ETH zürich

PPS pps-lab.com

Integrity Protection Challenges for Real-World FHE

Christian Knabenhans, Alexander Viand, Anwar Hithnawi

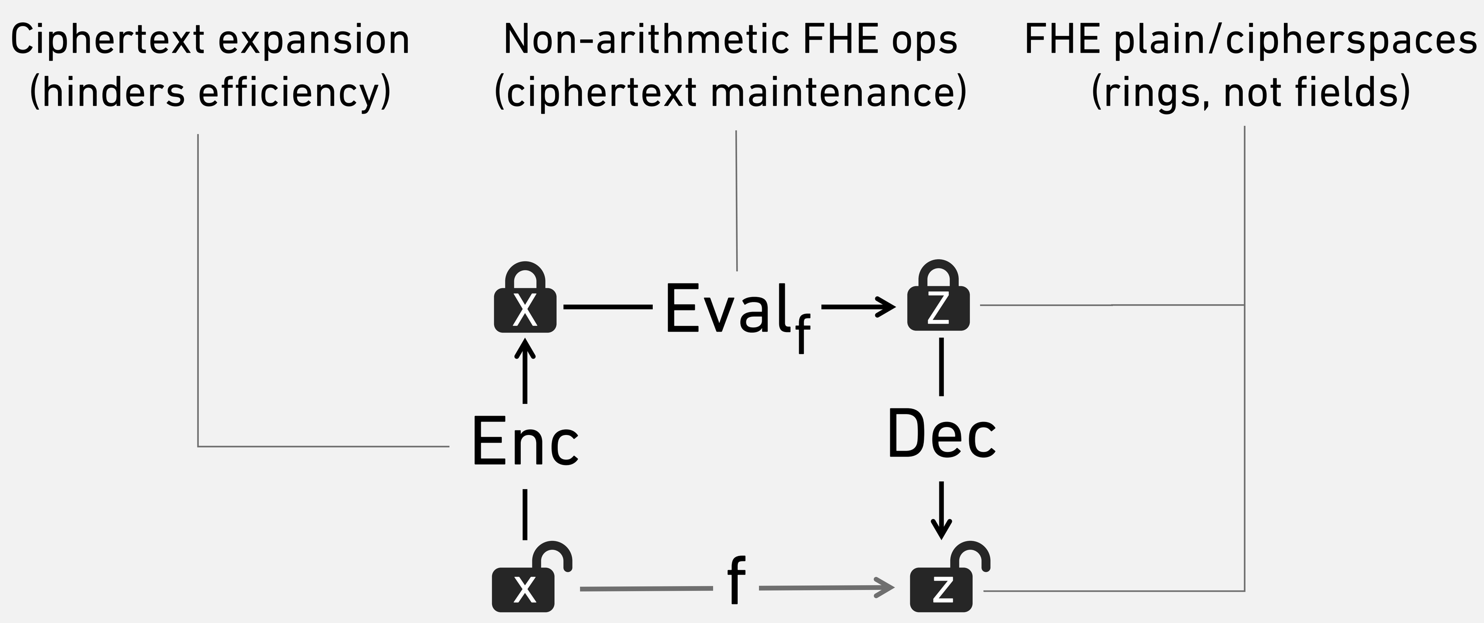
1 Motivation

Fully Homomorphic Encryption (FHE) enables computation on encrypted data, while preserving the privacy of inputs and outputs

FHE is malleable by construction:

- **Loss of correctness** in applications with a malicious server.
- **Loss of privacy** (e.g., through key recovery attacks) for a malicious server with access to some decryption oracle.
- Real-world FHE deployments are inherently at risk of exposing decryption oracles

Why is integrity protection for FHE challenging?



2 State-of-the-Art Integrity Primitives for FHE

Where are current integrity primitives for FHE lacking?

Approach	Expressivity	Efficiency	Concrete Overhead	ZK	Implemented
Veritas [CKPH22]	●●●	●●●	×1.5 - ×50	✗	✓
Proofs over Fields [Gro16]	●●●	●○○	$\times 10^4$ - $\times 10^5$	✓	✓
Hom. Hashing [BCFK21]	●○○	●○○	?	✓	✗
Rinocchio [GNS21]	●●○	●○○	?	✓	Ongoing
Trusted Execution Environments	●●●	●●●	×4 - ×20	✓	✓

→ Lack in expressivity and/or efficiency

Correctness $y = f(x)$

Main goal of current primitives: Server output is exactly the evaluation of the public function f over the client's input

✓ ✓ ✓

Why is correctness alone not sufficient?

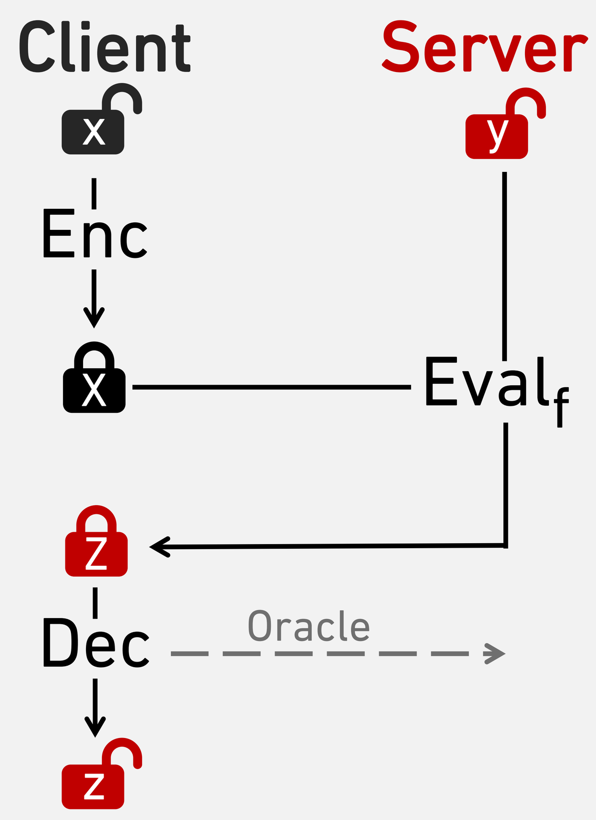
- Correctness has been the primary focus for FHE integrity
- Outsourcing is very often costlier than computing locally on plaintexts
  → **FHE is mostly worthwile in multi-party settings**

3 Integrity Properties for Real-World FHE Use-Cases

In 2-party setting, adversary has potentially malicious inputs:

- Correctness no longer a meaningful property to prove
- Decrypting server output is a privacy risk for the client
- Server could return different results for different queries
- Server may not expend work

→ We formulate **3 desirable properties for FHE deployments**



Property 1: Privacy Protection $\mathrm{Dec}(z) \not\rightarrow x$

Client-Server interaction does not leak the client's private inputs
→ Requires protection against key-recovery attacks and resilience in the face of decryption oracles

? ?

Key recovery protection requires an IND-CCA1 (Non-adaptive Indistinguishability against Chosen-Ciphertext Attacks) FHE scheme

In real-world settings, the adversary is adaptive, which can be thwarted by consistency

Property 2: Consistency $y_1 = y_2 = \dots = y_n$

Server always uses the same inputs for multiple client queries
→ Implies **determinism**, **fairness**, and **non-adaptivity**

?
Proof system needs to be efficiently composable with a commitment scheme for FHE

✓
Wrapper code checks that inputs match the server's committed values

Property 3: Proof-of-Effort $y \neq 0$

Being malicious is at least as expensive as being honest
→ Incentive to correctness for malicious-but-rational adversary

✗
Adversary can choose inputs s.t. circuit is satisfied, but has not been executed

✓
Client gets proof that circuit code was executed correctly

4 Future Directions for FHE Integrity

More useful integrity primitives, which are more:

- efficient (e.g., leveraging hardware acceleration)
- expressive (with native support for FHE)
- composable with other proof systems

[Gro16] J. Groth, "On the Size of Pairing-Based Non-interactive Arguments", in *Advances in Cryptology – EUROCRYPT 2016*
[BCFK21] A. Bois, I. Cascudo, D. Fiore, and D. Kim, "Flexible and Efficient Verifiable Computation on Encrypted Data", in *Public Key Cryptography – PKC 2021*
[CKPH22] S. Chatel, C. Knabenhans, A. Pyrgelis, and J.-P. Hubaux, "Verifiable Encodings for Secure Homomorphic Analytics", arxiv.org/abs/2207.14071
[GNS21] C. Ganesh, A. Nitulescu, and E. Soria-Vazquez, "Rinocchio: SNARKs for Ring Arithmetic", eprint.iacr.org/2021/322