Poster: Verifiable Fully Homomorphic Encryption

Alexander Viand*, Christian Knabenhans*, Anwar Hithnawi
firstname.lastname@inf.ethz.ch

ETH Zurich

***Abstract*—Fully Homomorphic Encryption (FHE) is seeing increasing real-world deployment to protect data in use by allowing computation over encrypted data. However, the same malleability that enables homomorphic computations also raises *integrity* issues, which have so far been mostly overlooked. While FHE's lack of integrity has obvious implications for correctness, it also has severe implications for confidentiality: a malicious server can leverage the lack of integrity to carry out interactive key-recovery attacks. As a result, virtually all FHE schemes and applications assume an honest-but-curious server that does not deviate from the protocol. In practice, however, this assumption is insufficient for a wide range of deployment scenarios. While there has been work that aims to address this gap, these have remained isolated efforts considering only aspects of the overall problem and fail to fully address the needs and characteristics of modern FHE schemes and applications. In this poster, we show the shortcomings of existing FHE integrity approaches, and propose a new notion for maliciously-secure verifiable FHE.**

We then instantiate this new notion with a range of techniques (notably using zero-knowledge proofs), analyzing them and evaluating their performance in a range of different settings. We highlight their potential, but also show where future work on tailored integrity solutions for FHE is still required.

I. Introduction

Fully Homomorphic Encryption (FHE), which enables computations on encrypted data, has recently emerged into practice. Thanks to theoretical improvements, and optimizations in both software and hardware implementations, it is starting to see use in real-world deployments (e.g., the Microsoft Edge Password Monitor). Computing on encrypted data inherently requires malleable ciphertexts (e.g., the addition of two ciphertexts is also valid ciphertext). This malleability raises the issue of *integrity*, as the server can deviate from the computation requested by the client. This has obvious implications for correctness, but can also have more severe consequences, allowing a malicious server to carry out key-recovery attacks [1] and undermining the confidentiality of FHE. So far, most work on FHE schemes and applications has chosen to side-step this issue by assuming a weak adversarial model. However, as FHE is starting to be deployed to protect critical information, we must move beyond these assumptions to a threat model that accurately reflects real-world adversaries.

Honest-but-Curious Assumption. Historically, the FHE research community has extensively made use of the assumption that the server running an FHE application would be honest-but-curious, rather than actively malicious. This assumption may be reasonable in some deployment scenarios (e.g., when FHE is used by trusted institutions cooperating on their own data). However, the necessity to trust the server to this extent is very limiting to the scope of application scenarios, since a violation of the assumption threatens not only correctness but also confidentiality. In addition, this weak threat model does not protect against bugs in the application code, or temporary breaches of an otherwise trusted party. A class of attacks known as *reaction attacks* exploits the interactive nature of real-world deployments to recover the FHE secret key. These exploit the fact that a server can craft a ciphertext that fails to decrypt correctly for certain secret keys, using the client's reaction or lack thereof as an oracle. Practical key-recovery attacks have been developed for all major FHE schemes [2]. Therefore, there is an urgent need to strengthen FHE to maintain strong guarantees in the context of these attacks.

Existing FHE Integrity Approaches. In order to remediate these attacks, a line of approaches has ported techniques from Verifiable Computation (VC) to FHE, with the aim to guarantee that a function was correctly executed [3]. While some of these approaches are concretely efficient, there is a significant gap between the assumptions made by existing work and the way state-of-the-art FHE schemes are used in practice. In particular, existing schemes can only tolerate adversaries limited to verification oracles, much weaker than the decryption oracles present in most real-world settings. While these approaches give robust correctness guarantees, they do not offer significantly stronger confidentiality guarantees compared to standard FHE.

Another line of works aims to construct FHE schemes that achieve indistinguishability against chosen ciphertext attacks (IND-CCA1) [2]. These schemes remain secure even in the presence of decryption oracles. Unfortunately, many of these constructions assume the presence of cryptographic primitives even stronger than FHE, and/or are too inefficient to implement in practice. Additionally, IND-CCA1-security does not imply any correctness guarantee, which is needed for real-world FHE applications.,

Finally, approaches from both of these lines of work are often not compatible with state-of-the-art FHE schemes as implemented and used in practice (and on their way to standardization), and thus of limited value for practitioners.

Contributions. This work is the first to consider integrity in the context of real-world FHE deployment settings, addressing

*Equal contribution

ZKP System	Toy			Small			Medium		
	Setup	Prover	Verifier	Setup	Prover	Verifier	Setup	Prover	Verifier
FHE [No Integrity]	0.003 s	0.002 s	0.001 s	0.807 s	0.011 s	0.009 s	1.053 s	0.014 s	0.010 s
Bulletproofs	-	7569.799 s	552.079 s	-	3957.122 s	278.433 s	-	8697.741 s	575.792 s
Aurora	-	1554.589 s	32.880 s	-	3750.477 s	79.323 s	-	5028.085 s	106.345 s
Groth16	198.640 s	195.941 s	0.002 s	479.222 s	472.711 s	0.002 s	642.470 s	633.741 s	0.002 s
Rinocchio[1]	0.485 s	0.320 s	0.096 s	46.700 s	305.000 s	0.153 s	56.90 s	443.000 s	0.181 s
TEE[2] (Intel SGX)	-	0.154 s	-	-	1.100 s	-	-	1.260 s	-

Table I: Performance results for different instantiations of verifiable Fully Homomorphic Encryption. For FHE, Setup = Key Generation, Prover = Homomorphic Computation and Verifier = Encryption/Decryption

the issue of FHE integrity holistically. This work aims to both highlight the issues arising from the gap between existing notions and real-world scenarios, and to propose efficient instantiations of a new robust notion for FHE integrity that effectively addresses these challenges.

II. Maliciously-Secure Verifiable FHE

We define a new notion of integrity for FHE that captures real-world FHE deployment settings, addressing the issues we identified in our analysis. Existing notions are usually ported from conventional cryptography to FHE in a black-box manner, and often fail to address the specific characteristics of modern FHE schemes and applications. In contrast, we present a natural clean-slate notion of verifiable FHE that composes the standard notion of FHE with modular integrity properties.

Verifiable FHE. Our core notion of verifiable FHE (vFHE) can be seen at the intersection of the notion of IND-CCA1-security for FHE, together with the notion of correctness in the VC literature (against a stronger, IND-CCA1-style adversary). On top of this core notion, we define additional modular integrity notions, outlined below (formal definitions can be found in the extended version of our preprint [4]).

Approximate FHE. Some FHE schemes (e.g., CKKS) operate over floating point numbers, and only guarantee approximate correctness. While more efficient than exact FHE schemes for many use cases (e.g., machine learning), these schemes have been shown to offer even weaker confidentiality guarantees, and integrity is harder to achieve for these schemes.

Server Inputs. Real-world FHE applications are usually not restricted to an outsourced computation setting, but often operate in a client-server 2-party setting, where the server provides input to the computation. These inputs offer a new attack surface for a malicious server, which is why we model them explicitly in our definitions.

Plaintext Inputs. Modern FHE schemes support ciphertext-plaintext operations, which are much faster than their ciphertext-ciphertext equivalents, and are heavily used in practice. We explicitly model these in our notions, which allows us to present a simpler generic integrity construction.

Input Predicates. For real-world applications, a client may want to enforce additional constraints on the server inputs. These predicates are a natural extension of the well-formedness predicate for server inputs, which is a necessary condition to achieve our core vFHE notion.

Server Privacy. We additionally incorporate the idea of circuit privacy from the FHE world, which states that the client is not allowed to learn anything besides the output of the computation. In particular, the server's input should remain hidden from the client.

III. Instantiating Verifiable FHE in Practice

Generic Construction. Previous approaches IND-CCA1-secure FHE relied either on very strong cryptographic constructs (e.g., indistinguishability obfuscation), on non-standard complexity assumptions (later shown not to hold in practice), or on an expensive double-encryption paradigm, Somewhat surprisingly, we show that *all* of our notions (which include IND-CCA1-security) can be achieved by combining any FHE scheme with a generic zero-knowledge proof (ZKP) system. This much simpler construction is made possible by taking advantage of the IND-CPA$^{\mathrm{D}}$-security achieved by most modern FHE schemes, and of the additional specificities of real-world FHE applications (e.g., plaintext inputs).

Concrete Constructions. We instantiate our notion of maliciously-secure verifiable FHE using a variety of different state-of-the-art ZKP systems. In the process, we highlight a series of fundamental challenges in bringing together FHE and ZKP systems, including the mismatch between the rings used in modern FHE schemes and the fields used in the vast majority of ZKP systems. We investigate several approaches to bridge this gap, and explore the trade-offs offered by novel ring-based ZKPs. We evaluate our instantiations on a variety of different workloads and compare them to a hardware-attestation–based approach (FHE-in-TEE) as a point of comparison. We show that verifiable FHE can be practical, but also highlight the need for future work on ZKP systems specifically designed for the unique characteristics of FHE.

[1] We use our implementation of the Rinocchio protocol (https://github.com/MarbleHE/ringSNARK), with our batched-encoding optimization.

[2] We use our implementation of the FHE-in-TEE paradigm (https://github.com/MarbleHE/FHE-in-TEE), with our verifiable delegation to CPUs.

References

[1] I. Chillotti, N. Gama, and L. Goubin, "Attacking FHE-based applications by software fault injections," *Cryptology ePrint Archive*, 2016.

[2] P. Fauzi, M. N. Hovd, and H. Raddum, "On the IND-CCA1 security of FHE schemes," *Cryptology ePrint Archive*, 2021.

[3] S. Chatel, C. Knabenhans, A. Pyrgelis, and J.-P. Hubaux, "Verifiable encodings for secure homomorphic analytics," July 2022.

[4] A. Viand, C. Knabenhans, and A. Hithnawi, "Verifiable fully homomorphic encryption." https://arxiv.org/abs/2301.07041v1, 2023. Extended version (v1).