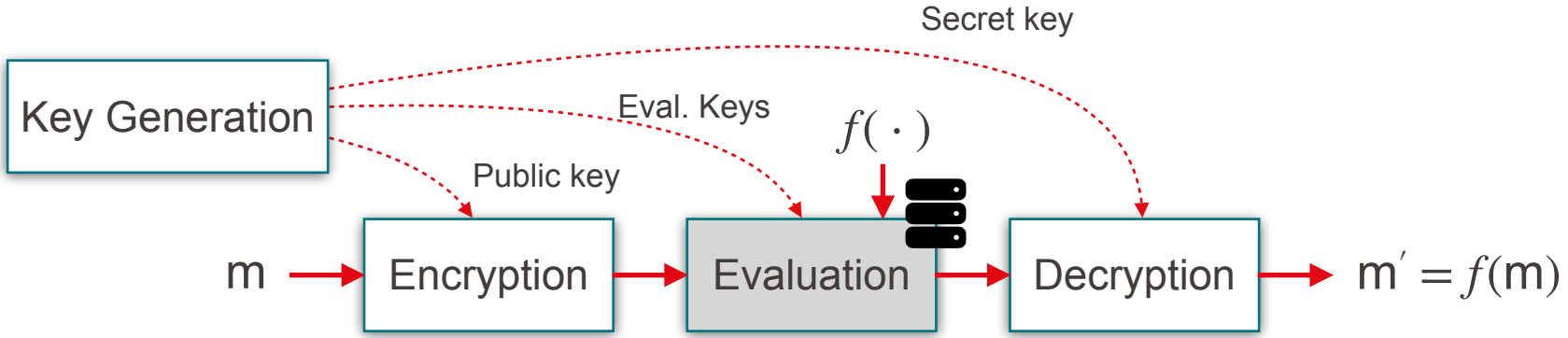
No Correctness

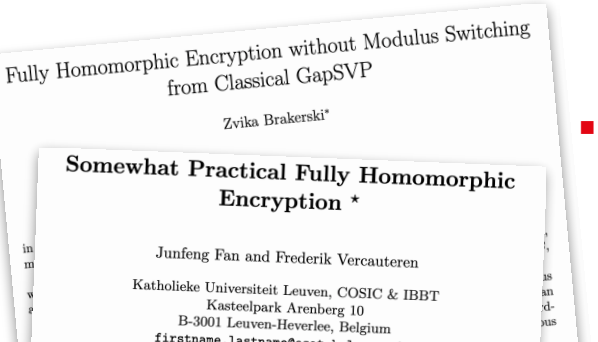
Unfairness

Privacy Leakage

Need for new protection mechanisms

Homomorphic Encryption Pipeline




Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP

Zvika Brakerski*

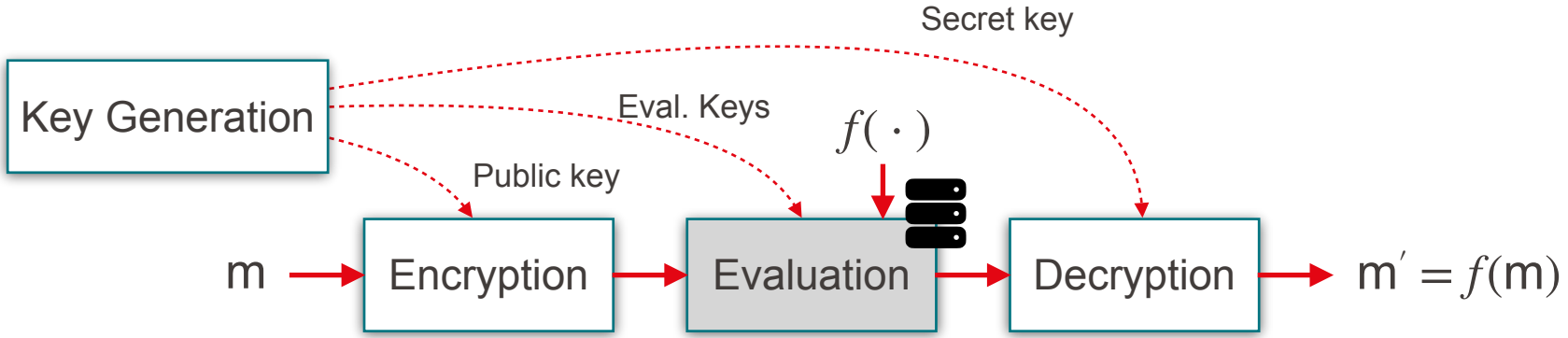
Somewhat Practical Fully Homomorphic Encryption *

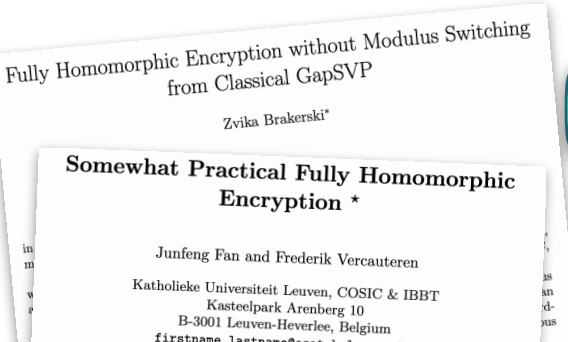
Junfeng Fan and Frederik Vercauteren

Katholieke Universiteit Leuven, COSIC & IBBT
Kasteelpark Arenberg 10
B-3001 Leuven-Heverlee, Belgium

- Works over polynomial rings

Challenges of the HE Pipeline




Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP

Zvika Brakerski*

Somewhat Practical Fully Homomorphic Encryption *

Junfeng Fan and Frederik Vercauteren

Katholieke Universiteit Leuven, COSIC & IBBT
Kasteelpark Arenberg 10
B-3001 Leuven-Heverlee, Belgium

Algebraic Structure

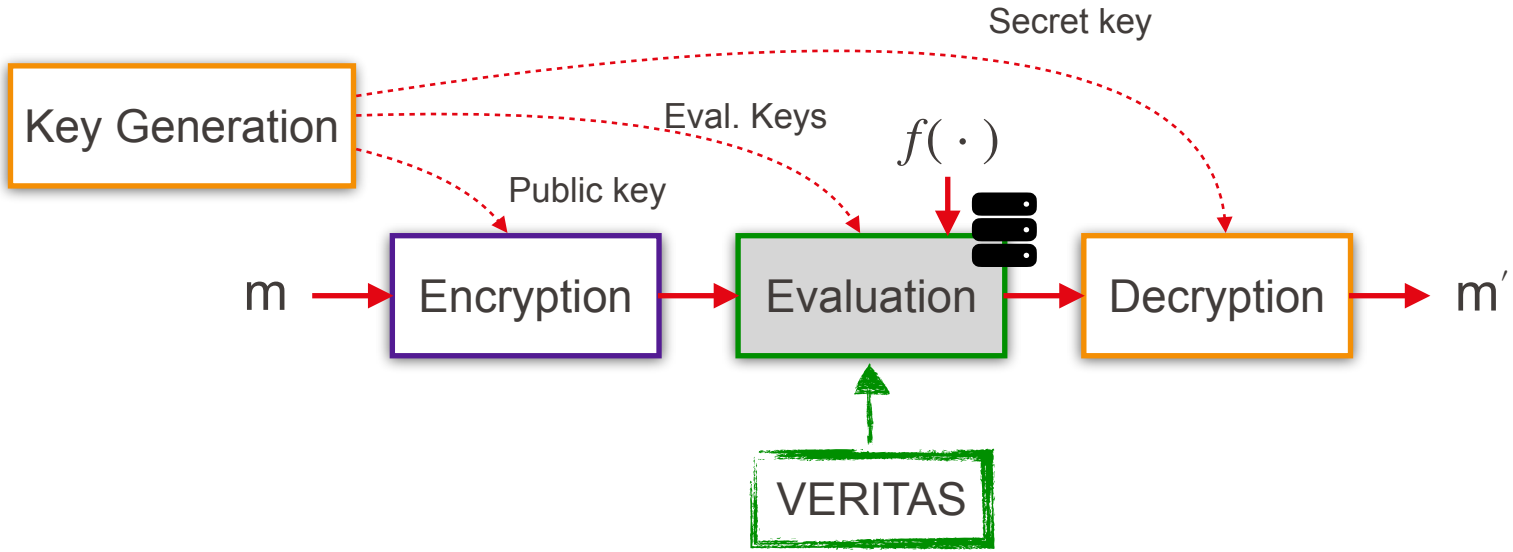
Ciphertext Expansion

Non-algebraic Ops.

HE Optimisation

Efficiency

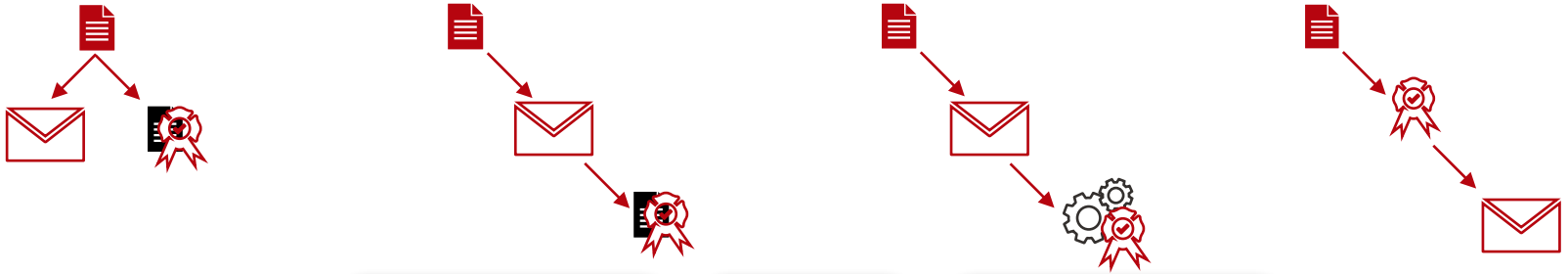
Verifying the Server Computation



Current Verifiable Homomorphic Computation Techniques

- Approaches to prove the evaluation by a malicious server

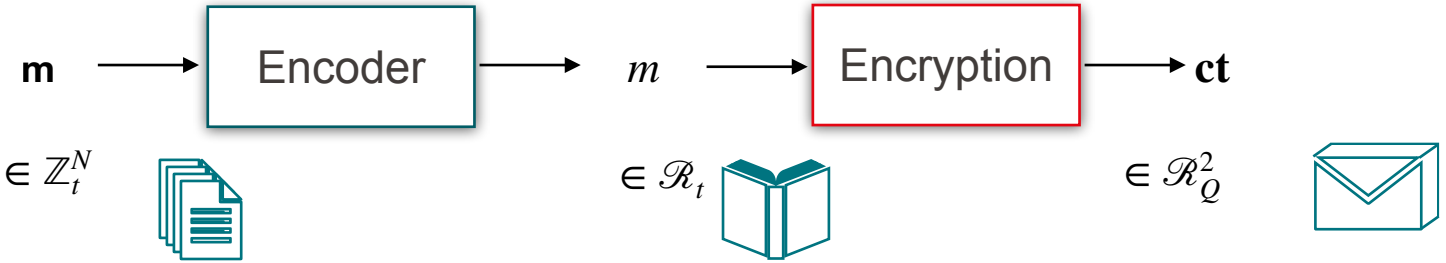
MAC-&-Enc	Enc-then-MAC	Compute and Prove	Plaintext MAC
e.g., LWZ18	e.g., FGP14	e.g., FNP20, BCFK21, GNS21	Ours



Security	Algebraic Structure	Efficiency	Non-algebraic Ops.

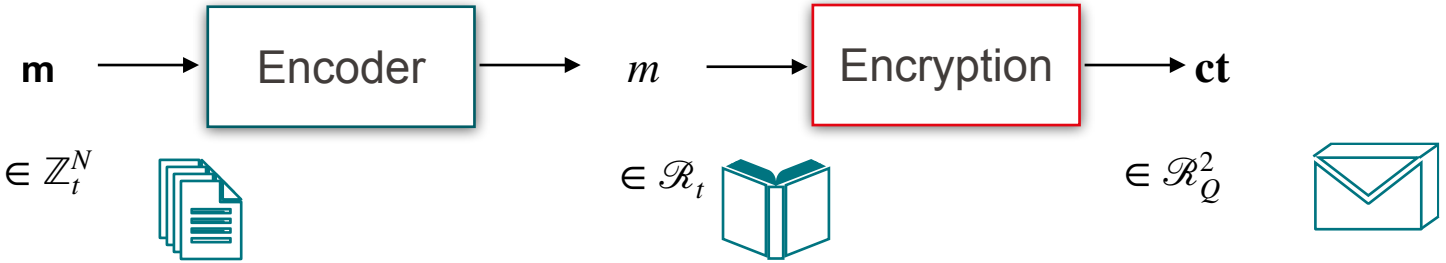
The Plaintext Encoding

- In FHE pipelines:



The Plaintext Encoding

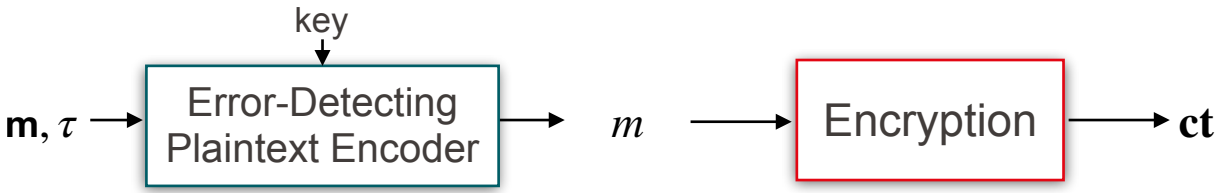
- In FHE pipelines:



- The encoding of the message is compatible with the homomorphic evaluation
- Operation on the ciphertexts lead to operation over the plaintext vector

Our Approach

- Rely on a **homomorphic authenticator** over finite fields
- Design a new data encoder for the plaintext

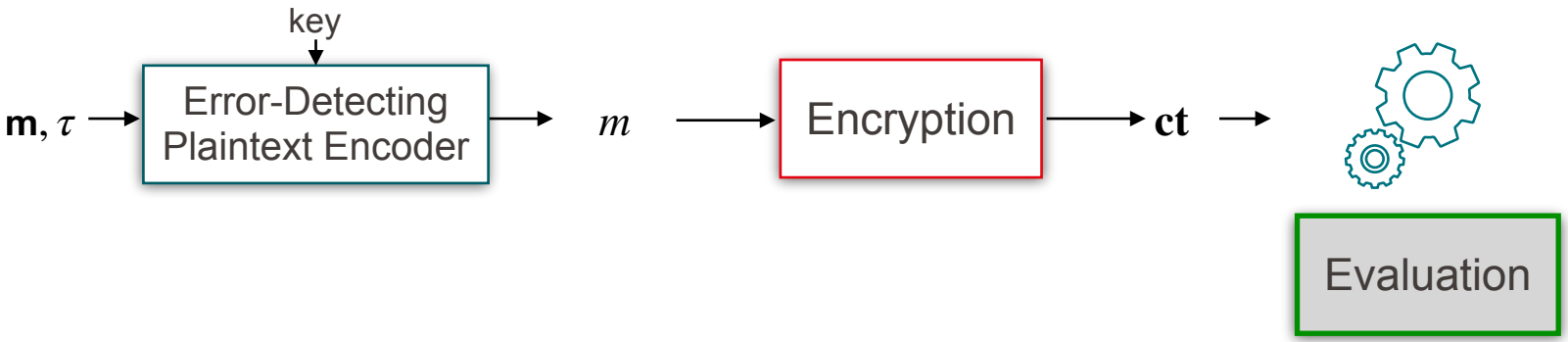


Each message **m** is associated with a identifier string τ

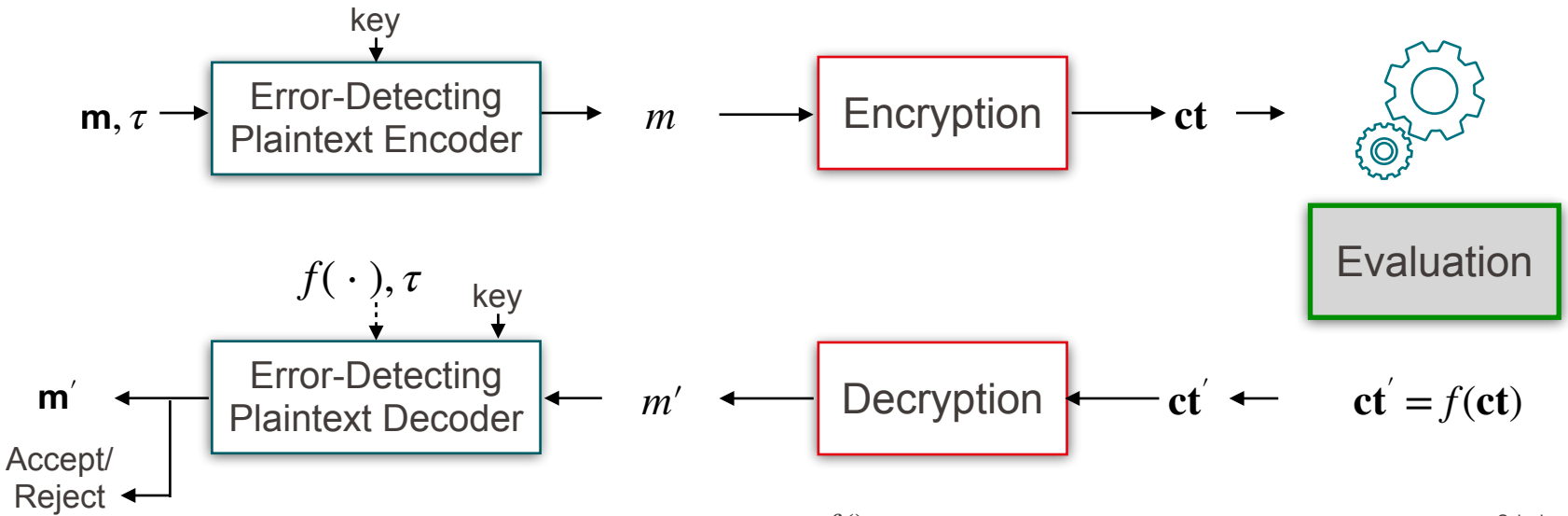
Example: $\mathbf{m} = (0,1,\ldots,1)$

τ = "Electrical activity per half hour of Alice's household"

Our Approach



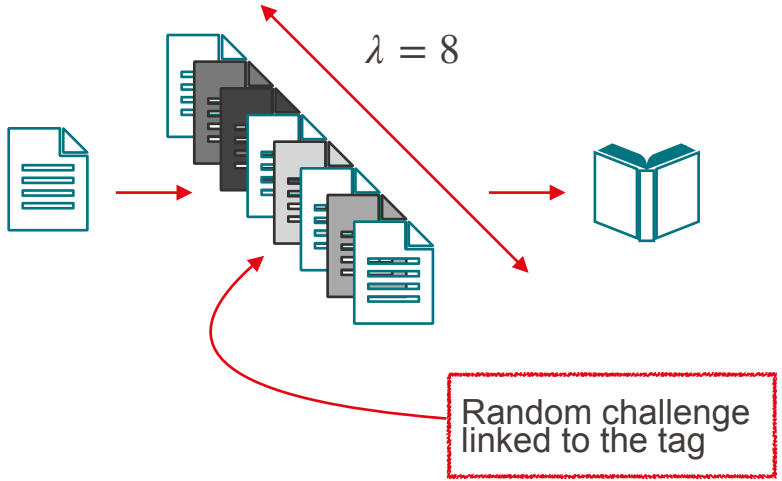
Our Approach



Challenge-values are used to evaluate $f()$ and check the result

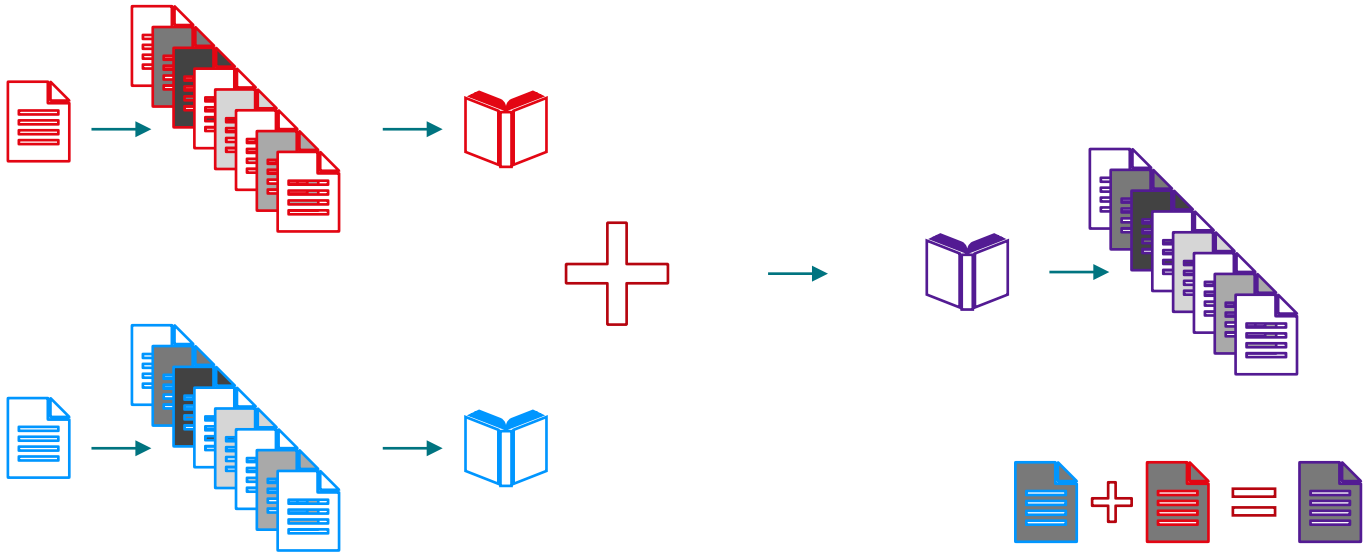
Replication Encoding (REP) — Intuition

- Replicate the data and add challenge values

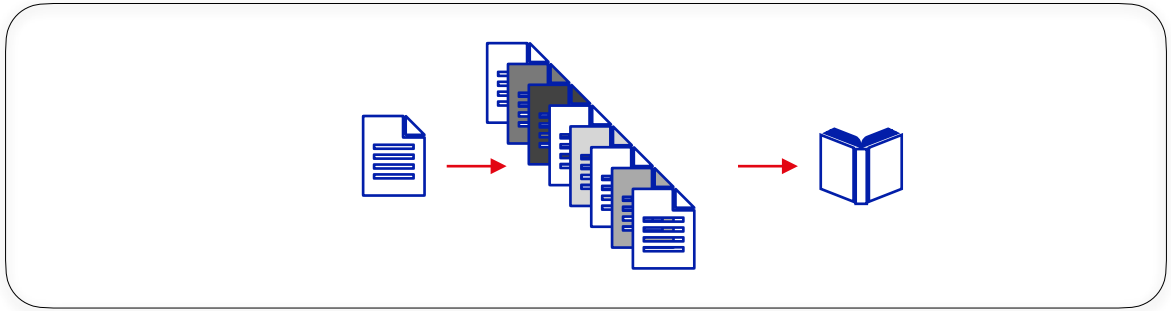


Replication Encoding (REP) — Intuition

- Replicate the data and add challenge values



Replication Encoding (REP) — Overview



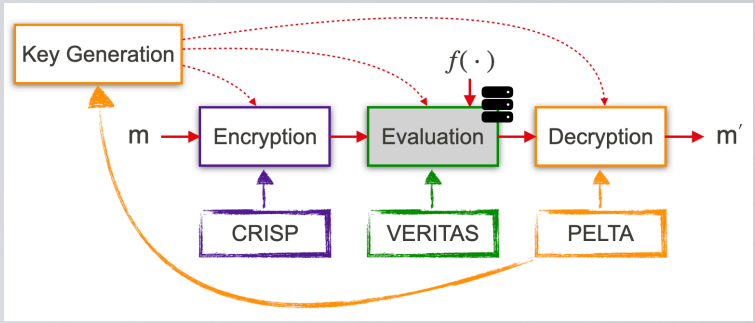
- Pros
  - Fixed overhead
  - Supports any HE op.
  - No assumption on the plaintext
- Cons
  - Linear overhead in λ
  - $\lambda/2$ challenge values

Implementation

- Easy-to-use library to change in a few lines already existing FHE pipelines
- Enable verifiability with as little as 1-3x computation overhead

- “Lattigo v3.0”, EPFL-LDS, Tune Insight SA

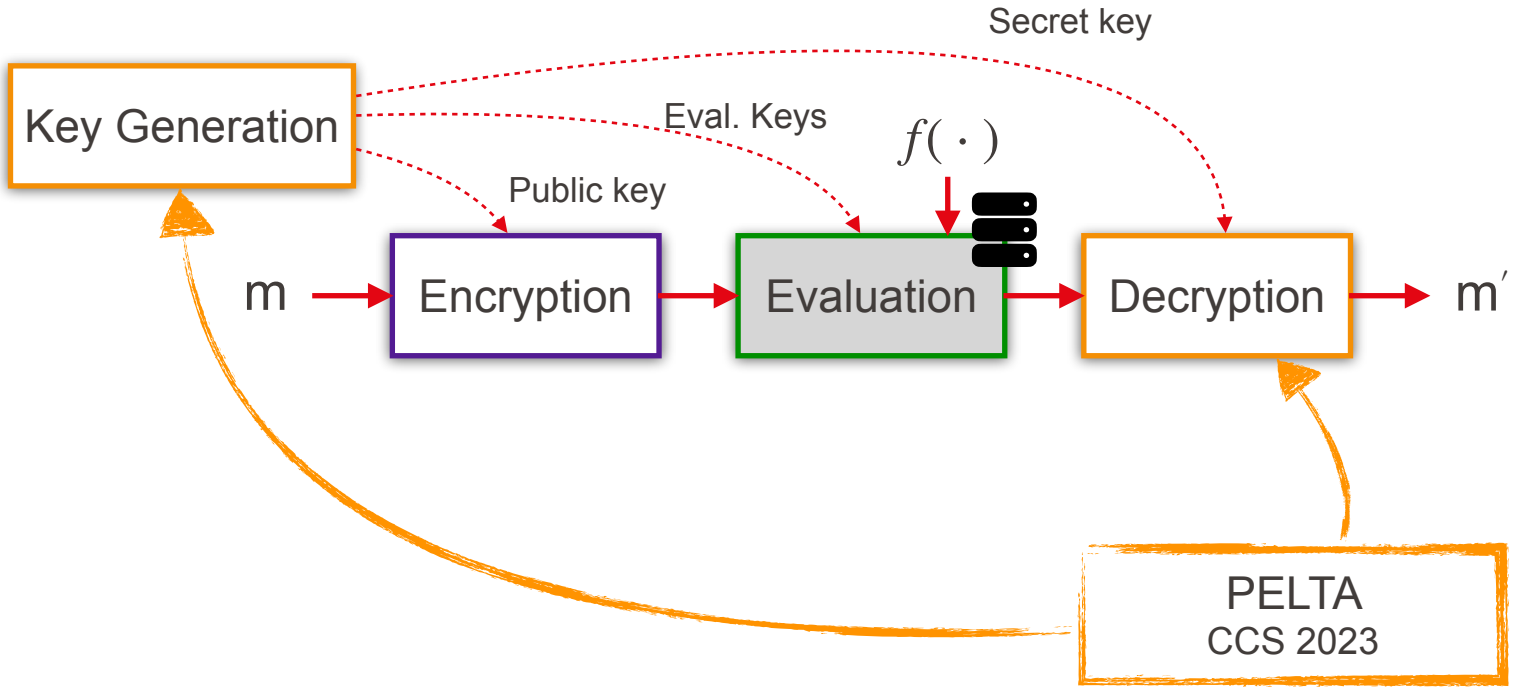
Summary of VERITAS



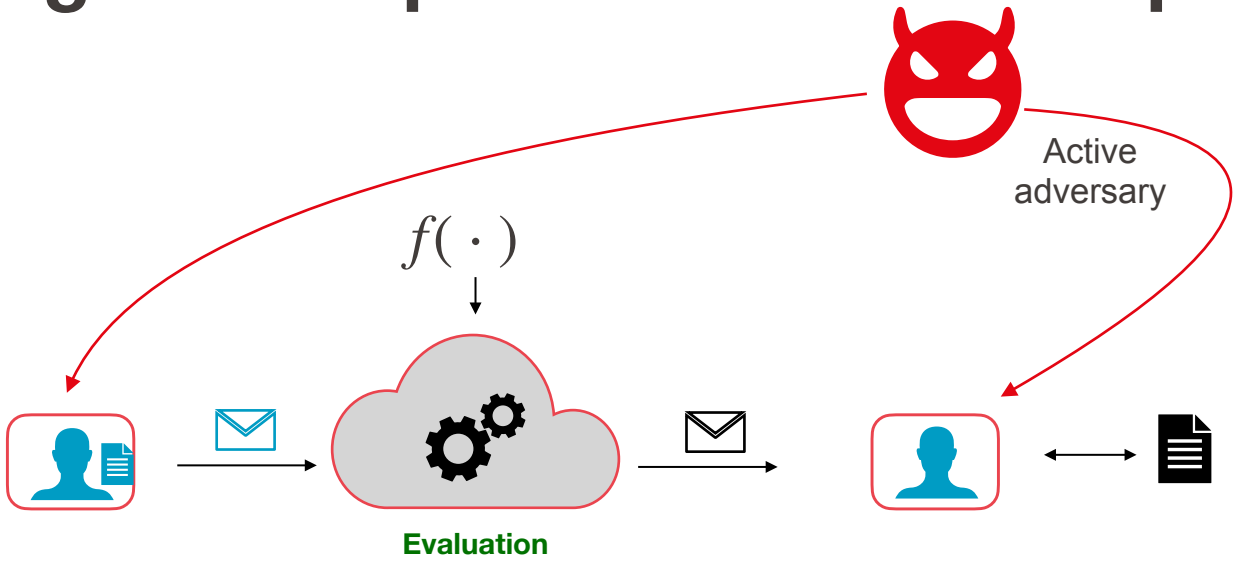
VERITAS

- **Efficiently** and **seamlessly** enable **verification** capabilities of **computation** under **encryption**
- Copes with several use-cases and a **wide range of operations**
- **Works off-the-shelf with state-of-the-art FHE** scheme and **all their operations**.

Verifying Client Operations

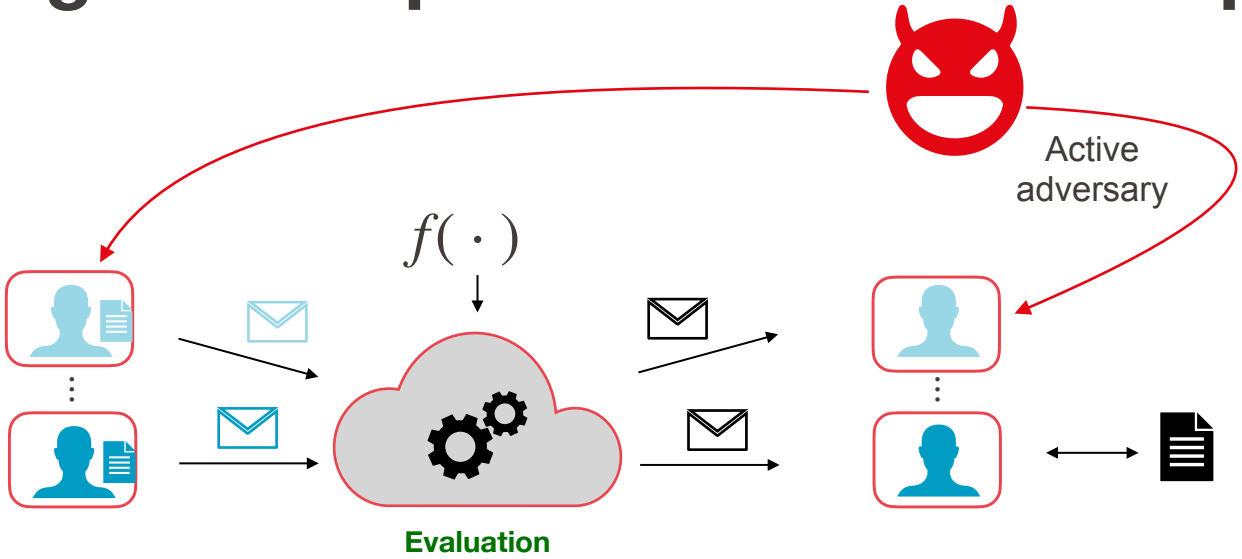


Verifying Client Operation in the HE Pipeline



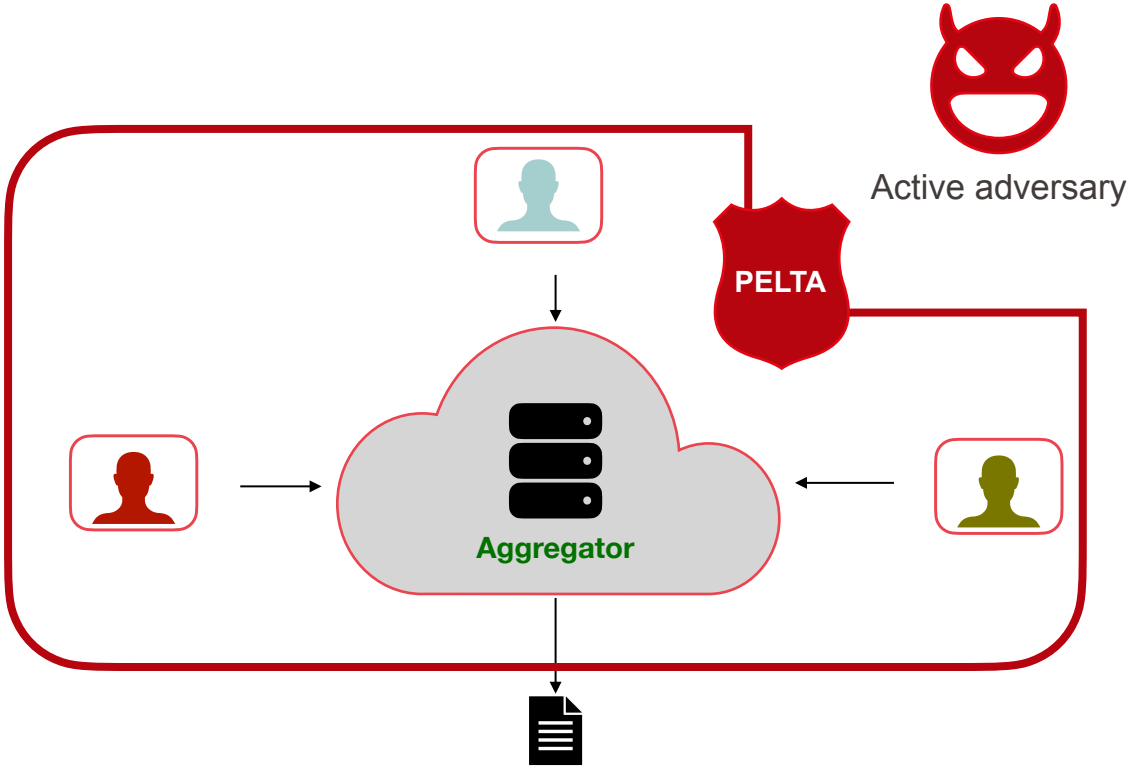
- Verifying the client operation makes little sense in the single client setting:
  - the verifier is the client

Verifying Client Operation in the MHE Pipeline



- We consider the **multi-client** HE setting
- The verifier is any of the client or an additional decryptor

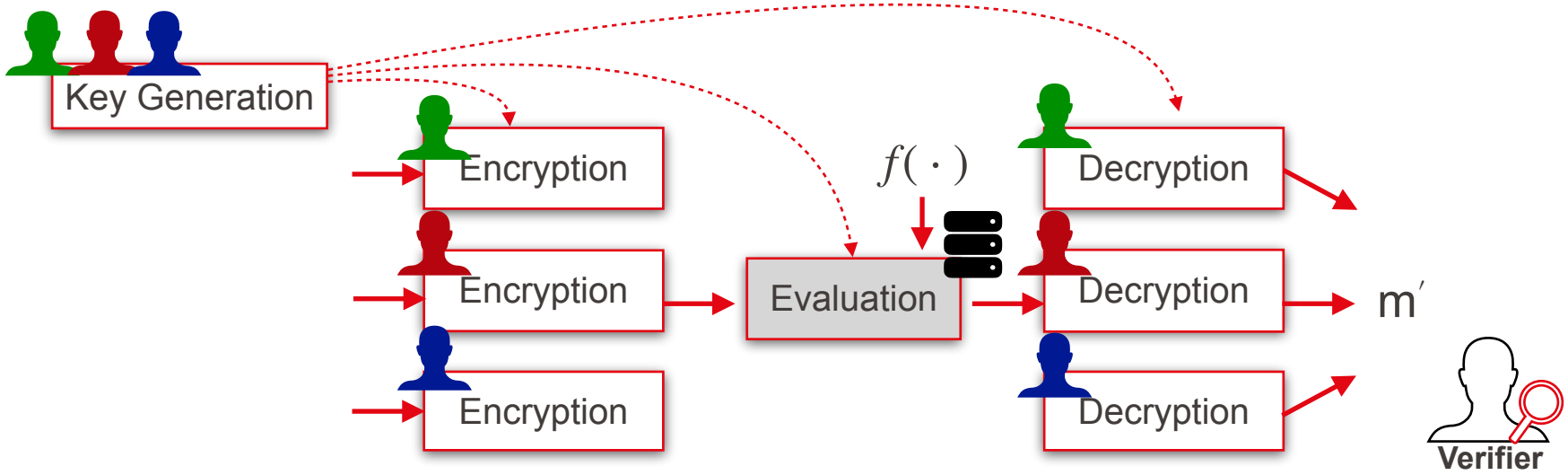
PELTA: Objectives



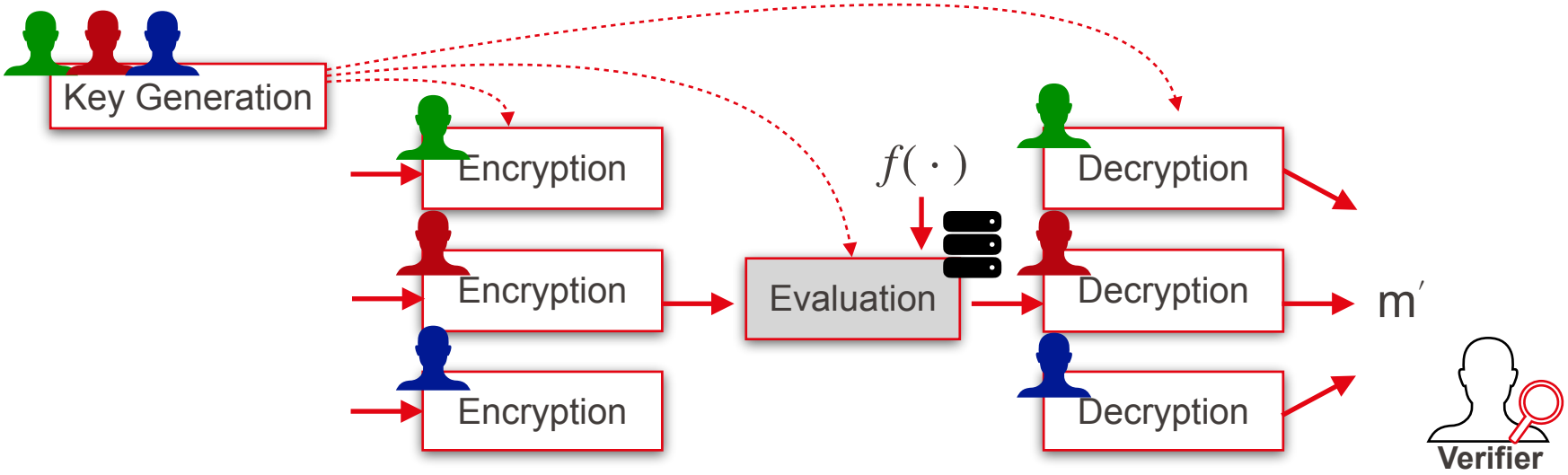
- Correctness
- Confidentiality

PELTA is designed to **prevent** a **malicious adversary** from **covertly** tampering with the process

Overview of HE-based MPC



Overview of HE-based MPC



- Two flavours of Multiparty FHE (MHE) in a nutshell:
  - **Multi-key HE** (ciphertexts under different keys)
  - **Threshold HE** (one collective key)
- Our work abstracts the difference

Classification of MHE Operations

- All MHE operations can be expressed by basic operations

Secret Key Generation

Encryption

Evaluation

Non-interactive

Public Key Generation

Client-aided Evaluation

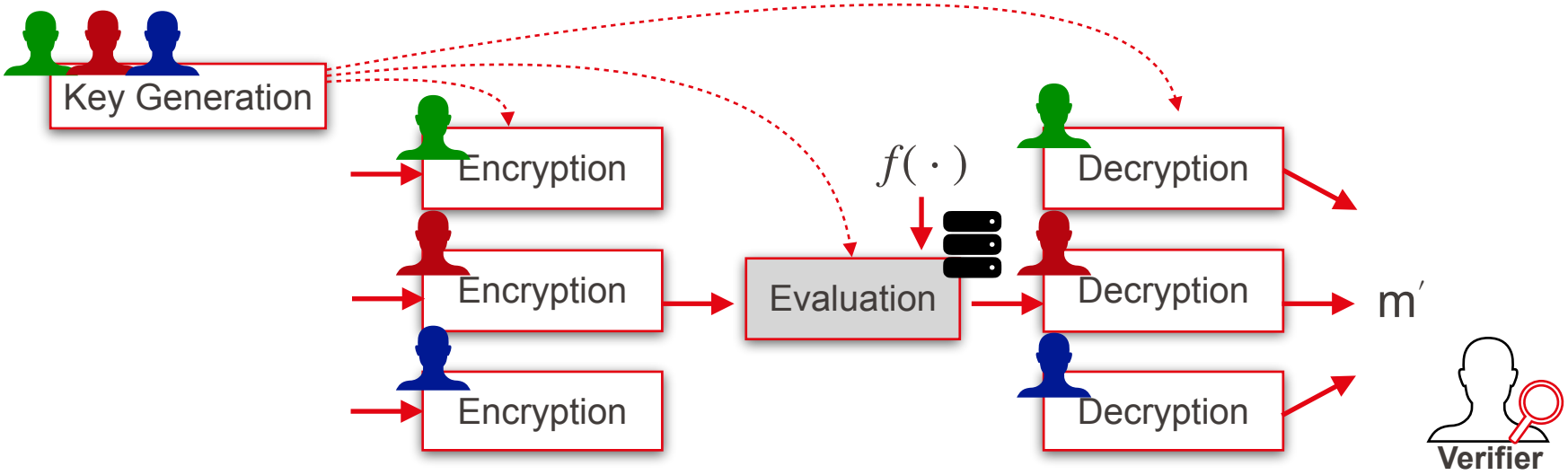
Decryption

Interactive

Eval. Key Generation

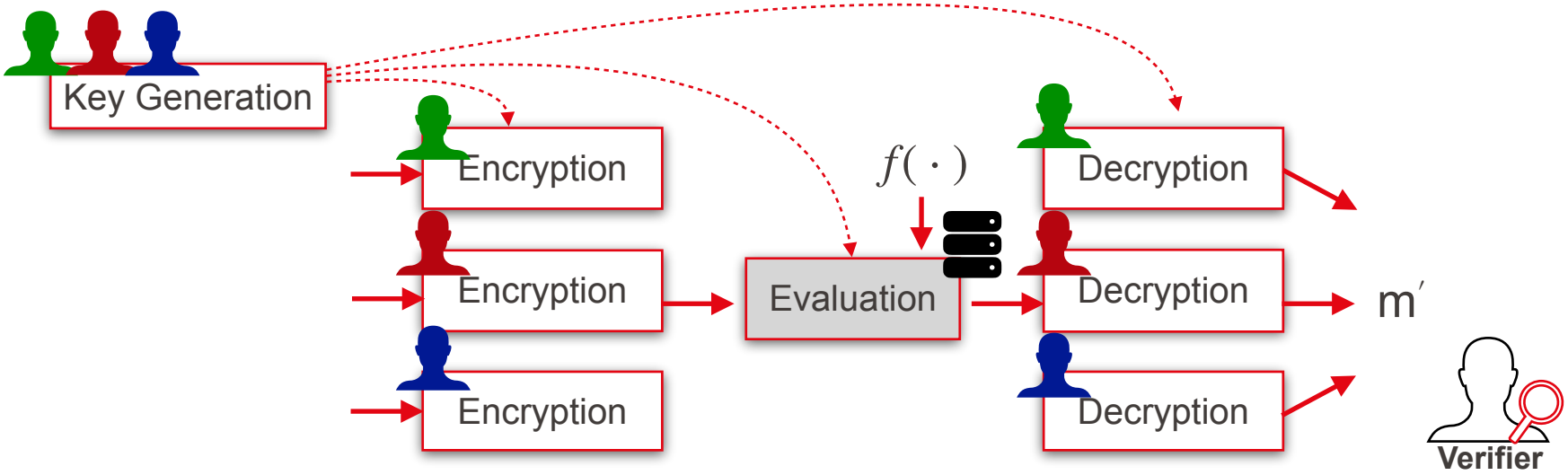
Aggregator

Overview of HE-based MPC



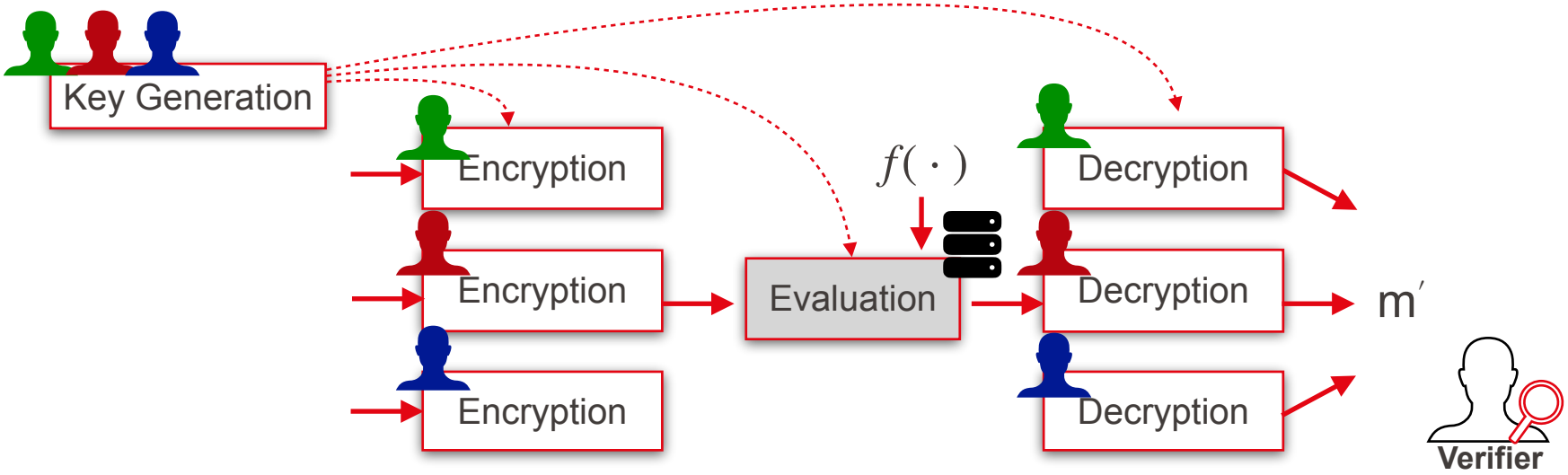
- Need to check **all the different blocks** ran by the clients

Overview of HE-based MPC



- Need to check **all the different blocks** ran by the clients
- Need to **link the blocks** together

Overview of HE-based MPC



- Need to check **all the different blocks** ran by the clients
- Need to **link the blocks** together
- Challenges:

Algebraic Structure | Ciphertext Expansion | Efficiency

Overview of MHE Operations

- **Simplification** of MHE operations' structure:

Compute a noisy linear product of the secret key with a public polynomial

Overview of MHE Operations

- **Simplification** of MHE operations' structure:

  Compute a noisy linear product of the secret key with a public polynomial

  - **<u>Step 1:</u> Local share generation**

  - **<u>Step 2:</u> Aggregation of the shares**

Overview of MHE Operations

- **Simplification** of MHE operations' structure:

Compute a noisy linear product of the secret key with a public polynomial

- **Step 1:** **Local share generation**

$$\text{e.g., } \mathbf{pk}_i = \mathbf{a} \cdot \mathbf{sk}_i + \mathbf{e}_i$$

Noise

i-th party secret key

- **Step 2:** **Sum of the shares**

$$\text{e.g., } \mathbf{cpk} = \sum_i \mathbf{pk}_i$$

Ex: Verifying Local Share Generation

- **Simplification** of MHE operations' structure:
  - **Step 1:** correct **local share generation**
    - Sample random polynomials and create a local share

      e.g., $\mathbf{pk}_i = \mathbf{a} \cdot \mathbf{sk}_i + \mathbf{e}_i$

      Common polynomial from a CRS
    - Generate a proof that
      - $\mathbf{sk}_i$ and $\mathbf{e}_i$ are small secrets
      - the linear relation is correct

Ex: Verifying Local Share Generation

- **Simplification** of MHE operations' structure:
  - **Step 1:** correct **local share generation**
    - Sample random polynomials and create a local share

e.g., $\mathbf{pk}_i = \mathbf{a} \cdot \mathbf{sk}_i + \mathbf{e}_i$

Common polynomial from a CRS

Cut-and-Choose

    - Generate a proof that
      - $\mathbf{sk}_i$ and $\mathbf{e}_i$ are small secrets
      - the linear relation is correct

Ex: Verifying Local Share Generation

- **Simplification** of MHE operations' structure:
  - **Step 1:** correct **local share generation**
    - Sample random polynomials and create a local share

e.g., $\mathbf{pk}_i = \mathbf{a} \cdot \mathbf{sk}_i + \mathbf{e}_i$

Common polynomial from a CRS

~~Cut-and-Choose~~ Inadequate

    - Generate a proof that
      - $\mathbf{sk}_i$ and $\mathbf{e}_i$ are small secrets
      - the linear relation is correct

Proof systems

Lattice-Based Commitment Scheme

- A lattice-based commitment schemes
  - Hiding by the M-LWE assumption
  - Binding by the M-SIS assumption

$$\vec{\mathbf{t}}_0 = \mathbf{B}_0\vec{\mathbf{r}}$$
$$\mathbf{t}_1 = \langle \mathbf{b}_1, \vec{\mathbf{r}} \rangle + \mathbf{m}_1 \qquad \in \mathscr{R}_q$$

- Can be used for zero-knowledge proofs, e.g.:
  - Linearity
  - Bounds
  - Opening

Proving Local Operation - Client's Proof

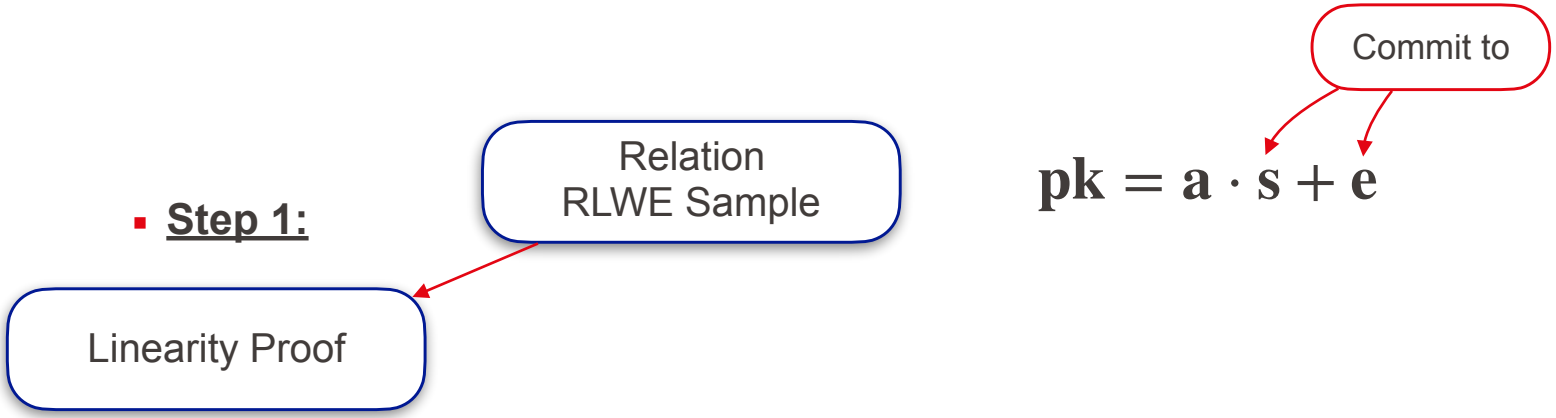
- **Step 1:**

Relation
RLWE Sample

Commit to

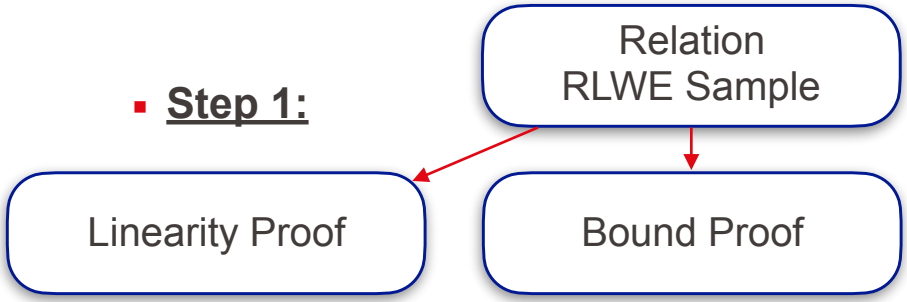
$$\mathbf{pk} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$$

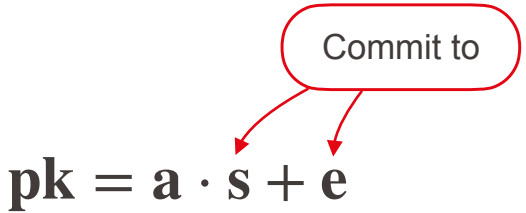
Proving Local Operation - Client's Proof



Proving Local Operation - Client's Proof

- **Step 1:**

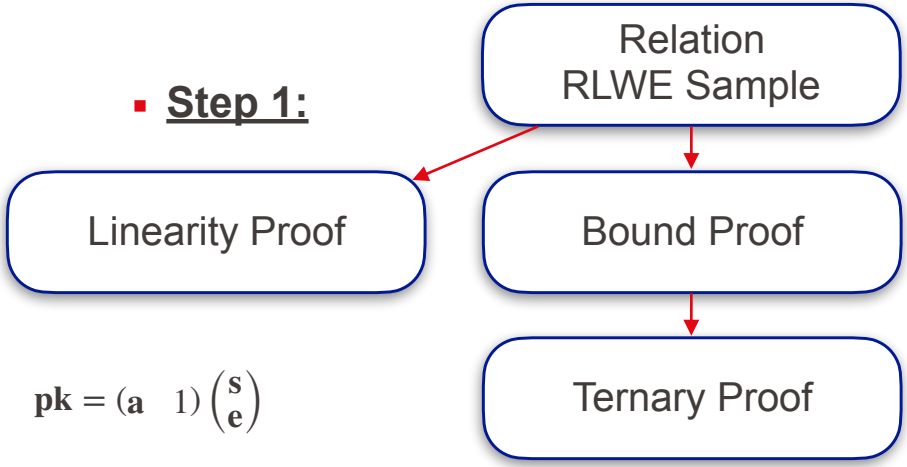


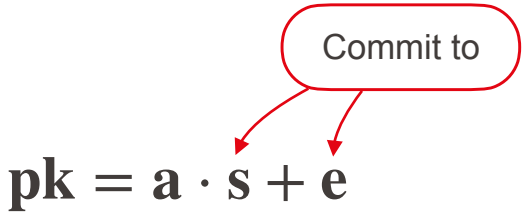


$$\mathbf{pk} = (\mathbf{a} \quad 1) \begin{pmatrix} \mathbf{s} \\ \mathbf{e} \end{pmatrix} \qquad \|\mathbf{s}\|_\infty = 1 \qquad \|\mathbf{e}\|_\infty \leqslant B$$

Proving Local Operation - Client's Proof

- **Step 1:**



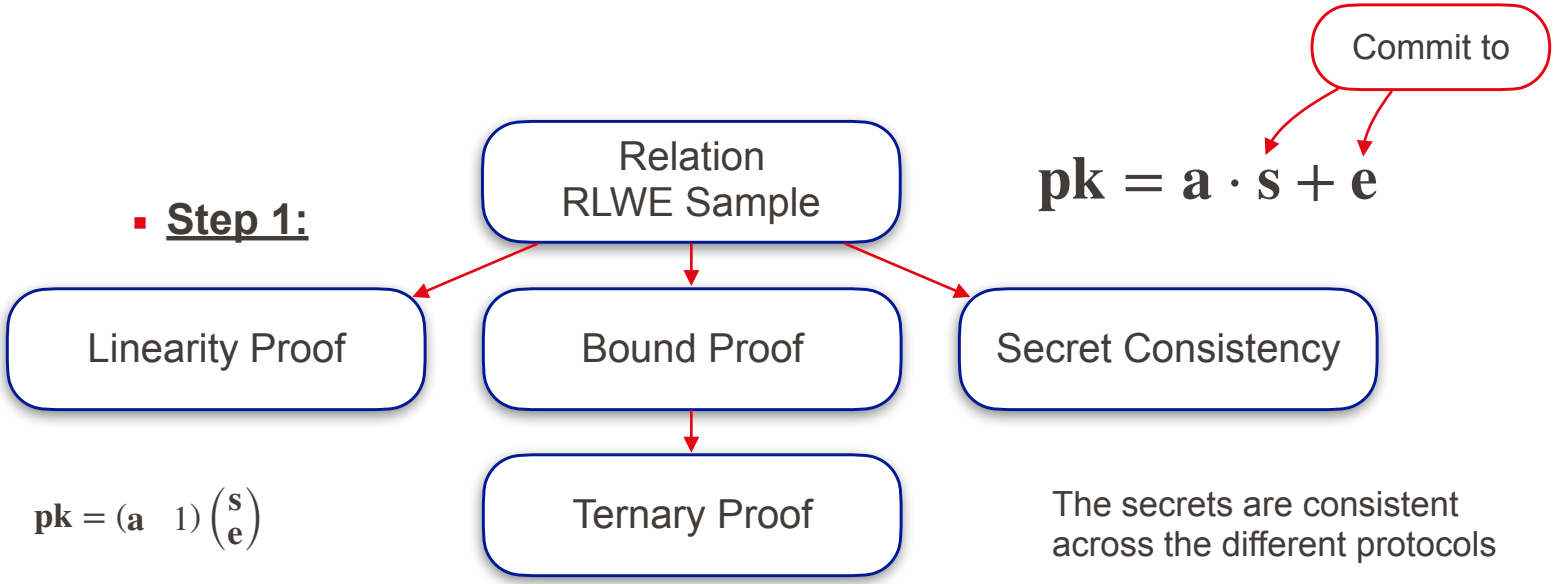


$$\mathbf{pk} = (\mathbf{a} \quad 1) \begin{pmatrix} \mathbf{s} \\ \mathbf{e} \end{pmatrix}$$

$$\|\mathbf{s}\|_\infty = 1 \qquad \mathbf{e} = \sum_i \mathbf{e}_i b_i \qquad \|\mathbf{e}_i\|_\infty = 1$$

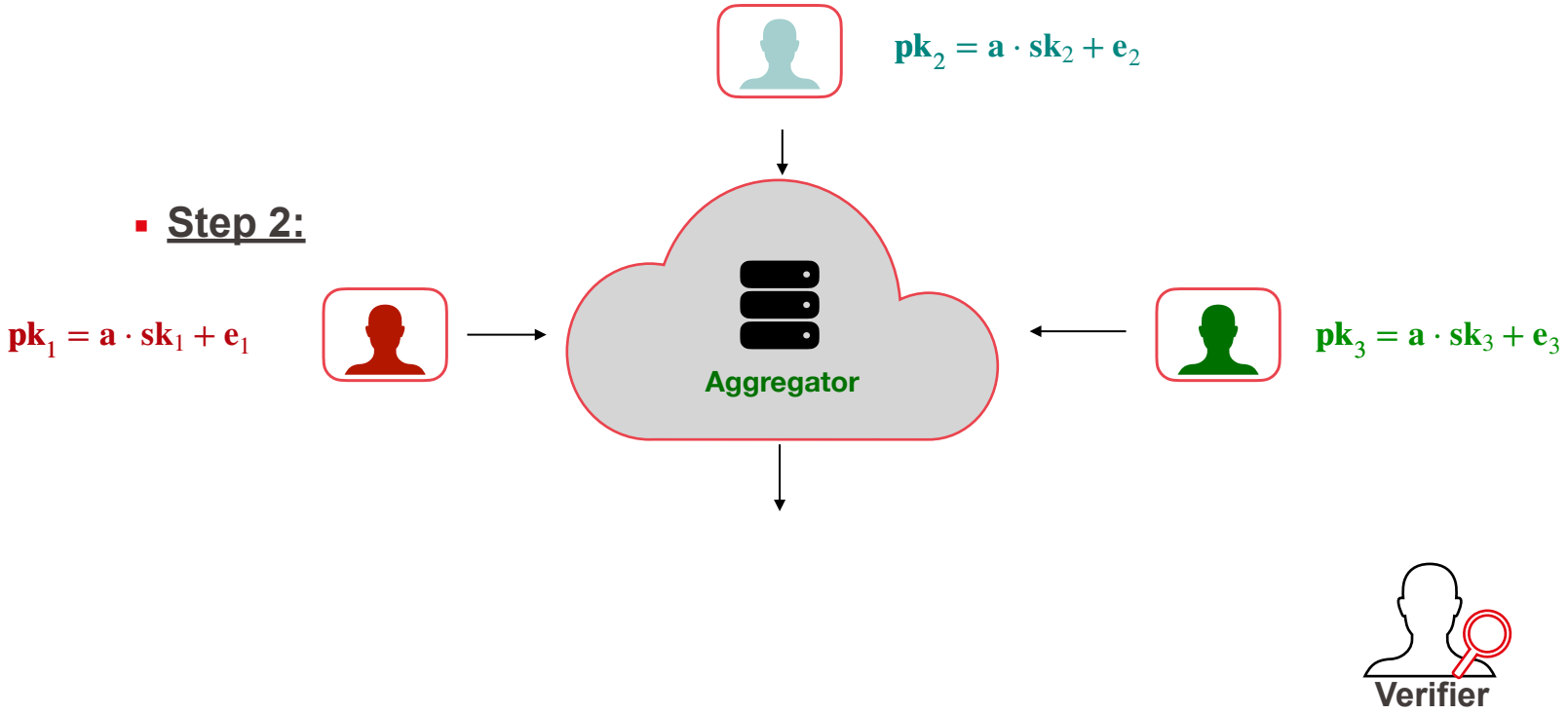
Proving Local Operation - Client's Proof

- **Step 1:**



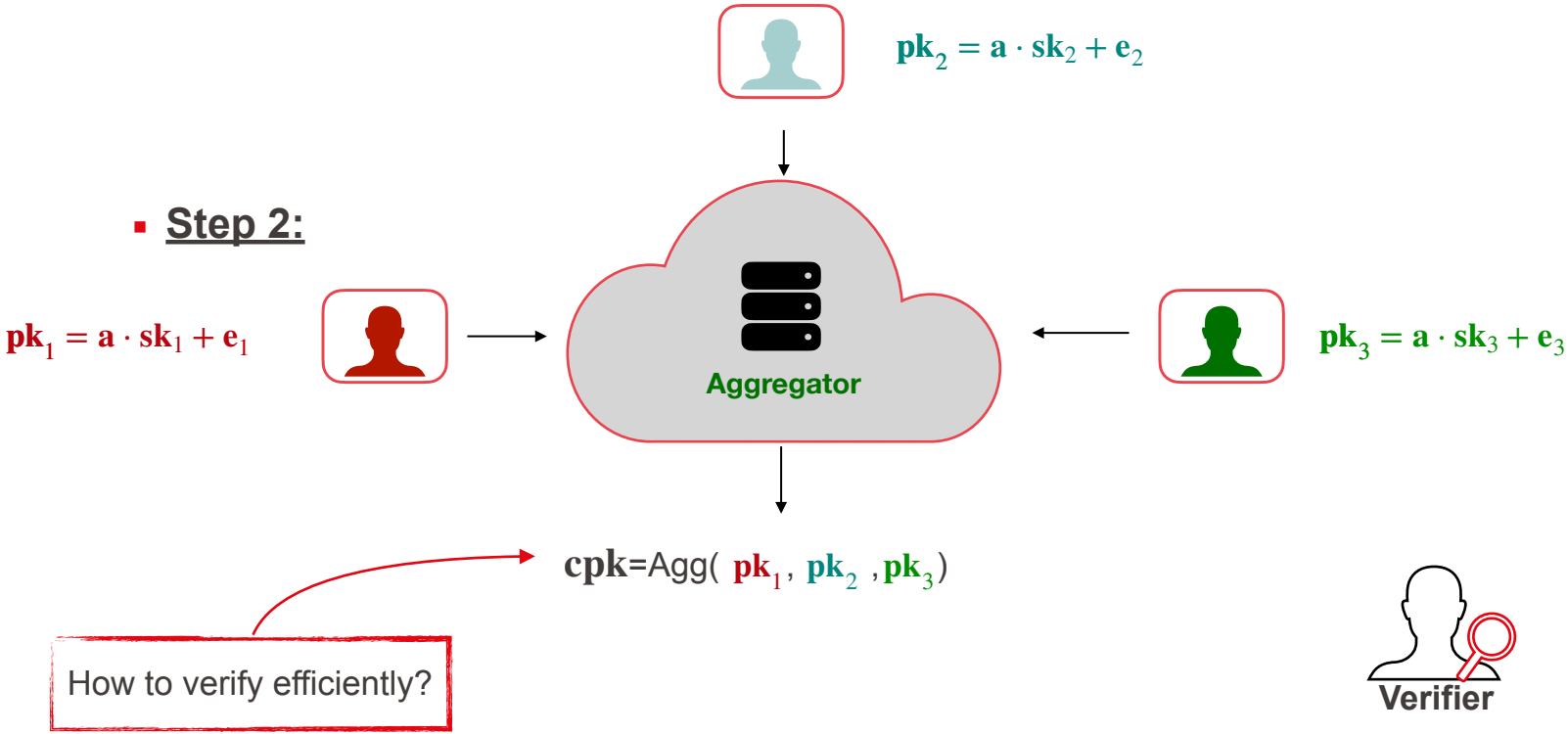
Proving Correct Aggregation - Overview

- **Step 2:**



Proving Correct Aggregation - Overview

- **Step 2:**



Proving Correct Aggregation

- **Step 2:** correct aggregation

Concatenation

$$(pk_1, \ldots,)$$

Verification:
- Concatenate the verified shared of each party

Summation

$$cpk = \sum_i pk_i$$

Verification:
- Needs a different approach

Verifying the Share Aggregation

Prover Verifier

- We rely on the polynomial identity lemma

For a non-zero polynomial P of degree d over a finite field F:

$$\Pr[P(\alpha) = 0, \alpha \leftarrow \$\ F] \leqslant \frac{d}{|F|}$$

- Sum the **evaluation** of the shares on a random point α

Verifying the Share Aggregation

- We rely on the polynomial identity lemma

For a non-zero polynomial P of degree d over a finite field F:

$$\Pr[P(\alpha) = 0, \alpha \leftarrow \$\ F] \leqslant \frac{d}{|F|}$$

- Sum the **evaluation** of the shares on a random point α

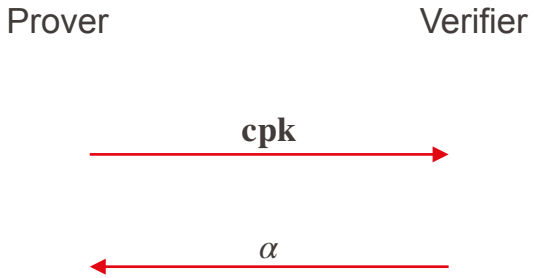
Verifying the Share Aggregation

- We rely on the polynomial identity lemma

For a non-zero polynomial P of degree d over a finite field F:

$$\Pr[P(\alpha) = 0, \alpha \leftarrow \$\ F] \leqslant \frac{d}{|F|}$$

- Sum the **evaluation** of the shares on a random point α



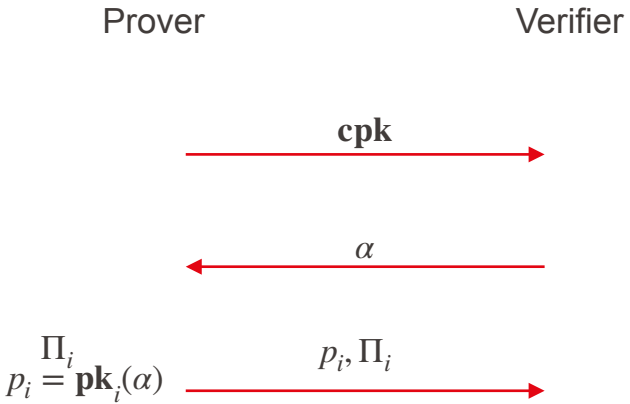
Verifying the Share Aggregation

- We rely on the polynomial identity lemma

For a non-zero polynomial P of degree d over a finite field F:

$$\Pr[P(\alpha) = 0, \alpha \leftarrow \$\ F] \leqslant \frac{d}{|F|}$$

- Sum the **evaluation** of the shares on a random point α



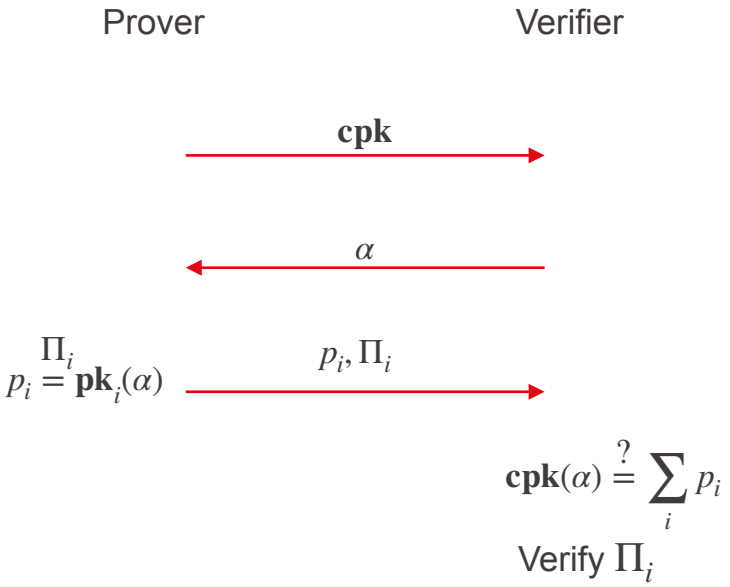
Verifying the Share Aggregation

- We rely on the polynomial identity lemma

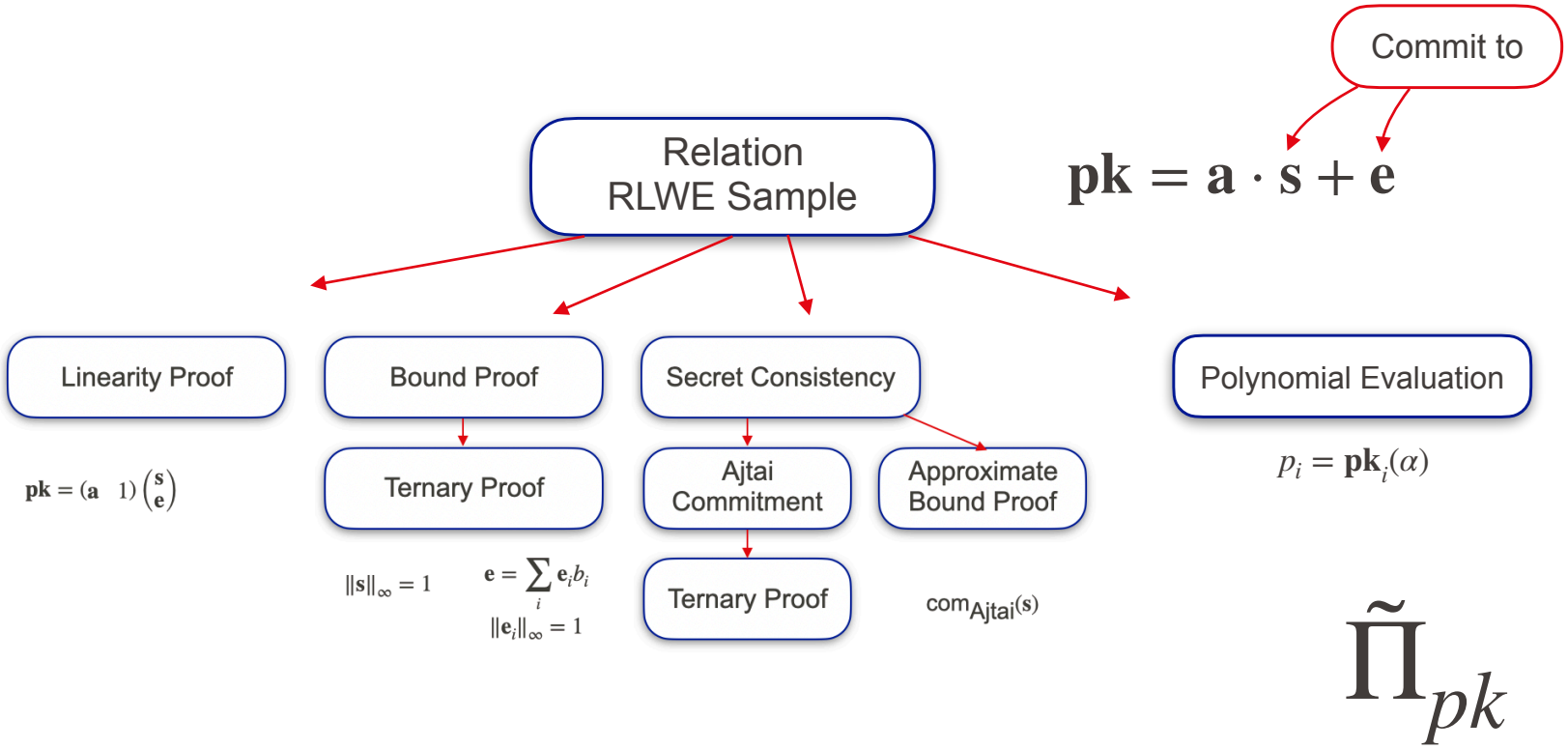
For a non-zero polynomial P of degree d over a finite field F:

$$\Pr[P(\alpha) = 0, \alpha \leftarrow \$\ F] \leqslant \frac{d}{|F|}$$

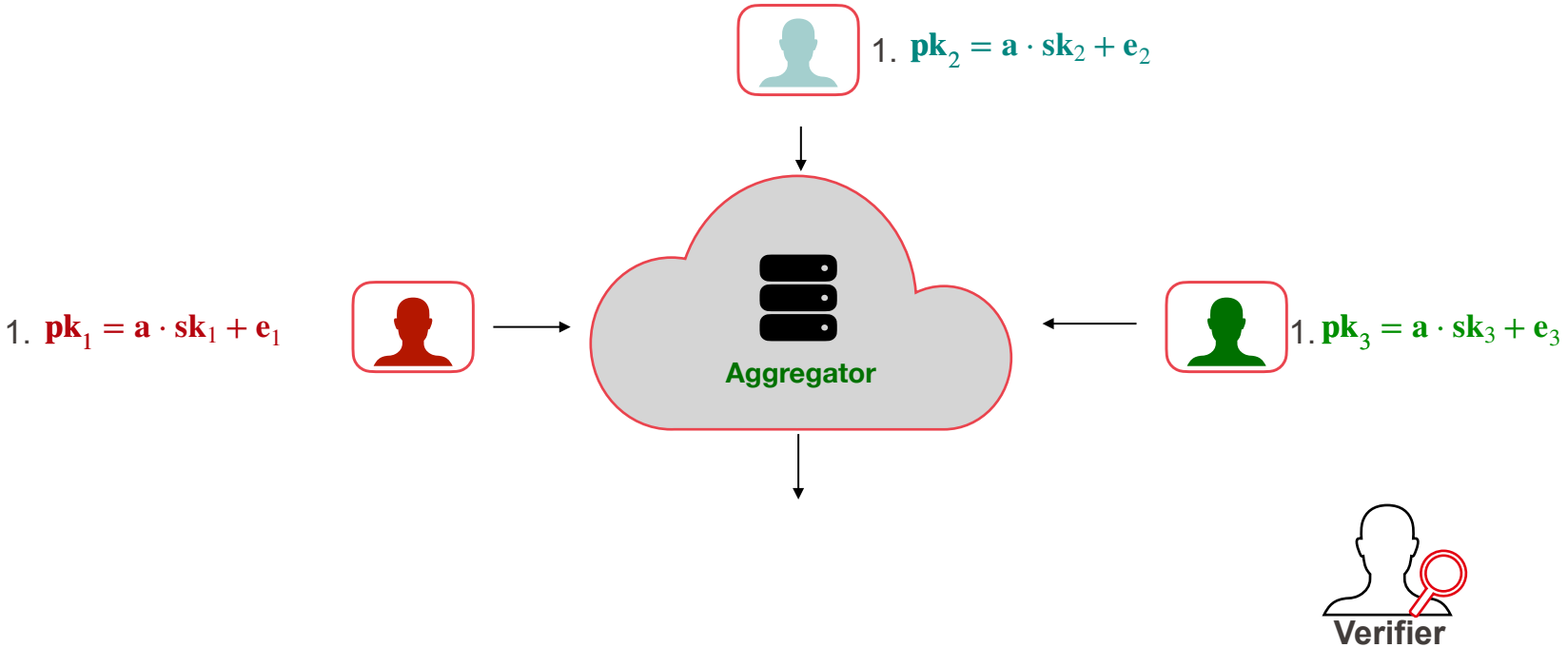
- Sum the **evaluation** of the shares on a random point α



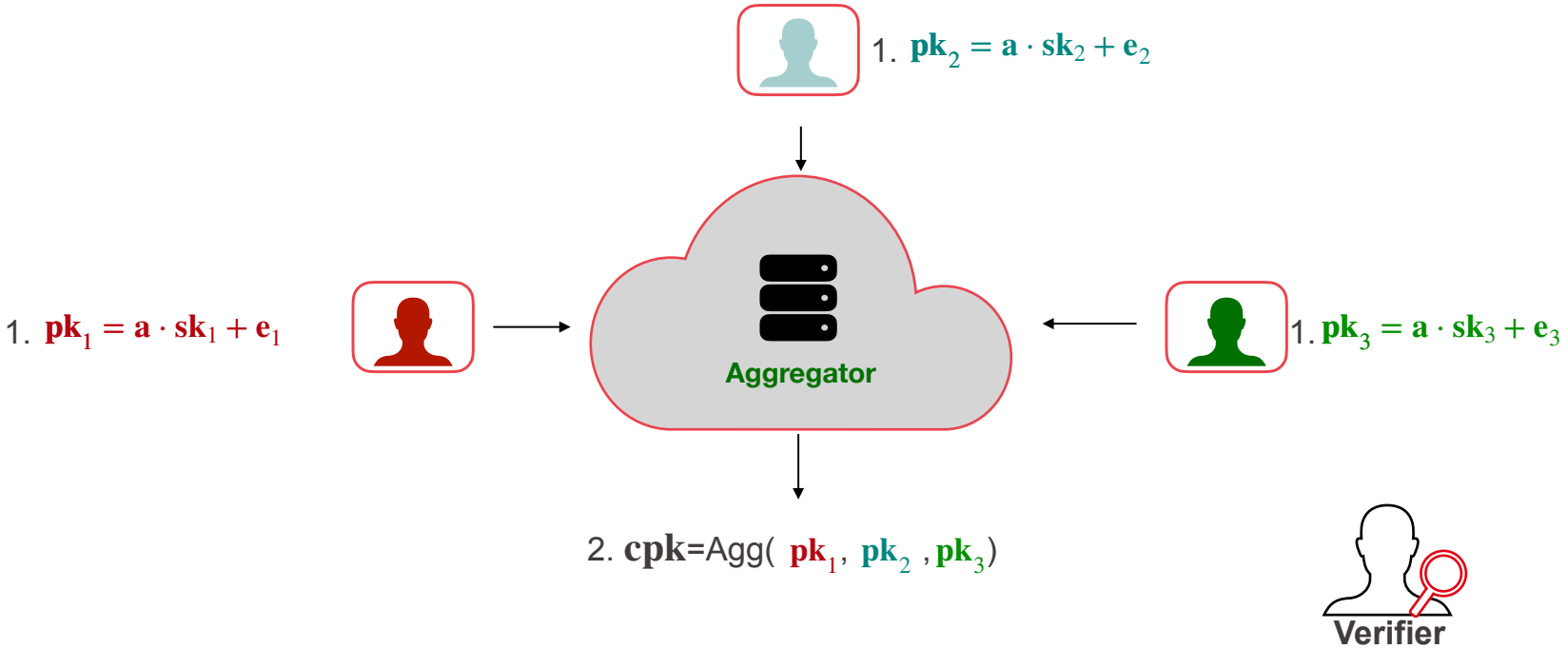
Changing the Client's Proof



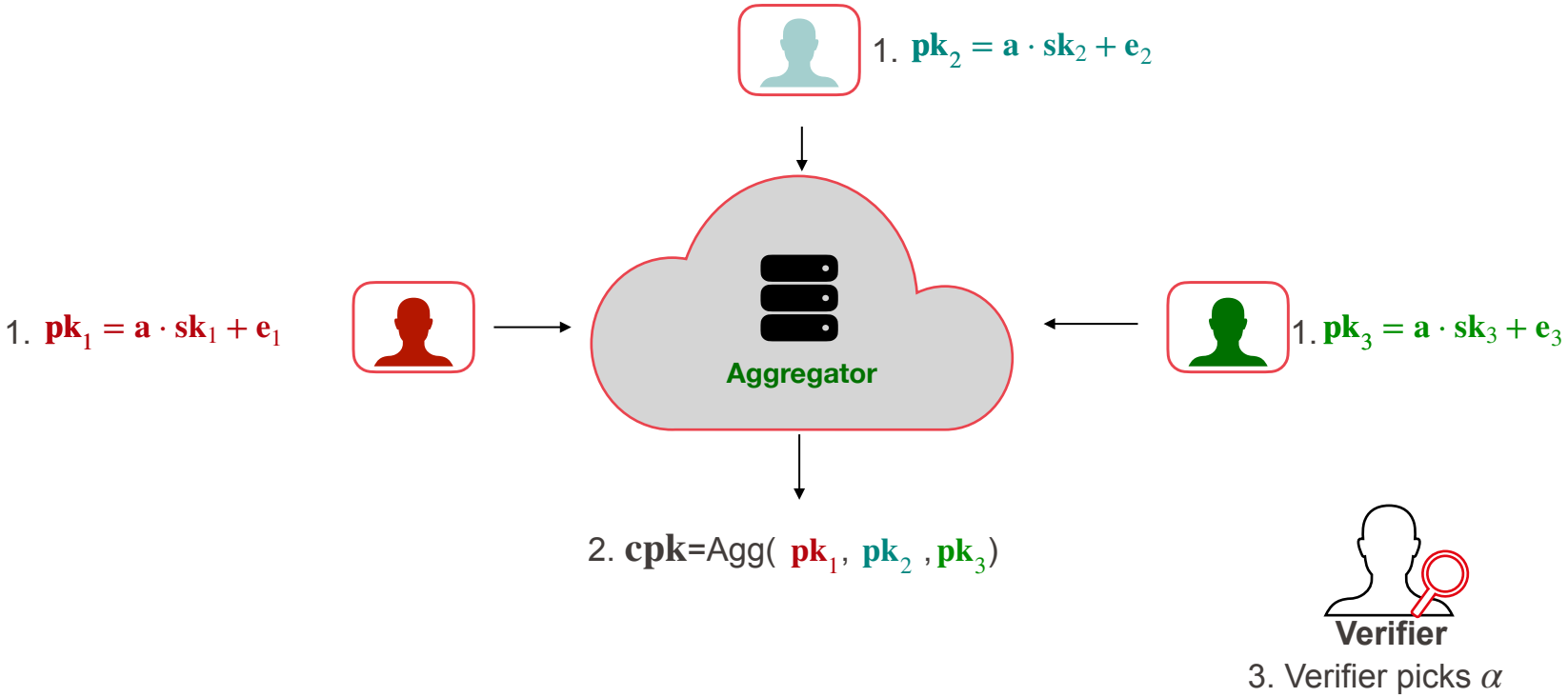
PELTA: Protecting MHE Step by Step



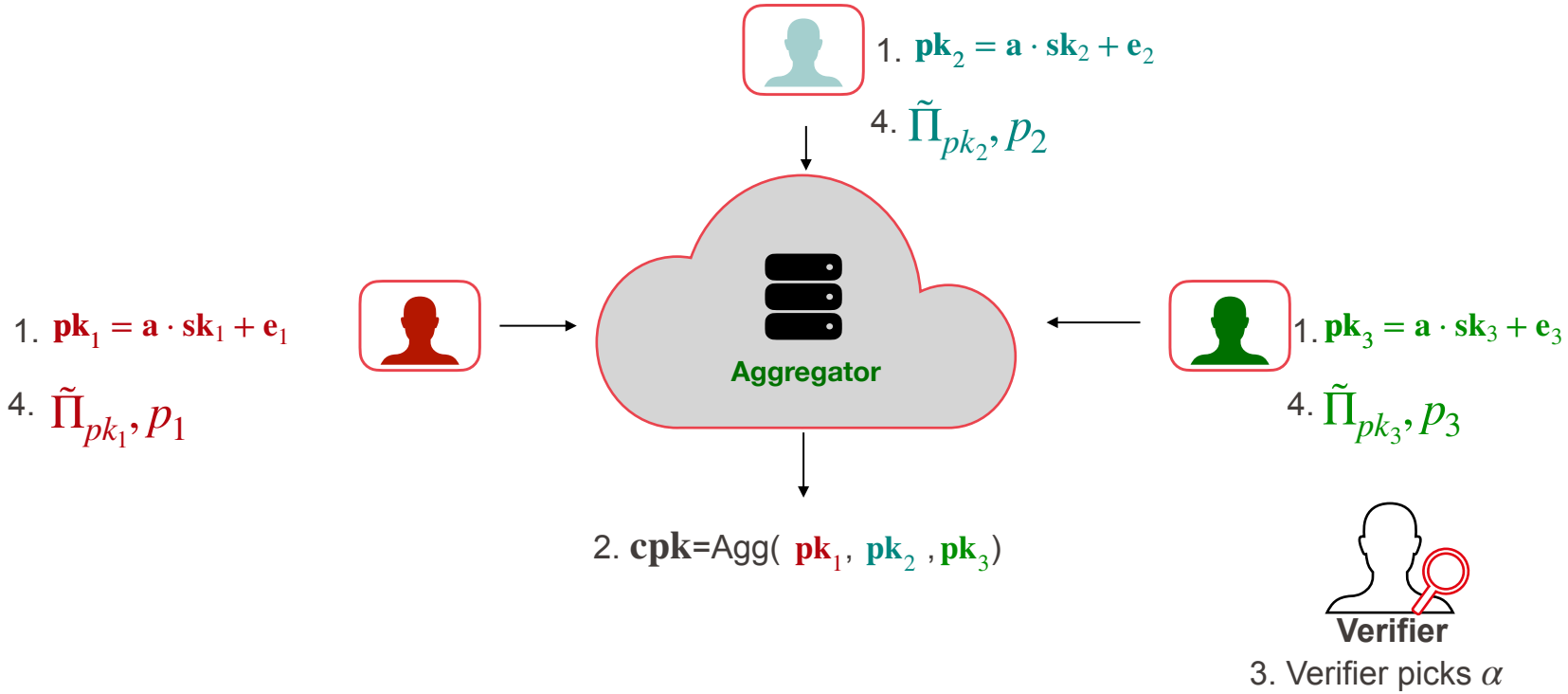
PELTA: Protecting MHE Step by Step



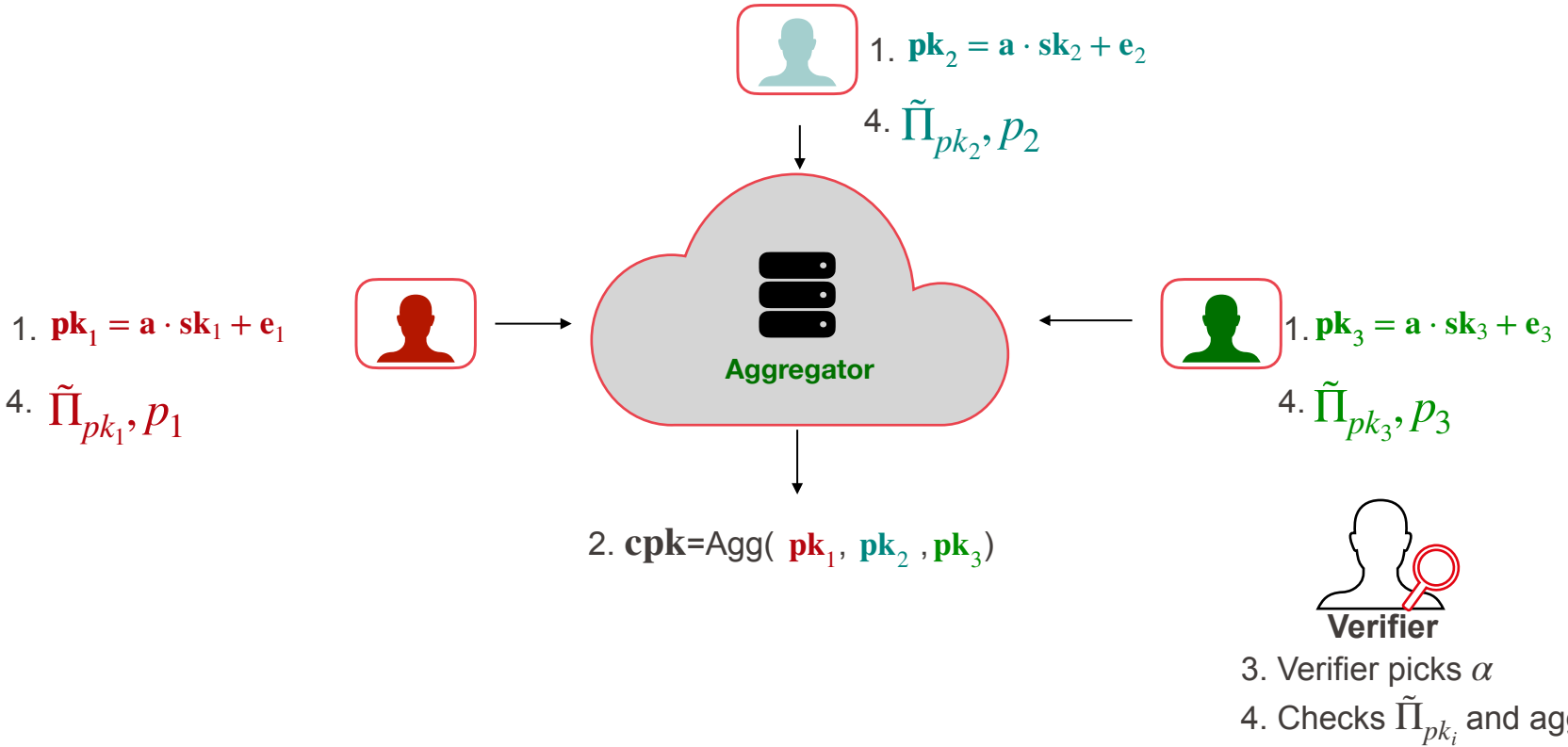
PELTA: Protecting MHE Step by Step



PELTA: Protecting MHE Step by Step



PELTA: Protecting MHE Step by Step



Evaluation

- Local Key Generation

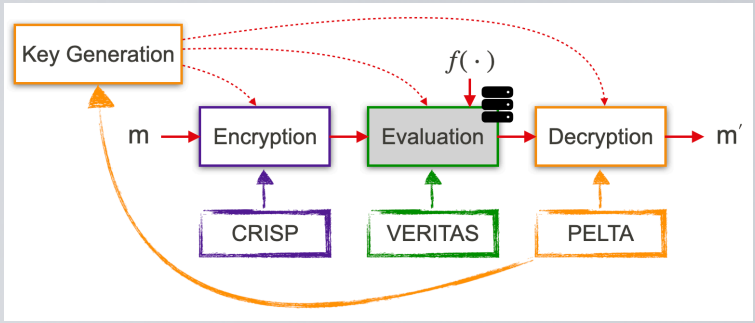
Performance results of the local key-generation protocol with logN=13 over a single sub-ring of the RNS

	Setup	Prover (s)	Verifier (s)	Proof (MB)	λ
Cut and choose	-	463	463	6.4	16
Aurora-based [Bos+20]	>5min	845	312	0.463	128
Ours	**12.2s**	**14.0**	**14.9**	2.05	**128**

Size can be improved using different ring sizes

Public key generation: 5ms - 0.13MB

Summary of PELTA



PELTA

Chatel et al., ePrint 2023/642
CCS'23

CRISP

Chatel et al., Usenix Sec 21

- **Practical solution** for verifying clients' operations in presence of **malicious adversaries**
- Combines **lattice-based commitment** and proof systems with FHE
- More than **one order of magnitude more efficient** that the prior work on similar problems

Conclusion

Thank you

VERITAS

Cloud-Computation Verification
Chatel et al., ArXiv 2022

PELTA

Multi-client Ops. Verification
Chatel et al., CCS 2023

CRISP

Client-Operations Verification
Chatel et al., Usenix Sec 21

- Our solutions are first steps towards efficiently protecting pipelines without losing the FHE functionalities