The Alliance of HE and TEE to Enhance their Performance and Security

Salvatore D'Antonio
Trust Up srl
Naples, IT
Email: salvatore.dantonio@trustup.it

Giannis Lazarou
EXUS Software LTD,
London, UK
Email: g.lazarou@exus.co.uk

Giovanni Mazzeo
Trust Up srl
Naples, IT
Email: giovanni.mazzeo@trustup.it

Oana Stan
Universite Paris-Saclay, CEA, List,
Palaiseau, FR
Email: oana.stan@cea.fr

Martin Zuber
Universite Paris-Saclay, CEA, List,
Palaiseau, FR
Email: martin.zuber@cea.fr

Ioannis Tsavdaridis
Cooperative Bank of Epirus L.L.C.,
Ioannina, GR
Email: itsavdaridis@epirusbank.gr

***Abstract*—While protection of data at-rest and data in-transit can be achieved using standard algorithms and technologies, the protection of data in-use is still, to a large extent, an open issue. Homomorphic Encryption (HE) and Trusted Execution Environment (TEE) are among the most popular approaches to shield computations. The former ensures high security guarantees but it suffers from a significant overhead. The latter, instead, provides lower execution time but it is affected by security drawbacks. In this paper, we propose `SOTERIA`, a privacy-preserving computation solution that combines HE and TEE to mitigate their limitations. The approach foresees the execution of sensitive processing with homomorphic encryption and the usage of a TEE to perform switches between different homomorphic cryptosystems. In fact, there are different kinds of computation algorithms where the co-existence of linear and non-linear functions makes the HE-based processing even more onerous. `SOTERIA` is developed in the context of the *ENCRYPT* project and will be validated in a use case of financial data processing.**

I. Introduction

In the last decade, the field of privacy-preserving computing techniques has gained momentum among the research and industrial communities. In fact, the protection of data-in-use is of paramount importance to ensure information security in the most vulnerable phase, i.e., during its computation in a non-trusted environment. In such a vulnerability window, malicious actors could gain access to system memory and thus to data in clear form. Different techniques are available to achieve the protection of data-in-use via computation over encrypted data, the most popular examples being: *i*) Homomorphic Encryption (HE); *ii*) Secure multi-party computation (MPC), *iii*) Functional Encryption (FE). A totally different approach is offered by Confidential Computing. According to the Confidential Computing Consortium, the protection of data in use is made by performing computations in a hardware-based Trusted Execution Environment (TEE). Each approach has its own advantages and disadvantages, thus the adoption of a specific technique highly depends on the use case requirements.

a) HE: The security model of HE foresees the data owner encrypts data in a trusted infrastructure and then outsources the processing — on ciphered data — in the non-trusted platform. The security guarantees are strong considering that data is always encrypted and that keys never leave trusted premises. A specificity of HE lies in the fact that different cryptosystems are suited to different kinds of computations. For instance, BGV [1], BFV [2], [3] or CKKS [4] cryptosystems fit well with linear processing over a large amount of data points in parallel, while the TFHE [5] encryption scheme is more suitable for non-linear processing over a single data point at a time. Switching between both types of cryptosystems can therefore be crucial for some applications. This has been well-known for some time and for more information on the differences between the cryptosystems see for instance [6].
However, HE is notoriously slow. Although recent work [7], [8], [9] on specific use-cases and using specialised algorithms has proven that it can be practical, its performance overhead can be prohibitive. Additionally, while it would be ideal to be able to switch between cryptosystems to increase the performance and the range of applications, existing methods [6] are not yet practical.

b) TEE: In a TEE, protection is achieved by preventing access to data (as opposed to transforming/dispersing it). TEEs create secure areas of a computing device where sensitive data is isolated from the rest of the device software and hardware. The security model of a TEE, instead, foresees the data owner pushes data in a trusted — CPU-secured — memory area in the non-secure environment where it will be protected from external attacks by means of hardware. The computation is then performed in the TEE on clear-text.
However, the trust model of a TEE is weaker than that of HE. The data owner must trust the TEE manufacturer and the software running in the secure enclave. Moreover, protection against side-channel attacks - while inherent when performing computation over homomorphic ciphers - needs to be fine-tuned to every different computation inside the TEE.

c) HE + TEE: In this paper, we present `SOTERIA`, a solution that combines TEE and HE to mitigate their limitations and draw on their respective strengths. `SOTERIA`

achieves several properties that would be unattainable without this marriage of TEE and HE:

- It performs efficient and protected crypto-scheme switches inside a TEE. We can therefore optimize the homomorphic processing of algorithms containing both linear and non-linear functions, which perform better with different crypto-schemes.
- Because the noise inside ciphertexts is refreshed inside the TEE, it allows the AI model that we use to have an arbitrarily large multiplicative depth, with no effect on parameter size and therefore on performance.
- It relies on HE for most of the computation time, in this way we reduce the time window in which data would have been exposed to side-channel attacks if it was processed inside the TEE.
- It is protected against the threat of a corrupted TEE through a "flooding" process that makes it impossible to distinguish real data from fake data.
- It achieves generic side-channel attack protection *whichever* computation is done on the client's data.

The SOTERIA solution will be validated in the context of the ENCRYPT project on the financial case study provided by the *Cooperative Bank of Epirus* (EPIBANK) and the *EXUS Software LTD* company. The former is a government banking agency that offers personal banking, loans, venture capital, and online banking services. The latter is a software publisher that develops solutions for banks and financial institutions, which supported EPIBANK to set up an AI-driven application to better manage its debt collection services offered to its clients and shape its overall strategy. The AI processing includes linear and non-linear processing on very sensitive clients' data, which requires the adoption of a privacy-preserving solution. The HE alone is not a viable solution since it would impact the performance too much, while the TEE alone would expose the data to side-channel attacks for the entire processing period and rely on an assumption of trust toward the TEE's manufacturer. We will demonstrate how SOTERIA helps to address the previously-mentioned limitations.

d) Paper outline.: The remainder of this work is organized as follows. Section II overviews background knowledge. Section III presents the driving case study and the related requirement elicitation activity. Section IV discusses the threat model set for this work. Section V presents the design of the SOTERIA solution, illustrated in Figure 1. Section VI discusses the related literature. Finally, Section VII concludes the document.

II. Preliminaries

A. Homomorphic Encryption

In 2009, Gentry [10] made a breakthrough in cryptography by proposing the first Fully Homomorphic Encryption (FHE) scheme, allowing to perfom, in theory, any type of computation over encrypted data. That is, Gentry specified a homomorphic encryption scheme E that computes $\mathsf{E}(\mathsf{m}_1 + \mathsf{m}_2)$ and $\mathsf{E}(\mathsf{m}_1 \times \mathsf{m}_2)$ from encrypted messages $\mathsf{E}(\mathsf{m}_1)$ and $\mathsf{E}(\mathsf{m}_2)$. Then, many leveled HE and FHE schemes have been proposed in the literature [11], [12], [13], [14], [15], [16]. In practice, a public key homomorphic encryption scheme HE = ($\mathsf{HE.Keygen}$, $\mathsf{HE.Enc}$, $\mathsf{HE.Dec}$, $\mathsf{HE.Eval}$) is defined by a set of probabilistic polynomial-time algorithms with respect to the security parameter k:

- $(\mathsf{pk}, \mathsf{evk}, \mathsf{sk}) \leftarrow \mathsf{HE.Keygen}(1^{\mathsf{k}})$: outputs an encryption key pk, a public evaluation key evk and a secret decryption key sk. The evaluation key is a key that is made public and is necessary for some computations depending on the cryptosystem used.
- $\mathsf{c} \leftarrow \mathsf{HE.Enc}_{\mathsf{pk}}(\mathsf{m})$: encrypts a message m into a ciphertext c using the public key pk.
- $\mathsf{m} \leftarrow \mathsf{HE.Dec}_{\mathsf{sk}}(\mathsf{c})$: decrypts a message c into a plaintext m using the secret key sk.
- $\mathsf{c_f} \leftarrow \mathsf{HE.Eval}_{\mathsf{evk}}(\mathsf{f}, \mathsf{c}_1, \ldots, \mathsf{c}_{\mathsf{k}})$: evaluates the function f on the encrypted inputs $\mathsf{c}_1, \ldots, \mathsf{c}_{\mathsf{k}}$ using the evaluation key evk.

Importantly, the Enc operation introduces a noise e inside the ciphertext for security purposes. This noise increases every time the $Eval$ primitive is called on the ciphertext. If the noise grows to be too big, the underlying plaintext can no longer be recovered with the secret key.

The BGV/BFV/CKKS levelled schemes are more appropriate to use in the cases where the computation to be evaluated in the homomorphic domain has a small multiplicative depth (i.e. the maximal number of successive multiplications applied on a ciphertext) which is known in advance. Moreover, due to their batching capabilities, they can perform parallel computation in a SIMD (Single Instruction, Multiple Data) fashion resulting in an amortized computational time i.e. per element of encrypted data. On the other hand, thanks to an operation called bootstrapping, the TFHE approach is more appropriate to use for the case of a deep (large multiplicative depth) and unknown at setup phase computation circuit. TFHE is also very much suited to the case of the computation of a complex non-linear function (such as a sigmoid) because its bootstrapping operation implements a Look-Up Table (LUT) operation.

Nowadays, we can mix and switch between several HE schemes (e.g. BGV, TFHE, CKKS, etc.) using the recent *theoretical* CHIMERA framework [6]. This solution allows to combine between the ciphertext representations of the three most popular FHE schemes based on Ring-LWE, using an unified plaintext space over the Torus. However, this hybrid framework remains for now mainly of theoretical interest since the required transformations are easier to perform in one direction than in the other (e.g. from BGV to TFHE, from CKKS to TFHE) and, most important, they come with a high computational overhead.

As for the application of the homomorphic techniques to the private inference step of neural networks, notable work includes, in a non-exhaustive way, CryptoNets [17], DiNN [18], nGraph-HE [19], LOLA [20], TAPAS [21], Faster CryptoNets [22].

B. TEE

A Trusted Execution Environment (TEE) [23] [24] is a protected environment of execution, in which highly sensitive operations can be executed in isolation from the main operating system and other applications that are considered untrusted. The TEE provides a secure area for running trusted applications, also known as enclaves or secure enclaves. These applications have their own secure memory space, access to hardware resources and cryptographic functions, and can be independently verified and authenticated. One of the most well-known examples of a TEE is Intel SGX (Software Guard Extensions) [25]. SGX enclaves provide hardware enforced confidentiality and integrity guarantees for running pure computations. The instructions running inside enclaves are responsible for the automatic encryption and decryption of the enclave's memory contents during the read and write operations. The enclave memory is protected from unauthorized access from the host including machine administrators with physical access. The processor contains the enclave encryption keys which are inaccessible even to components running with higher hardware privileges including the kernels, OSs, or other hypervisors. By limiting the application's Trusted Computing Base (TCB) to the CPU and CPU-Cache, SGX provides unprecedented confidentiality and integrity guarantees against malicious OS kernels and supervisor software. Intel equipped SGX with a Remote Attestation feature that ensures integrity verification of enclaves using a secret key fused into the processor which internally is used to derive other keys. The derived keys are used to build enclave attestations, creating a quote with a signature on the whole contents of the enclave at its creation. An external attestation service confirms that a given enclave runs a particular piece of code on a genuine Intel SGX processor, which can then be trusted by other components.

III. The Financial Case Study Requirements

The design of the SOTERIA solution has been driven by the financial case study belonging to the ENCRYPT project. The *Cooperative Bank of Epirus* (EPIBANK) is a government banking agency that offers personal banking, loans, venture capital, and online banking services. EPIBANK is supported by the EXUS company — a software house that develops solutions for banks and financial institutions — to set up an AI-driven application to better manage its debt collection services offered to its clients and shape its overall strategy.

A. The need for confidentiality

The functionalities offered by this AI-driven solution include among others: client stratification, behaviour forecasting and overall scoring of the bank agents responsible to handle each client. Since these functionalities are tailored specifically to the needs of EPIBANK, clients' historical data (over a long period of 0.5-1 year) has to be delivered to EXUS, so that EXUS will be able to develop and train its AI-models. EPIBANK wants confidentiality guarantees about the actual identity of the person or any other sensitive data related to the person that could lead to its actual identification. Simple pseudo-anonymization techniques are not sufficient considering that the processing will occur in a non-trusted platform and that pseudo-anonymization has been proven countless times to be largely ineffective. At the same time, EPIBANK cannot be expected to have (and manage) the resources to train its own AI models.

B. The AI model

We want to be able to address this use-case for the widest range of possible AI models and therefore not restrict our approach to a specific one. We only assume two things of the model and its inference phase:

- there are one or more *linear* phases during the inference. This phase consists of a scalar product computation. In the case of a neural network for instance, this phase corresponds to the weighted sum of the activation values at any given layer.
- there are one or more *non-linear* phases during the inference. This phase consists of the computation of a non-linear function over some data. In the case of a neural network this corresponds to the activation function[1] which can be a sigmoid function or a ReLu for instance.

C. A cryptography-only solution

There are a number of cryptographic solutions that can be designed to solve this specific type of problem. The three main methods are the following:

- Homomorphic Encryption (HE). As presented in more detail in Section II-A, HE is a cryptographic method to allow the evaluation of functions over encrypted data. Its main drawbacks are (1) the performances and memory overhead that it incurs; and (2) the fact that some operations are simply too complex to implement in a practical manner. Although any function can *theoretically* be evaluated using HE binary gates, for complex applications such as the one that interests us here, the performance cost is prohibitive. However, despite its limitations, as we will argue here, this solution remains interesting to be applied for the inference of the AI model. By fine tuning the underlying homomorphic cryptosystems and optimising the homomorphic evaluation, the performance costs could be reduced to acceptable levels. The main idea is to use the strengths and advantages of each homomorphic cryptosystem for specific parts of the computation and switch (in a secure manner with the help of the TEE) between the different homomorphic schemes. As such, BFV/BGV or CKKS could be successfully applied to the linear part of the inference step (by batching the data for performing the scalar product). Afterwards, for the non-linear part in which Boolean logic is needed and more complex operations (e.g. comparison, argmax, etc.) are to be performed, one can use TFHE and its boostrapping

[1]See definition and examples here: https://deepai.org/machine-learning-glossary-and-terms/activation-function

method or even more recent optimisation approaches such as Functional Bootstrapping [26].

- Functional encryption (FE). Although they do not introduce the name itself, [27] formalize the notion for the first time in 2010. In short, while HE evaluates a function over encrypted data and provides an encrypted result, FE's purpose is to evaluate *and* decrypt over encrypted data. This means the overall result is obtained in the clear domain, making it unfit for this specific use-case.
- Multi-Party Computation (MPC). In general, MPC was designed to implement privacy-preserving applications where multiple, mutually distrusting parties cooperate to compute a function over data distributed across participants. Since the inception of MPC as a theoretical concept, the cryptographic community has developed a vast number of efficient applications, through the means of both *generic* (allowing for any kind of computation) and *specialized* (designed specifically for a given set of computations) protocols. All of the MPC techniques have in common a reliance on an online communication phase between the two (or more) parties involved as the computation is going on. In our case, we want to allow for a single request from the financial institution and no further communication and therefore rule this method out.

D. A TEE-only solution

One possible solution for our use-case is for the AI service provider to host a TEE as described in Section II-B. The financial institution can send its data over to the TEE using classical public key cryptography. The TEE decrypts the data and computes over the machine learning model that the AI service provider provided. Then the result is encrypted using the public key from financial institution and sent back. If the machine learning evaluation does not require too much memory usage, this can be a viable stand-alone solution. However it suffers from two main flaws. Because the hardware is proprietary, one flaw is the trust that one needs to have that the manufacturer did not, in fact, introduce a back-door in their product. The second flaw is that, contrary to HE, side-channel attacks are very much possible over the TEE. Therefore one would need to protect against such attacks and adapt the countermeasures every time the AI model to compute changes. This requires constant care and a good amount of security know-how.

E. The need for a TEE-HE hybrid solution

Of all the cryptographic methods presented above, HE is the one that received the most development [28]. In terms of security, HE can meet requirements of the mentioned use-case. However, the cost to pay in terms of performance is high and it needs optimization techniques to become practical. Besides the performance penalty, there are two additional relevant aspects impairing wide acceptance of this cryptographic technique by the industrial community, i.e., the *Cipher Text Expansion* (CTE)[29] and the *Unverifiable Conditionals* (UC) [30]. The combined use of HE with TEE can mitigate these issues. The TEE, in fact, plays the role of a trusted area where verify conditionals and perform the HE ciphering/deciphering without the need of doing everything in the data owner premises. At the same time, while a TEE-only solution can be practical from the performance perspective, it has security drawbacks as well:

- One drawback is that a TEE is subject to side-channel attacks (e.g. [31], [32]). These are attacks that infer some things on the cryptographic keys or the data handled by the TEE from power consumption or processing time for instance (other methods using sound for instance exist as well). These attacks *can* be countered but they require specific designs for every computation that is run. Therefore, whenever the machine learning model from the AI service provider changes (which is quite often in these cases), the TEE needs to be protected against side-channel attacks by a computer security expert.
- The second drawback is that we need to trust the TEE completely with all of the data, both from the financial institution and the machine learning model of the AI service provider. This is - one could say - the point of a *Trusted* Execution Environment. Still, we would like to explore ways to protect against a TEE that does not completely function as intended. We aim to provide, on top of the trust that we give the entity that creates the TEE (a private company which may have ulterior motives), some added measure of trust linked to secure cryptographic standards.

For this reason, we want to provide a solution using a hybrid model (HE + TEE) that draws on the strengths of both technologies to provide a *practical* and *secure* solution to the use-case.

IV. Threat model

a) Threats from inside the AI service provider server.: The objective of this work is to provide protection of *data-in-use* against attackers having high privileges. A *privileged attacker* is an entity (human or not) which is able to control the hardware/software infrastructure used to run the applications on the hosting platform (e.g., the Cloud). This typically includes: server nodes, host OS, virtualization frameworks, and general purpose system software. The most dangerous profile of privileged attacker is the *malicious insider*. This category includes system administrators of the cloud provider, who have a privileged role in the infrastructure. In addition to malicious insiders, it is also important to note that multi-tenancy opens the door to a potentially large number of privileged attackers, represented by users with (originally) normal privileges. As an example, a tenant with a legitimate user account (with normal privileges) might maliciously exploit a vulnerability in the VM/container software, access the –typically shared– host OS, and *escalate* to *rootly* powers. Regardless of how super powers were gained, privileged attackers have full control of the entire cloud stack, and can thus use many potential *vectors* to exploit systems' weaknesses and steal or alter sensitive data (and/or code) while it is *in-use*.

b) Threats from the TEE.: In an approach in which the data is sent encrypted to the TEE for decryption and this one continues the computation over clear data, the TEE is considered by default a trusted entity. However, this is not the assumption we made for our present work and we include the TEE in our threat analysis. As a reminder, there are several levels of trust that one can give an actor in any cryptographic protocol.

- Honest and Blind. An entity can be trusted to be completely honest to the point where they do not look at data that is given to them in clear form.
- Honest but Curious. An entity can be honest in the sense that it performs all operations as the protocol dictates. However it will try to extract any information it can from what it is shown.
- Rational. A rational adversary can stray from the established protocol but will only do so if that gives it additional information *and* it has a high probability of not being caught.
- Malicious. A malicious adversary can do anything to prevent the execution of the protocol and extract as much information as possible.

The SOTERIA architecture offers protection in the case of an honest-but-curious TEE, one which does not stray from the protocol but feeds back some of what it sees to an actor (e.g. the manufacturer) trying to glean information either on the AI service provider model or on the data from financial institution.

c) Threats from side-channel adversaries.: The use of our hybrid TEE-HE approach permits us to use countermeasures to side-channel attacks on the implementation of the encryption and decryption cryptographic functions implemented on the TEE. Since those are the only operations that happen on the TEE, there is no need to apply specific countermeasures every time the AI model changes.

V. The SOTERIA Design

a) The overall design: Figure 1 shows the SOTERIA's architectural design. At the initialization phase, the data must be sent to the TEE and in particular to the so-called *Controller* unit, whose responsibility is to manage the processing flow. In order to perform a secure data uploading phase, the user establishes with the *Controller* a TEE-terminated TLS secure channel extended with remote attestation features, which allows to prove that the communication is actually occurring with a secure enclave. In case no HE keys were available for the particular user, the *Controller* generates new HE keys and stores them using the TEE sealed hardware key so that their access can happen from inside the authorized TEE only. Not even the user knows the HE keys. The data is encrypted with the HE scheme that must be used for the first execution phase (either BGV/BFV for a linear phase or TFHE for a non-linear phase), and the processing finally started. During the computation, the *Linear/Non-Linear Processor* performs its job and notifies to the *Controller* once done. For the computation, the linear processor uses a BGV or BFV cryptosystem in order to take advantage of their SIMD capability. On the other hand, the non-linear computation makes use of the TFHE scheme and its boostrapping technique for a fast evaluation.
Importantly, the *Controller* takes note of the end of the computation of the processor and then receives two sets of ciphertexts from it:

- The actual ciphertext results of the computations but given in a random order chosen by the processor.
- Random ciphertexts generated by the processor and sprinkled among the good cipheretxts.

This is what we call the "flooding" operation. The TEE then switches the crypto-scheme (by decryption and re-encryption), and sends the data back out to the processor so that the subsequent execution phase can start. Such a flow is then iterated continuously until the final result is obtained.

b) Protection against side-channel attacks: It is important to notice that in this solution the TEE works in a one-shot fashion. It is started on-demand to only execute cryptographic operations. By doing so, we reduce the time window in which data would have been exposed to side-channel attacks if it was processed inside the TEE. On top of this, the operations executed inside the TEE will always be the same, whatever AI model is used by the server. Therefore we can protect the encryption/decryption operations of the HE cryptosystems from side-channel attacks using standard methods once, and never have to adapt those protections even when the model changes.

c) Protection against the TEE: The use of flooding (randomized order and dummy ciphertexts sent to the TEE for decryption) ensures that even an adversary that would have access to the TEE data (say through a backdoor introduced by the manufacturer) cannot actually glean user information or information about the model used by the server. If enough fake data is sent for decryption and re-encryption, we can ensure protection against a TEE backdoor at a low performance cost given that encryption and decryption are fast operations in lattice-based cryptography (the schemes we consider for this article are all lattice-based).

d) Optimized HE computation: We can optimize the homomorphic processing of algorithms containing both linear and non-linear functions, which perform better with different crypto-schemes. This would be either impossible or much costlier in terms of performance without the use of a TEE. On top of this, the multiplicative depth of the AI algorithm that we use can now be arbitrarily large, without affecting the size of the parameters.

VI. Related Work

In the literature, there are already research works which explore the combined use of HE and TEE. Sadat et *al.* [33] proposed a hybrid solution based on the Intel SGX technology and Partial HE (PHE) to preserve the privacy of genomic statistical analysis. The core idea consists in doing a subset of computations outside of the PHE flow, in the TEE, to gain in terms of performance. Wang et *al.* [34] described the adoption of SGX and HE with the main purpose of improving

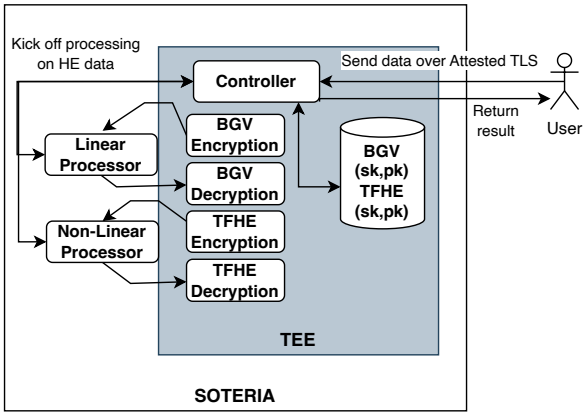

Fig. 1. The SOTERIA architecture

the performance of HE bootstrapping procedures via cloud computing resources. The user leverages HE on-premises and only requires the execution of the heavy bootstrapping procedure to the cloud. SGX is used to protect such execution since data would reside unencrypted in memory for a short time window.

Drucker et *al.* [35] implemented a voting system using the TEE of SGX and a partial HE scheme. The model uses a TEE (i.e., Intel SGX) to secure the code and data integrity, and the PHE scheme for encrypting the data. The goal is to protect against attackers trying to exploit the malleability of PHE by preserving the integrity with the TEE. At the same time, the PHE guarantees the data confidentiality independently of the trustworthiness of the TEE.

Coppolino et *al.* [36] proposed the concept of a VIrtual Secure Enclave (VISE), an approach that combines HE and TEE to overcome their limitations in a typical field-cloud deployment. In particular, with the proposed approach, the authors aimed at bypassing HE issues — specifically: the ciphertext expansion and unverifiable conditionals — and the limited memory size affecting the Intel SGX technology.

To the best of our knowledge, our solution is the first one proposing a TEE-protected switching mechanism of HE schemes, which ensures better performance and reduces the exposure of processing to possible side-channel attacks.

VII. Conclusion

In this paper, we presented SOTERIA, an architecture allowing to combine homomorphic encryption with the TEE in order to provide a secure and more efficient data processing. This privacy-preserving solution can be successfully applied for the inference step of AI models and, in particular, for the financial use case of the ENCRYPT project. The data processing will be performed in homomorphic domain and divided into a linear phase and a non-linear one. The main idea of our hybrid design makes use an honest-but-curious TEE as a way of ameliorating the performances of this HE computation by performing, between the linear and non-linear processing, the switch between different homomorphic schemes inside the TEE.

This is a conceptual design. We plan to implement and evaluate the performance of our secure TEE-HE solution for the evaluation of the AI model of the financial use case of ENCRYPT project, using WASI (WebAssembly System Interface). Another perspective is to extend the proposed framework with a remote attestation of the SGX enclave, similarly to the VISE approach [36]. Finally, we plan to investigate other manners of enhancing the security and the performances of privacy-preserving computation by mixing homomorphic encryption with TEE.

Acknowledgment

This project has received funding from the European Union's Horizon Europe Research and Innovation Programme under Grant Agreement No. 101070670 (ENCRYPT - A Scalable and Practical Privacy-preserving Framework).

References

[1] Z. Brakerski, C. Gentry, V. Vaikuntanathan, Fully homomorphic encryption without bootstrapping 27.

[2] Z. Brakerski, Fully homomorphic encryption without modulus switching from classical GapSVP, in: R. Safavi-Naini, R. Canetti (Eds.), Advances in Cryptology – CRYPTO 2012, Vol. 7417, Springer Berlin Heidelberg, pp. 868–886, series Title: Lecture Notes in Computer Science. doi:10.1007/978-3-642-32009-5_50.
URL http://link.springer.com/10.1007/978-3-642-32009-5_50

[3] J. Fan, F. Vercauteren, Somewhat practical fully homomorphic encryption 19.

[4] J. H. Cheon, A. Kim, M. Kim, Y. Song, Homomorphic encryption for arithmetic of approximate numbers, in: T. Takagi, T. Peyrin (Eds.), Advances in Cryptology – ASIACRYPT 2017, Springer International Publishing, pp. 409–437.

[5] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène, Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds, in: J. H. Cheon, T. Takagi (Eds.), Advances in Cryptology – ASIACRYPT 2016, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016, pp. 3–33.

[6] C. Boura, N. Gama, M. Georgieva, D. Jetchev, Chimera: Combining ring-lwe-based fully homomorphic encryption schemes, Cryptology ePrint Archive, Report 2018/758, https://eprint.iacr.org/2018/758 (2018).

[7] O. Chakraborty, M. Zuber, Efficient and accurate homomorphic comparisons, in: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, WAHC'22, Association for Computing Machinery, pp. 35–46. doi:10.1145/3560827.3563375.
URL https://doi.org/10.1145/3560827.3563375

[8] A. Grivet Sébert, R. Pinot, M. Zuber, C. Gouy-Pailler, R. Sirdey, SPEED: secure, PrivatE, and efficient deep learning 110 (4) 675–694. doi:10.1007/s10994-021-05970-3.
URL https://doi.org/10.1007/s10994-021-05970-3

[9] D. Maeda, K. Morimura, S. Narisada, K. Fukushima, T. Nishide, Efficient homomorphic evaluation of arbitrary uni/bivariate integer functions and their applications, report Number: 366.
URL https://eprint.iacr.org/2023/366

[10] C. Gentry, et al., Fully homomorphic encryption using ideal lattices., in: STOC, Vol. 9, 2009, pp. 169–178.

[11] Z. Brakerski, V. Vaikuntanathan, Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages., in: CRYPTO, Vol. 6841 of Lecture Notes in Computer Science, Springer, 2011, pp. 505–524.

[12] Z. Brakerski, C. Gentry, V. Vaikuntanathan, (Leveled) Fully Homomorphic Encryption Without Bootstrapping, in: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, 2012, pp. 309–325.

[13] J. Fan, F. Vercauteren, Somewhat practical fully homomorphic encryption., IACR Cryptology ePrint Archive 2012 (2012) 144.

[14] M. Van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, Fully homomorphic encryption over the integers, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2010, pp. 24–43.

[15] A. López-Alt, E. Tromer, V. Vaikuntanathan, On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption, in: Proceedings of the forty-fourth annual ACM symposium on Theory of computing, ACM, 2012, pp. 1219–1234.

[16] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène, Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds, in: Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22, Springer, 2016, pp. 3–33.

[17] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, J. Wernsing, Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy, Tech. Rep. MSR-TR-2016-3 (February 2016).
URL https://www.microsoft.com/en-us/research/publication/cryptonets-applying-neural-networks-to-encrypted-data-with-high-throughput-and-accuracy/

[18] F. Bourse, M. Minelli, M. Minihold, P. Paillier, Fast homomorphic evaluation of deep discretized neural networks, in: Proceedings of CRYPTO 2018, Springer, 2018.

[19] F. Boemer, Y. Lao, C. Wierzynski, ngraph-he: A graph compiler for deep learning on homomorphically encrypted data, CoRR (2018).

[20] A. Brutzkus, O. Oren Elisha, R. Gilad-Bachrach, Low latency privacy preserving inference, in: Proceedings of the 36th International Conference on MachineLearning, Long Beach, California, PMLR 97, 2019.

[21] A. Sanyal, M. Kusner, A. Gascón, V. Kanade, Tapas: Tricks to accelerate (encrypted) prediction as a service, in: ICML, 2018.

[22] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, L. Fei-Fei, Faster cryptonets: Leveraging sparsity for real-world encrypted inference, CoRR (2018).

[23] P. Maene, J. Gotzfried, R. de Clercq, T. Muller, F. Freiling, I. Verbauwhede, Hardware-based trusted computing architectures for isolation and attestation, IEEE Transactions on Computers PP (99) (2017) 1–1. doi:10.1109/TC.2017.2647955.

[24] L. Coppolino, S. D'Antonio, G. Mazzeo, L. Romano, A comprehensive survey of hardware-assisted security: From the edge to the cloud, Internet of Things 6 (2019) 100055. doi:https://doi.org/10.1016/j.iot.2019.100055.
URL http://www.sciencedirect.com/science/article/pii/S2542660519300101

[25] V. Costan, S. Devadas, Intel sgx explained, Cryptology ePrint Archive, Report 2016/086, http://eprint.iacr.org/2016/086 (2016).

[26] P.-E. Clet, M. Zuber, A. Boudguiga, R. Sirdey, C. Gouy-Pailler, Putting up the swiss army knife of homomorphic calculations by means of TFHE functional bootstrapping, report Number: 149.
URL https://eprint.iacr.org/2022/149

[27] D. Boneh, A. Sahai, B. Waters, Functional encryption: Definitions and challenges, in: Y. Ishai (Ed.), Theory of Cryptography, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 253–273.

[28] M. Alloghani, M. M. Alani, D. Al-Jumeily, T. Baker, J. Mustafina, A. Hussain, A. J. Aljaaf, A systematic review on the status and progress of homomorphic encryption technologies, Journal of Information Security and Applications 48 (2019) 102362. doi:https://doi.org/10.1016/j.jisa.2019.102362.
URL https://www.sciencedirect.com/science/article/pii/S2214212618306057

[29] M. Naehrig, K. Lauter, V. Vaikuntanathan, Can homomorphic encryption be practical?, in: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, CCSW '11, ACM, New York, NY, USA, 2011, pp. 113–124. doi:10.1145/2046660.2046682.

[30] D. Chialva, A. Dooms, Conditionals in homomorphic encryption and machine learning applications, CoRR abs/1810.12380 (2018). arXiv:1810.12380.

[31] Y. Xu, W. Cui, M. Peinado, Controlled-Channel attacks: Deterministic side channels for untrusted operating systems, in: S&P (Oakland), 2015.

[32] J. V. Bulck, F. Piessens, R. Strackx, Sgx-step: A practical attack framework for precise enclave execution control, in: Proceedings of the 2nd Workshop on System Software for Trusted Execution, SysTEX'17, 2017. doi:10.1145/3152701.3152706.

[33] M. N. Sadat, M. M. A. Aziz, N. Mohammed, F. Chen, S. Wang, X. Jiang, SAFETY: secure gwas in federated environment through a hybrid solution with intel SGX and homomorphic encryption, CoRR abs/1703.02577 (2017).

[34] W. Wang, Y. Jiang, Q. Shen, W. Huang, H. Chen, S. Wang, X. Wang, H. Tang, K. Chen, K. Lauter, D. Lin, Toward scalable fully homomorphic encryption through light trusted computing assistance (2019). doi:10.48550/ARXIV.1905.07766.
URL https://arxiv.org/abs/1905.07766

[35] N. Drucker, S. Gueron, Achieving trustworthy homomorphic encryption by combining it with a trusted execution environment, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications 9 (2018) 86–. doi:10.22667/JOWUA.2018.03.31.086.

[36] L. Coppolino, S. D'Antonio, V. Formicola, G. Mazzeo, L. Romano, Vise: Combining intel sgx and homomorphic encryption for cloud industrial control systems, IEEE Transactions on Computers 70 (5) (2021) 711–724. doi:10.1109/TC.2020.2995638.