SoK: Opportunities for Accelerating Multi-Party Computation via Trusted Hardware

Tong Liu[1], Zhen Huang[2], Jiaao Li[3], Jianyu Niu[1], Guoxing Chen[2], Yinqian Zhang[1]
[1]Southern University of Science and Technology [2]Shanghai Jiao Tong University [3]Tsinghua University
[1]{12332466@mail.sustech.edu.cn, niujy@sustech.edu.cn, yinqianz@acm.org}
[2]{xmhuangzhen, guoxingchen}@sjtu.edu.cn [3]lja23@mails.tsinghua.edu.cn

Abstract—**Multi-Party Computation (MPC) allows a set of parties to evaluate a joint function without revealing their private inputs to each other. However, existing MPC protocols suffer from high computation and communication complexity, making them incapable of practical privacy-preserving applications. Unlike MPC, which relies on expensive cryptography, Trusted Execution Environments (TEEs) aim to efficiently protect the integrity and confidentiality of run-time code and data from the hardware level. Despite this, TEE is not a panacea due to its vulnerability to various attacks to deteriorate their confidentiality. Therefore, integrating MPC protocols with TEE emerges as a promising approach to achieving both high performance and security. In this paper, we surveyed existing works of integrating TEE into MPC to build a privacy-preserving framework with optimized communication and computing overhead. These integrated MPC-TEE protocols vary from different trust assumptions of TEEs' confidentiality, by which they can be categorized into four classes: TEE with complete confidentiality, TEE with access pattern leakage, TEE with no confidentiality or integrity, and hybrid models of the above three cases. By analyzing these works, we summarized the focal points of integrating two approaches under different trust levels and provided paradigm cases for each level. We also figured out several challenges in integrating TEE and MPC protocols and suggested several future research directions that are worthy of further study.**

I. Introduction

Nowadays, the data volume involved in analyses and computations has grown exponentially due to the pursuit of more accurate and effective training and inference in Machine Learning (ML) [1]. This forces individual organizations with limited data to turn to a collaborative computing approach to jointly utilize their data. The traditional collaborative method requires a single party to collect all participants' data and then compute, which may lead to the leakage of valuable or sensitive data.

Multi-Party Computation (MPC) is a family of protocols that allow a group of mutually distrusted parties to compute a joint function on their input data without knowing the input data of each other. MPC has recently experienced rapid development, numerous optimized general-purpose MPC protocols have been proposed [2]–[5], and a continuous surge of customized protocols for ML [6]–[9], has emerged. MPC protocols are composed of several cryptographic techniques including Secret Sharing [10], Garbled Circuits [11], Oblivious Transfer [12], *etc*. Informally speaking, to protect sensitive data from unauthorized access, these cryptographic algorithms encrypt private data, which complicates the computational procedures and potentially necessitates additional rounds of communication. Although the performance of MPC protocols has been significantly improved over the past decade [13]–[15], MPC still suffers from high computation and communication complexity, making them incapable of either general-purpose applications or AI-specific applications.

To alleviate the high computational costs, TEE [16]–[18] offers an alternative way for conducting private computations among multiple data sources at the hardware level. TEE provides run-time isolation to protect the confidentiality and integrity of the code and data used by programs running within the TEE. This substantially reduces complexity during the computation process compared with MPC. Programs running inside a TEE can serve as a trusted party, providing secure computing services. While TEE is instrumental, it is not a panacea. It remains vulnerable to attacks that can compromise either its confidentiality or integrity guarantees. For example, one well-known issue is the access pattern leakage, which significantly impacts the confidentiality of TEE.

With MPC and TEE having their respective pros and cons, a natural question is raised: ***how to combine MPC with TEE, attaining the best of these two worlds, to realize both efficient and secure privacy-preserving computation?*** Driven by this question, many researchers leverage TEE's integrity and confidentiality on program executions to construct more practical and general MPC protocols [19], [20]. Generally, TEE aims to ensure the integrity of the running program as well as the confidentiality of corresponding data. The deployment of TEE can replace or simplify cryptographic algorithms in MPC to achieve the same level of security, accelerating the performance of MPC protocols. Furthermore, utilizing some of the functionalities in MPC protocols that are responsible for confidentiality can in turn counteract potential access pattern leakage when deploying TEE.

In this paper, we survey existing works of integrating TEE with MPC protocols to build more efficient privacy-preserving computation frameworks. Through an investigation into diverse studies with varying security assumptions, we

Tong Liu, Jianyu Niu, and Yinqian Zhang are affiliated with the Research Institute of Trustworthy Autonomous Systems and the Department of Computer Science and Engineering of Southern University of Science and Technology (SUSTech).

have categorized several situations that can notably accelerate MPC protocols. These scenarios generally presuppose basic integrity in TEE and propose designs tailored to different confidentiality assumptions. Furthermore, some works blend different security assumption schemes based on the degree of data confidentiality requirements.

Based on this point, the studies we surveyed are classified into four categories according to the trust level in TEE's confidentiality, they are: 1) trust TEE with complete confidentiality or integrity, 2) consider TEE with access pattern leakage, 3) assume TEE has no confidentiality and 4) hybrid model. Depending on the trust level, the focus of these works varies. For the first category, TEE is assumed to offer complete confidentiality, it then can be treated as a Trusted Third Party (TTP). The encrypted data can be decrypted and processed directly within TEE and then transmitted the result to all participating parties. When access pattern leakage is considered, MPC protocols need to focus on the design of access logic to conceal the traces of program executions within TEE and mitigate the impact of side-channel attacks. Even when treating TEE as an entity with no confidentiality or even no integrity, MPC protocols can still entrust some calculations involving trivial data to TEE for some effective preprocessing, or employ TEE to validate messages and results to enhance the threat model that can be handled by MPC protocol. The hybrid model could be the combination of the above conditions according to the specific requirements.

Integrating these two techniques can effectively reduce the substantial expenses related to MPC protocols, bringing them within practical reach. This allows for the implementation of data-intensive computations, *e.g.* ML, within a reasonable time cost. It also enhances some semi-honest protocols [6], [21]–[24] to be able to counter malicious attacks with significantly reduced costs, which was exceptionally challenging.

We focus on these four categories, summarizing the scenarios and the foremost issues that should be settled under each level based on the research we investigated. Our attention will be on the functionality of TEE in the framework and the enhancements it brings to MPC protocols. Based on the introduced works, we also analyze the shortcomings of existing methods and the challenges faced in further improving performance. We then extend research directions worth exploring in the future, considering these limitations and challenges.

Contributions. Our main contributions are as follows:

- We categorize existing works into four types by their distinct trust models: TEE with full confidentiality, TEE with access pattern leakage, TEE without confidentiality or integrity, and hybrid model.
- We conduct a comprehensive analysis of these works, including the resistible threat model, protocol purpose, enhancement on MPC, and the functionality of TEE, then summarize the general implementation paradigm and associated considerations.
- We summarize the existing shortcomings of existing works and analyze the challenges of addressing these shortcomings, providing a prospective view of potential directions for improvement in the future.

Roadmap. Sec. II briefly outlines the fundamental aspects of MPC and TEE, and Sec. III provides classification criteria for the current work. Sec. IV elaborates on the technical details of the related works, and Sec. V identifies the difficulties in the integration. Potential areas for further research are provided in Sec. VI. Finally, the paper is concluded in Sec. VII.

II. Background

In this section, we will introduce the background of this paper, which contains preliminaries of Multi-Party Computation (MPC) and Trusted Execution Environment (TEE).

A. Multi-Party Computation

MPC presents a secure computation approach to jointly computing a function $f(x_1, \cdots, x_n)$ among a set of n parties, denoted as $\langle P_1, \cdots, P_n \rangle$, using their private inputs $\langle x_1, \cdots, x_n \rangle$. Each party will eventually get the output of the function and learn nothing more than it.

Threat Model. For MPC protocols, there are usually two threat models: semi-honest adversary and malicious adversary.

- *Semi-honest Adversary.* Semi-honest adversaries are assumed to follow the protocol honestly but still be curious about others' private information.
- *Malicious Adversary.* Malicious adversaries may not follow the protocol faithfully and might deviate from their behaviors to gain more information.

Security Properties. MPC protocols usually aim to achieve the following three security properties.

- *Privacy.* No party should learn anything more than the protocol's officially revealed data.
- *Correctness.* Honest parties ultimately receive correct outputs from the function.
- *Availability.* Corrupted parties should not be able to prevent honest parties from receiving their output.

Here, publicly available information about other parties' input refers to the final output of the function and any pre-agreed portions. All information beyond this scope, including individual parties' inputs and any intermediate data not consented to be shared, must be kept private. Malicious parties should be unable to gain any useful information by any means. Each honest party is guaranteed that the output that it receives is correct. More concisely, the output of the MPC protocol with input x and function f should be identical to $f(x)$. In other words, the adversary should not be able to disrupt the computation by carrying out a disruptive behavior.

MPC Primitives. MPC typically comprises a set of cryptographic techniques, including but not limited to secret sharing [10], Oblivious Transfer (OT) [12], Garbled Circuit (GC) [11], and Beaver Triples [25].

1) Secret Sharing. In secret sharing [10], secret data are divided into multiple shares and then distributed to different participants for computation. The complete secret can only be reconstructed when a subset has more than the threshold number of members.

2) Oblivious Transfer. In Oblivious Transfer (OT) [12], the sender obliviously transfers certain private information to the receiver. The receiver gets nothing more than the transferred data, while the sender cannot learn which data is transferred. Specifically, the sender holds N indexed information $(M_1, M_2, ..., M_N)$, and a receiver chooses the information indexed by k $(1 \leq k \leq N)$ while not leaking k to the sender. This is also referred to as 1-out-of-N OT.

3) Garbled Circuit. Garbled Circuit (GC) [11] is one of the most popular approach [3] [4] [2] in MPC. GC supports two parties P_1 and P_2 jointly compute the function $f(x, y)$ without the trusted third party, where P_1 holds $x \in X$, and P_2 holds $y \in Y$.

4) Beaver Triples. Beaver Triples [25] use triplet a, b, c to help compute $x * y$ within the framework of secret sharing, without revealing complete x and y. The triplet a, b, c satisfies $ab = c$. Participants obtain a_i, b_i, c_i respectively, then compute $e_i = x_i - a_i$, $f_i = y_i - b_i$, and reconstruct e, f. This yields $xy = ef + af + be + c$.

MPC Overhead. The cryptographic techniques encompassed within MPC serve to safeguard the privacy of secret inputs during computations, albeit at the expense of increased computation and communication overhead. In some application scenarios, such elevated costs are often deemed impractical.

B. Trusted Execution Environment

TEE Platforms. Trusted Execution Environment (TEE) is a secure region designed to provide confidentiality and integrity for code and data within it. TEE is generally implemented on a CPU, shielding regions from potentially malicious privileged software, such as operating systems. Representative TEE platforms include Intel SGX [17], AMD SEV [16], and ARM TrustZone [18] [26].

Intel SGX is a representative process-level TEE. It divides applications into trusted and untrusted parts. The trusted parts are running on isolated memory, called enclaves. Enclave interfaces enable communication between external applications and enclaved functions. SGX also offers data sealing to encrypt and export secret data from enclaves, using encryption policies for access control.

On the other hand, AMD SEV is a popular approach to VM-level TEE. AMD SEV uses the encrypted virtual machine (VM) to protect applications while trusting the OS kernel. The AMD-SP manages the Virtual Encryption Key (VEK) for each VM, while the MEE encrypts and decrypts memory pages using the VEK.

Security Issues. Studies indicate that TEE is vulnerable can be categorized into physical attacks and software attacks.

1) Physical Attacks. The adversary tries to control or destroy the physical entities of the target such that the target system fails to provide service or the malicious user processes can have unrestricted access to the target system. The simplest model of physical attack is the Denial of Service (DoS) model, which disconnects the target system's network or the power supply. There are several common physical attacks such as port attacks [27], bus tapping attacks [28], chip attacks [28], and power analysis [29].

2) Software Attacks. Software attacks refer to using the malicious system or software inside it to drive the target system to an abnormal state. There exist many kinds of software attack techniques such as peripherals attacks [30], Rowhammer attacks [31], and cache timer attacks [32]. In peripheral attacks, adversaries can use malicious system software to compromise the target system and then perform unauthorized DMA. Rowhammer attack frequently refreshes the content of DRAM, causing bit flips in the memory cells that are responsible for managing memory translation or making secure decisions. Cache timer attacks can be mounted only on the user programs running at the lowest priority. The adversary will launch a user process to access the specific memory address, fill up the cache, then learn the deduce sensitive information.

TABLE I
COMPARISON BETWEEN MPC AND TEE

Technique	Rigorous Security Proof	Performance	Scalability	Flexibility
MPC	✔	*Low*	*Low*	*Low*
TEE	✘	*High*	*High*	*High*

C. Comparing MPC protocols and TEE

In this section, we provide a high-level comparison between MPC and TEE in terms of rigorous security proof, performance, scalability, and flexibility, shown in Table I.

Rigorous Security Proof. The MPC protocol is composed of cryptographic algorithms, and compared to TEE, it has more rigorous security proofs. As a result, it can offer better security and confidentiality than TEE. However, the implementation of cryptographic algorithms also comes with higher computational and communication overheads.

Performance. TEE, on the other hand, achieves a secure computational environment through hardware support. It no longer relies on high-overhead cryptographic algorithms for encryption, leading to highly efficient computations.

Scalability. MPC protocols can typically support few participants and can be challenging to scale for numerous participants due to the complexity of cryptographic protocols and the need for multiple rounds of communications. Since computations can be isolated within individual TEE, it is easier for TEE to scale up for applications involved with many parties.

Flexibility. Though MPC protocols are versatile and can handle a wide range of computations, the designs of MPC protocols depend on the specific applications or computations they are intended for. Conversely, TEE is tailored for running programs within isolation, offering a high level of control and customization for developers, which gives higher flexibility.

Summary. MPC offers more stringent security proofs to ensure the confidentiality of sensitive data, while TEE takes advantage of performance, scalability, and flexibility. Thus, combining these two approaches can build a more advanced framework, holding the potential to retain both the strong confidentiality of MPC and the efficiency of TEE simultaneously. This integration empowers the framework to handle more complex MPC scenarios without compromising security.

III. Taxonomy and Criteria

In this paper, we survey existing works that integrate TEE and MPC protocols for reducing cryptography overhead. Specifically, according to different trusted assumptions of TEE's confidentiality, we categorize them into four types: trust TEE with complete confidentiality, consider TEE has access pattern leakage, TEE without confidentiality, and hybrid model which combines the situations above.

- **TEE with Complete Confidentiality.** If all participants assume TEE offers complete confidentiality, TEE can be viewed as a Trusted Third Party (TTP) since it can provide both integrity and confidentiality. In this setting, the main issues include building a secure channel for communications between TEE and parties and providing robust attestation mechanisms, *etc*. Usually, this architecture achieves the best performance.
- **TEE with Access Pattern Leakage.** To mitigate or eliminate attacks that analyze runtime information and access patterns, MPC protocols deployed in TEEs can reduce the need for cryptographic primitives designed to protect confidentiality. This enables them to focus specifically on addressing access pattern leakage of TEEs, particularly in mitigating side-channel attacks. It achieves an adequate level of confidentiality with minimal overhead, leading us to consider this as an independent category.
- **TEE with No Confidentiality.** When TEE is considered as an entity without confidentiality, sensitive computation can no longer be entrusted to TEE. However, some preprocess calculations involved with trivial input data can still be employed in TEE. Alternatively, TEE's integrity keeps the program inside TEE to be uncontrollable by a malicious party, which also enables TEE to role as an attestation service. This configuration may not significantly enhance performance, but it can still optimize communication complexity and strengthen the protocol's threat models.
- **Hybrid Model.** Compared to considering TEE as a none confidentiality party and implementing MPC protocol in a pure cryptographic way, a hybrid model of cryptography and TEE can greatly improve performance. The data can be divided into several parts according to the degree of secrecy. According to various requirements, we can combine multiple models offering different levels of confidentiality to achieve the highest efficiency while meeting security needs. This method employs a variable threshold, allowing the expected trade-off between confidentiality and performance to be achieved in practice.

Our criteria primarily focus on the threat models that can be defended by these proposed frameworks, the use cases' purposes of MPC protocols within the framework, the functionality of TEE, and whether the deployment of TEE has improved the MPC protocols' performance or enhanced the threat models that the MPC protocols can handle. Since the enhancement of the threat model in traditional MPC protocols requires extremely high overhead, we consider this promotion as part of the acceleration of the MPC protocols.

IV. Use Case Analysis

This section analyzes recent works using TEE-based MPC schemes, as documented in Table II. These schemes are categorized into four types according to the trust levels of TEEs. Specifically, we first provided an overview of the current application scenarios within each category and summarized the issues that require attention. Following this, using their research as an example, we explored specific technical details, elaborating on how the problem set is instantiated in practice. We focused on discussing the problems addressed by these technologies, along with which part of the computation is optimized. Additionally, we explored how this design enhances the performance and threat model level of MPC protocols through the use of TEE.

A. Trust TEE with Complete Confidentiality

Problem Statement. By fully entrusting a TEE with confidentiality and integrity, it can effectively function as a trusted third party (TTP). With a secure communication channel, MPC can be implemented by simply gathering parties' inputs into TEE, performing the pre-agreed functions, and then returning the output to those participants. Employing this architectural approach can enable MPC schemes to support a larger number of participants with minimal performance degradation. These approaches [33] [21] [22] facilitate accelerated processing, as no communication is required for the function execution inside the TEE and all computations within the TEE operate on the declassified plaintext.

Case Analysis. Küçük *et al.* [33] present how trusted hardware, like Intel SGX, can establish a trustworthy remote entity (TRE) suitable for many-party applications in the malicious threat model. In this approach, TEE is assumed to possess complete confidentiality and integrity. External entities cannot observe or interfere with the internal program execution. TEE is only vulnerable to physical attacks. The essential prerequisites within this framework are secure communication channels and a root trust attestation service authenticated by all participants. These elements enable the TEE to work as a TTP to privately facilitate the necessary computations among multiple parties, without cryptographic mechanisms and multiple rounds of communications.

They focus on privacy-preserving energy metering and compare the performance of an SGX-based TRE to a Trusted

TABLE II
Use Cases Comparison. The notations GP, FaaS, ML, and PMT are short for Generate Purpose, Function-as-a-Service, Machine Learning, and Private Membership Test, respectively.

Framework	Trust Model	Threat Model	Purpose	Enhancement on MPC		Functionality of TEE
				Threat Model	Performance	
Küçük *et al.* [33]	TEE with Complete Confidentiality	Malicious	GP	✔	✔	TTP for execution
Bahmani *et al.* [21]		Malicious	GP	✔	✔	TTP for execution and attestation
Alder *et al.* [22]		Malicious	FaaS	✔	✔	TTP for execution
Ohrimenko *et al.* [34]	TEE with Access Pattern Leakage	Semi-honest	ML	✘	✔	Provide execution on oblivious I/O
Shaon *et al.* [35]		Semi-honest	ML	✘	✔	Provide execution on vectorized data
Felsen *et al.* [36]		Semi-honest	GP	✘	✔	Provide execution on Universal Circuit
Tamrakar *et al.* [37]		Semi-honest	PMT	✘	✔	Provide execution with fixed pattern
Riazi *et al.* [38]	TEE with No Confidentiality or Integrity	Semi-honest	ML	✘	✔	STP for offline phase preprocess
Lu *et al.* [39]		Semi-honest and Malicious	GP	✘	✔	STP for offline phase preprocess
Kumar *et al.* [6]		Malicious	ML	✔	✘	Provide message attestation
Jie *et al.* [40]		Malicious	ML	✔	✔	Provide execution on encrypted data
Wu *et al.* [23]	Hybrid Model	Malicious	GP	✔	✔	Provide execution based on trust level
Choi *et al.* [41]		Semi-honest	GP	✘	✔	STP for partial execution
Gupta *et al.* [24]		Semi-honest and Malicious	GP	✔	✔	STP for partial execution

Platform Module (TPM) based system. The results demonstrate that SGX-TRE offers comparable performance, requiring fewer lines of code and providing memory protection. Further improvements in the SGX remote attestation protocol can enhance the SGX-TRE's potential, and future work involves optimizing its design and exploring multiple enclaves and enclave structures for the TRE.

Bahmani *et al.* [21] explore the use of Intel SGX for general MPC. SGX's Isolated Execution Environments (IEE) provide secure and trustworthy execution environments, protecting against malicious parties and ensuring code and I/O integrity. This framework regards programs and code running within SGX as a TTP. Bahmani *et al.* also formally define a new concept, Labeled Attested Computation (LAC), which enhances security attestation for communication by attaching specific labels to inputs and outputs. The paper then provides detailed insights into constructions of MPC computations functionalities based on this LAC model. Their design involves loading the functionality into a TEE, generating keys for attestation, and labeling frameworks for verifying the information transmitted. This approach offers the capabilities to construct more efficient and scalable MPC protocols for applications in cloud computing, with minimal overhead on participants.

Alder *et al.* [22] present S-FaaS, the first implementation of

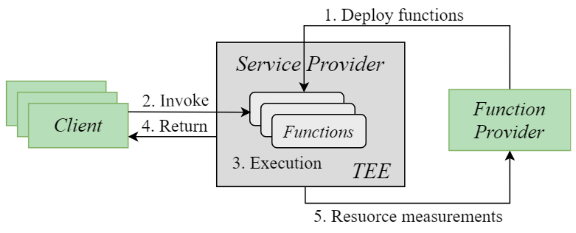

Fig. 1. S-FaaS Framework Workflow. The figure shows the workflow of the Secure FaaS framework [22], where the service provider was deployed in TEE, providing private function executions for clients and resource measurement service for function providers.

Function-as-a-Service (FaaS) with strong security and accountability guarantees backed by Intel SGX. As shown in Figure 1, this framework is composed of function providers, clients, and service providers. The service provider will first ask function providers to deploy functions on the service end and ensure privacy for both the function and clients' inputs/outputs by SGX. Additionally, SGX guarantees the integrity of function execution and performs correct computations. Consequently, clients and function providers can participate as distinct parties in collaborative privacy-preserving MPC, in which SGX serves as the TTP, eliminating the originally high expenses. The design introduces a new key distribution enclave and a novel transitive attestation protocol to match the dynamic event-driven nature of FaaS. S-FaaS includes resource measurement mechanisms to securely measure compute time and memory allocations. The results show that the resource measurement mechanisms have excellent performance and S-FaaS can also integrate with smart contracts for decentralized payment.

All the aforementioned works operate within the threat model of malicious adversaries. They employ the TEE nearly identically, treating it as a trusted third party. This facilitates the acceleration of MPC protocols by enabling specific private function calculations.

B. TEE with Access Pattern Leakage

Problem Statement. Though TEE functions as a secure and isolated region, it does not protect against side-channel attacks. A malicious host can exploit the correlation between the access pattern and the function's input by analyzing the access statistics, subsequently gaining extracted information or even private data embedded within the access patterns. In such scenarios, the MPC protocols are not required to offer comprehensive confidentiality. Instead, it can be designed specifically to address access pattern leakage, mitigating the need for majorities of cryptographic algorithms composed in MPC protocols that contribute to high overheads. The following works are analyzed in this category. [34] [35] [36] [37]

Case Analysis. Ohrimenko *et al.* [34] present a practical system for privacy-preserving multi-party machine learning by proposing data-oblivious machine learning algorithms. The computations in TEE are assured to be confidential even with only processor chips is impenetrable. This framework is constructed by designed oblivious primitives, customized to specific operations. These oblivious primitives reform the algorithms' access pattern to make the interaction trace indistinguishable from that produced by an algorithm simulator that executes identical operations on the public parameters. Algorithms written with oblivious primitives ensure the access pattern of TEE with arbitrary input to be indistinguishable from the host outside TEE and show strong and provable guarantees that the adversary could learn nothing about sensitive data by analyzing statistics of access patterns. Relying on a library of general-purpose oblivious primitives, the framework mitigates side-channel attacks. Hence, some functions of MPC protocols can be delegated to TEE utilizing these primitives, while other resource-intensive primitives within the MPC protocol are eliminated.

Shaon *et al.* [35] propose a generic data analytics framework that leverages TEE in the multi-party cloud computing environment. The framework presents a high-level Python-inspired language that handles data in a vectorized way to hide the detailed information access during the computation. This language allows efficient, generic, and oblivious execution of data analytics tasks. Moreover, they introduce BigMatrix, an abstraction for handling large matrix operations in a data-oblivious manner that supports vectorization. With BigMatrix, managers can handle encrypted matrix data storage transparently, clients are allowed to access data in a block-by-block manner with integrity and privacy protection, such a vectorized computation automatically hides important sensitive information. By integrating these two techniques, this framework achieves comprehensive oblivious data access to counteract access pattern leakage. Similarly, vectorized patterns offer numerous optimizations over traditional MPC. More importantly, due to native support for vector and matrix operations, they are particularly favorable for convolution computations in ML making it serves as an ideal paradigm for deploying TEE with side-channel risks into MPC for the ML area.

Felsen *et al.* [36] present a secure and private function evaluation framework with SGX. Secure function evaluation (SFE) functioned as an MPC protocol, while Private Function Evaluation (PFE) considers the evaluating function as the private input of one of the participants. The approach to weaken the side-channel attacks in this framework is to use the boolean circuit for the function evaluation. In a boolean circuit, all possible paths of the function are executed, and memory accesses are independent of the inputs of any gate, the adversary can hardly figure out the inputs of an exact gate by gathering information while paging. Besides, Felsen *et al.* indicate that since the evaluation of the boolean gate takes constant-time cost, the timing attack does not work as well. Thus, this architect can efficiently defend the side-channel attack without causing cryptographic overheads, this protocol is also suitable for PFE. However, when the evaluating function f is complicated, the cost of sending f in an encoded way could increase by several orders of magnitude. Therefore, they leverage a Universal Circuit (UC) to simulate a boolean function. In this paradigm, only the function-type input should

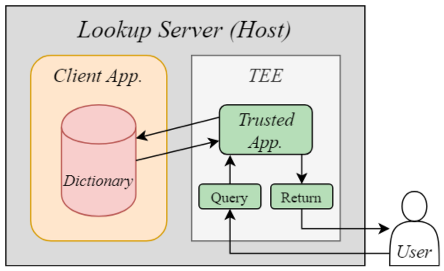

Fig. 2. Cloud-based Private Membership Test (PMT). The cloud-based PMT [37] provides a lookup service with a dictionary within a client application, user can perform a PMT by querying the trusted application deployed in the TEE of the server.

be transmitted, which then can be simplified to 4 programming bits. This framework applies to most general computations and can significantly reduce the majority of encryption overhead and communication overhead.

Tamrakar *et al*. [37] present a scalable Private Membership Test (PMT) framework using trusted hardware TEE shown in Figure 2. This framework highlights the drawbacks of the prevalent PMT algorithm Private Set Intersection (PSI), such as high overhead and low scalability. Tamrakar *et al*. then propose a paradigm entrusting the execution of the PMT algorithm to a TEE. While the emphasis of the framework is not primarily on addressing side-channel attacks, it does provide insights into this aspect of the threat model. The framework employs conventional techniques to mitigate such attacks by cutting off dependencies between input and access patterns, including establishing fixed access patterns, performing constant-time processing, ensuring that every query remains exactly one full carousel cycle, and the utilization of Oblivious RAM [42]. However, these methods can lead to the wastage of resources. For instance: when the input is small, the same I/O process and runtime costs are incurred, which is not flexible and can impact efficiency.

C. TEE without Confidentiality

Problem Statement. Due to the presence of side-channel attacks, private data inside TEE still can be potentially observed by the adversary host. Although we have discussed how to mitigate or eliminate the damage caused by side-channel attacks by modifying the architecture of the MPC, the accompanying withdrawal is that architectural design needs more complicated requirements for assuring confidentiality in TEE. However, even assuming TEE provides no guarantee of confidentiality, it can still take over certain functions to achieve optimization such as reducing communication costs and computational overhead or enhancing security functionality. We mainly outline the following works. [38] [39] [6] [40]

Case Analysis. The framework Chameleon [38], proposed by Riazi *et al*. adopt the TEE-accelerated MPC model. Chameleon is the first research suggesting utilizing TEE as a Semi-honest Third Party (STP) to optimize MPC In Chameleon. In the setting of STP, TEE is not required to provide confidentiality. The STP only participates in the offline phase, generating correlated randomness for subsequent online stages. This reduces the interaction rounds in the MPC protocol. Since the STP is not involved in the online phase, it has no access to either parties' private inputs or the executed program. Moreover, Chameleon optimizes the Du-Atallah [43] protocol based on STP. Leveraging the integrity of trusted hardware, the STP optimally designs the utilization of secret sharing to perform better vector dot products. This design further improves the efficiency of matrix multiplication, which subsequently enhances the computational performance of this framework.

Similar to Chameleon, Lu *et al*. [39] introduce the *semi-trusted hardware model* to illustrate that potentially malicious hardware can still be used to accelerate MPC executions. Lu *et al*. formally define the security models for TEE in scenarios without confidentiality and even without integrity in this *semi-trusted hardware model*. In the setting of *semi-trusted hardware model*, trusted hardware functionality could be corrupted by the adversary, who can obtain any inputs from parties or even arbitrarily modify the function's execution in TEE. This paper mainly introduced the design of Random OT (ROT) and Garbled Circuits (GC) in a semi-trusted hardware model. For random OT generation, with TEE deployed on the receiver, the sender only needs to send a random seed, ROT copies then can be computed locally in polynomial time. The process of GC is quite similar to ROT, TEE is deployed on the GC's evaluator to generate the GC tables and locally return the results. Therefore, the communication overhead is reduced to linearly related to the input size and is independent of the circuit size. Lu *et al*. further utilize this concept in [44], addressing some associated drawbacks and progress more enhancing performance.

Kumar *et al*. [6] present Cryptflow with a component named Aramis that can elevate the semi-honest MPC protocol to one that can securely withstand malicious adversaries. Different from the designs above, TEE is no longer used to simply accelerate MPC protocols but rather to efficiently enhance the threat model that the protocol can handle, where such enhancements were deemed to incur substantial costs. TEE with its integrity, can verify computation results or transmitted messages, addressing the challenging problem present in MPC protocols. Aramis does not require confidentiality, secure verification on communication messages is guaranteed by the integrity of TEE, boosting the semi-honest MPC protocol to securely defend against malicious adversaries. In this module, signing keys and verification keys are generated for each TEE instance, every message sent between parties within a semi-honest protocol needs to be calculated by the TEE attestation function with the key pair. Therefore, TEE is used to validate these messages, also ensuring the correctness of the information transmitted between parties. This approach allows the protocol to effectively counter malicious adversaries, enhancing the security assumption of MPC.

Jie *et al*. [40] introduce a Graph Neural Network (GNN) training and inference framework designed to facilitate the outsourcing of users' data to servers for computational purposes. Their approach considers a scenario where the servers are potentially malicious, capable of employing side-channel attacks when utilizing SGX enclaves, attempting to infer user data during model training, and even purposely deviating from the protocol then generating poison results. To address these security concerns, the authors develop a framework that leverages SGX to ensure code integrity through the attestation feature present in TEE. They also adopt a 2-out-of-2 secret sharing scheme to divide the data and distribute them to different servers, to safeguard data privacy. Hence, TEE's confidentiality is guaranteed through secret sharing, eliminating the need to consider side-channel attacks. The correctness of the MPC protocol deployed within TEE is ensured by TEE's integrity. Consequently, this model allows the system to handle malicious scenarios only relying on secret sharing, providing both security and high performance.

D. Hybrid Trust Model

Problem Statement. Based on the discussions in the previous three subsections, we have defined trust models for three levels of trust. Moreover, some works consider the trust of the TEE to encompass two or all three levels of trust. Functions often consist of multiple parts, each with varying privacy requirements for inputs. Thus, breaking functions into pieces and deploying integrating methods at different trust levels based on demands can provide a targeted approach to address the main issues under this requirement. This allows each part of the function to achieve an optimal solution locally. In this subsection, we present the scenario of hybrid levels of trust and introduce these representative works. [23] [41] [24]

Case Analysis. Wu *et al.* [23] explore the hybrid trust model in a specific SQL scenario when different parties have different trust levels in TEE. They propose a generic framework called HYBRTC to realize hybrid trust MPC. HYBRTC considers three levels of trust among different parties: complete trust, partial trust, and distrust. They first introduce a new notion of multifaceted trust hardware as a formalization of TEE-like hardware and provide rigorous security analysis in the Universal Composability(UC) model. Then the framework is instantiated in a scenario of a privacy-preserving distributed query on multiple databases by presenting some typical secure SQL operations.

HYBRTC is a framework of hybrid trust computing which is composed of a client $\mathcal{C}$ and multiple TEE-enabled servers $\mathcal{S}$ as shown in Figure 3. $\mathcal{C}$ may have complete trust or partial trust on a server $\mathcal{S}_i$, whereas $\mathcal{S}_i$ distrust each other. HYBRTC mainly has three steps with two functions f and g and parameters between $\mathcal{C}$ and $\mathcal{S}$. Specifically, if $\mathcal{C}$ completely trusts $\mathcal{S}_i$, parameters will be encrypted by an agreed session key between $\mathcal{C}$ and $\mathcal{S}_i$. If $\mathcal{C}$ partially trusts $\mathcal{S}_i$, the function f will be split into two parts and be processed locally in $\mathcal{C}$ and $\mathcal{S}_i$ respectively. Then a 2PC will be conducted to get the final result of f. The function g should be performed as the standard

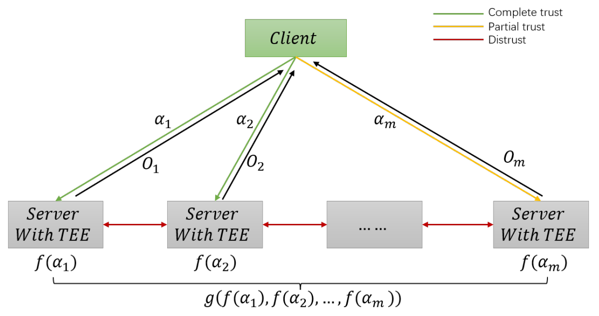

Fig. 3. Hybrid Trust Computing Framework [23]. The framework is composed of a group of TEE servers with no mutual trust. Clients can entrust their functions to different TEE servers depending on their trust level to this server.

cryptographic MPC protocols and generate partial output O_i, which can then be reconstructed to output.

Choi *et al.* [41] present a balanced approach to two-party secure function evaluation (2P-SFE) on SGX-enabled processors. They design a protocol that partitions the function f into pieces and allows the protocol designer to choose which components are evaluated within the enclave and which ones utilize cryptographic techniques. This facilitates participants to separate functions based on data sensitivity. The portions requiring strong security assurance are computed using the origin steps of MPC, while the remaining parts are delegated to TEE. This architecture presents a promising trial on the pursuit of the trade-off between security and performance. Choi *et al.* formalize the notion of 2P-SFE for this setting and proved that their protocol meets this requirement. This design offers excellent insights for resolving hybrid trust models and provides effective methods for function partitioning. It grants developers exceptional flexibility and programmability.

Gupta *et al.* [24] conduct an implementation of the two-party secure function evaluation (2P-SFE) on SGX-enabled devices, thereby enhancing its practicality for various applications. This design introduces the SGX-supported 2P-SFE protocol, tailored to enable SFE to withstand side-channel attacks specific to SGX. Subsequently, it focuses on the hybrid utilization of the original 2P-SFE protocol and SGX-supported 2P-SFE. Gupta *et al.* establish a choice of the protocol used, by partitioning the secrecy levels among parties' inputs. They provided a detailed workflow for the data structures in 2P-SFE, *e.g.* in the form of GC, encrypted data are interconverted with the structures that are capable of SGX-supported 2P-SFE in different trust levels, achieving the desired hybrid model effect. They developed two protocols for secure computation in the semi-honest model specifically tailored for this platform for simplicity. However, Gupta argued that the enhancement of the threat model for this protocol can be much easier than traditional MPC protocols. Simply running the function N times can ensure the computation is not interfered, where N refers to the threshold of malicious nodes.

V. Challenges of Use Cases

Based on the use cases outlined in Section IV, we have identified several challenges in the design of TEE-based MPC schemes. These challenges encompass security assumptions and centralized root of trust in TEE, as well as the detection and revocation of compromised TEE.

A. Security Assumptions of TEE Implementations

One challenge of using TEE for MPC is to figure out how secure the existing TEE implementations are or make a practical assumption on TEE. As discussed in II-B, current TEE implementations face varieties of software and physical attacks [45]–[47]. Implementations of TEE are constantly evolving as more attacks are discovered. As Intel makes it clear that SGX does not defend against side-channel attacks, a series of side-channel attacks based on cache access patterns have been proposed. In the context of using TEE for MPC, researchers assume TEE security under the condition that access-pattern leakage can be addressed, and they propose data-oblivious or constant-time algorithms [22], [34], [35], [37]. This assumption can be broken by recently disclosed power consumption-based side-channel attacks against constant-time algorithms [48]. Consequently, researchers are exploring new TEE-based MPC designs that do not assume the confidentiality of TEE states [6], [38], [39].

While the aforementioned side-channel attacks target specific secrets within the enclave, speculative execution attacks enable the adversary to extract arbitrary secrets inside the enclave, even the remote attestation key. With the leaked remote attestation key, adversaries can impersonate any enclave and deceive the client, thereby undermining the integrity guarantee provided by TEE. The severity of such attacks, which completely compromise TEE's security guarantees, leads researchers to assume that hardware patches released by vendors can address these issues. Disregarding these threats as out of scope when developing TEE-based MPC schemes would question the practicality of the designed schemes. Therefore, recent works have acknowledged this concern and proposed hybrid trust models in which some TEE platforms could potentially be entirely compromised [23], [24], [41].

B. Centralized Root of Trust

After making proper assumptions about the security of TEE implementations, another challenge is to deal with the centralized root of trust design. TEE usually relies on remote attestation to establish trust with the client as described in II-B. The roots of trust in existing remote attestation schemes are usually root secrets within the processors. These secrets are usually embedded into the processors during manufacture. The manufacturer holds copies of these secrets or the corresponding certificates of these secrets, which are needed when verifying remote attestation requests. The manufacturer might choose to run its remote attestation service such as Intel Attestation Service [49], provide certificates to third parties (such as data centers) to set up third-party attestation services (*e.g.* Intel DCAP), or release certificates to the public to allow enclave users to attest enclaves by themselves. All these settings require enclave users to trust the manufacturer to manage the root secrets properly. However, such a centralized root of trust design might incur risks of a single point of failure. Such concern could even be exacerbated when the manufacturer might be compelled to disclose the secrets.

The centralized root of trust might also restrict the scalability of the MPC designs. Specifically, while platforms possessed by different entities could be viewed as different parties in MPC, whether enclaves running on different platforms can be considered as different parties are debatable since these enclaves might be endorsed by the same manufacturer, *e.g.* when SGX enclaves running on different platforms are using the same Intel Attestation Service to attest themselves to other parties. On the other side, involving more manufacturers would need to deal with the trustworthiness of different manufacturers which depends on various factors such as the reputation, the scale, or even the registration place. Raising the bar of trusted manufacturers would lead to a limited number of TEE participants (*e.g.* , Intel and AMD), while a lower bar requires to consider potentially compromised TEE implementations.

C. Detection and Revocation of Compromised TEE

Due to various attacks on TEE, some TEE implementations or platforms could be compromised, despite what security assumptions are made during design. Hence, it is critical to (1) timely detect and (2) properly revoke compromised TEE.

Detecting compromised TEE could be quite challenging. For example, a TEE platform is considered compromised when its root secret (used as the root of trust) is leaked. An adversary who obtains the root secret has no incentive to disclose this compromise. Instead, it would prefer to conceal it and run arbitrary code that could bypass remote attestation with the leaked root secret.

Some existing remote attestation schemes, such as Intel Enhanced Privacy ID (EPID) [49], do provide mechanisms to verify whether an attestation report is from a compromised TEE, but they require the compromised TEE to be disclosed to the attestation service, and it is not clear which party is authorized to disclose compromised TEE. If an adversary gains the authentication to disclose TEE, it can report to the attestation service that some honest TEE platforms as compromised ones to exclude them from participating in the MPC protocols.

VI. Opportunities and Future Work

In this section, we discussed the promising research directions of combining TEE and MPC protocols, which are discovered and summarized from the cases mentioned above. Specifically, these cases show a growing trend of using TEE in MPC either to promote the system performance [39] or to improve the security level [6]. The challenges, security assumptions as well as limitations in these studies bring new insights into the future directions of TEE-aided MPC designs and implementations.

A. New Security Primitives

One security primitive for TEE that should be enhanced is to guarantee the state continuity property. The state continuity mandates that when a protected module resumes execution from an interruption (e.g., reboots, power outages, or system crashes), it should resume the same state before the interruption [50]. Rollback attacks pose a serious threat to the integrity and security of applications running within the TEE that can replay old transactions or cryptographic operations, leading to financial losses, data breaches, or unauthorized access [51]–[56]. Also, when TEE is recovering from faults, the state storage service can propose a rollback attack by transmitting the outdated state of the entity, which subsequently impacts state continuity. State continuity violation has severe consequences in many applications, such as payments [57], trusted storage [58], smart contract [59], as well as authentication rate limiting [60], [61]. In many applications, such as auctions and payment systems, the system has to guarantee state continuity.

Since TEE can crash due to physical attacks such as powering off the machine, it is imperative to design a more sophisticated recovery mechanism to achieve efficient recovery of the previous state from a TEE crash, along with preventing state rollback results from delayed state recording. There have been numerous studies [52] [53] [55] [62] [63] addressing rollback attacks and preserving state continuity for trusted hardware when fault recovery occurs. Nevertheless, within the framework of combining TEE and MPC, it has the potential to construct a more secure rollback-resistant state recovery by utilizing the design of the MPC protocols.

B. Standard Security Assumption

There are various security assumptions including treating TEE as a trusted third party, considering TEE with access pattern leakage, and TEE without confidentiality or integrity. As we can see, each system designer has its security assumption of TEE, and there is no agreement on the security models in the community. There are two main reasons for the divergence. First, system designers have different views of the root trust. In particular, they have different confidence in no backdoor, steganography, and kleptography from the manufacturers and centralized attestation mechanism for code integrity. Second, system designers think differently about attacks such as side-channel attacks. New attacks or defenses are proposed continuously, which determines the confidentiality guarantee of TEE. Thus, up to now, there hasn't been a unified security assumption, leading to a bewildering array of security assumptions in various research studies.

The divergence in trust models leads to various designs. While this can promote new studies, it also distracts researchers' efforts. One future work is to propose a standard security model. As this integration framework becomes more mature, there is a need for a unified and practical security standard and evaluation system to simulate real-world use cases, for a more precise assessment of the framework's performance, security, flexibility, *etc*. We argue that it is hard to use one security assumption for all MPC scenarios, by practical scenarios and awareness of multiple attacks, establishing a widely recognized evaluation criteria or exemplary choice will be a crucial direction for standardizing the entire research framework in the future.

C. Decentralized TEE Identity Management and Attestation

As discussed in Sec. V-B, one of the weaknesses is the centralized root of trust design in TEE. Due to the trust requirements associated with this centralized attestation service, the system will lack fault tolerance and robustness. To address this, distributed solutions are typically adopted. Subsequently, one promising research direction for enhancing this MPC-TEE framework is to provide decentralized TEE identity management and attestation against a single point of failure. Additionally, the decentralized attestation model can address the issues when participants lack trust in the trust root, providing stronger security assumptions for the management and attestation of TEE entities. Moreover, due to the increasing number of TEE providers, future architectures need to consider that each participant may not acquire TEE from the same provider, *i.e.*, , heterogeneous TEE. The authentication of heterogeneous TEE is challenging and cannot rely on a single root-of-trust mechanism. This urgency drives the need for a distributed service that supports the management and authentication of heterogeneous TEE nodes. For example, ARM research's Veracruz project [64] aims to support cloud-based IoT applications with heterogeneous TEE. The project, under the Confidential Computing Consortium [65], demonstrates some of the hurdles that one has to overcome to support an infrastructure of heterogeneous TEE, which is an excellent paradigm for this area.

D. Other Privacy-Enhancing Technologies

There are potentially analogous TEE-based methodologies that can enhance the efficiency and security of Privacy-enhancing technologies (PETs) – Homomorphic Encryption (HE), Zero-Knowledge Proofs (ZKPs), and Functional Encryption (FE). Currently, some works leverage TEE to assist PETs like ZKPs [66], [67], HE [68], and FE [69]. These TEE-based approaches offer promising avenues for optimizing the performance and security of these cryptographic techniques. By leveraging the capabilities of TEE, it becomes possible to mitigate certain computational and security challenges associated with HE, ZKPs, and FE.

E. End-Game Scenario

Due to its design with strict confidentiality and integrity, MPC incurs unavoidable computational and communication costs. With the continuous development, if TEE can obtain stronger guarantees of confidentiality and integrity, it may completely replace the existence of MPC. Even if TEE cannot immediately counter all attacks, MPC inspires privacy computing protocols tailored for TEE. Leveraging the characteristics of TEE, privacy computing can focus only on the aspects that TEE cannot protect, *e.g.* providing integrity against side-channel attacks or offering correctness check. This will reduce significant computational and communication costs, greatly improving the performance achievable by privacy computing.

VII. Conclusion

In this paper, We conducted a comprehensive analysis of existing works that integrate TEEs with MPC. We categorize existing works into four types by their distinct trust models: TEE with full confidentiality, TEE with access pattern leakage, TEE without confidentiality or integrity, and hybrid model. We summarize the existing shortcomings of existing works and analyze the challenges of addressing these shortcomings, providing a prospective view of potential directions for improvement in the future. We also show that there are still many possibilities for solving existing weaknesses *e.g.* side-channel attacks and applying TEE to MPC protocol to gain considerable performance with drawbacks, such as hybrid model, remote attestation, or innovation of encryption methods. These new exploration directions may also help MPC to be applicable in more computing fields.

Acknowledgement

We would like to thank our shepherd, Gang Tan, and the anonymous reviewers for their helpful comments and feedback. We also thank Hanzheng Lyu for the fruitful discussions. This work is supported in part by the Shenzhen Science and Technology Program under Grants RCBS20221008093248075 and JSGG20220831095603007; and in part by the NSFC under Grant 62302204.

References

[1] A. L'Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine learning with big data: Challenges and approaches," *IEEE Access*, pp. 7776–7797, 2017.

[2] D. Demmler, T. Schneider, and M. Zohner, "Aby - a framework for efficient mixed-protocol secure two-party computation," in *Network and Distributed System Security Symposium*, 2015.

[3] M. Keller, "Mp-spdz: A versatile framework for multi-party computation," Cryptology ePrint Archive, Paper 2020/521, 2020.

[4] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," *IACR Cryptol. ePrint Arch.*, p. 289, 2008.

[5] S. Zahur and D. Evans, "Obliv-c: A language for extensible data-oblivious computation," *IACR Cryptol. ePrint Arch.*, p. 1153, 2015.

[6] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow: Secure tensorflow inference," in *IEEE Symposium on Security and Privacy*. IEEE, May 2020.

[7] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 19–38.

[8] S. Wagh, D. Gupta, and N. Chandran, "Securenn: 3-party secure computation for neural network training," *Proceedings on Privacy Enhancing Technologies*, pp. 26–49, 2019.

[9] N. Chandran, D. Gupta, A. Rastogi, R. Sharma, and S. Tripathi, "Ezpc: Programmable and efficient secure two-party computation for machine learning," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2019, pp. 496–511.

[10] A. Shamir, "How to share a secret," *Commun. ACM*, p. 612–613, 1979.

[11] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, 1986, pp. 162–167.

[12] M. O. Rabin, "How to exchange secrets with oblivious transfer," *IACR Cryptol. ePrint Arch.*, p. 187, 2005.

[13] Y. Lindell, "Secure multiparty computation," *Commun. ACM*, p. 86–96, 2020.

[14] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y. an Tan, "Secure multi-party computation: Theory, practice and applications," *Information Sciences*, pp. 357–372, 2019.

[15] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Foundations and Trends® in Privacy and Security*, pp. 70–246, 2018.

[16] D. Kaplan, J. Powell, and T. Woller, "AMD memory encryption," White paper, 2016.

[17] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution." *Hasp@ isca*, 2013.

[18] T. Alves, "Trustzone: Integrated hardware and software security," *Information Quarterly*, pp. 18–24, 2004.

[19] J. I. Choi, K. R. Butler *et al.*, "Secure multiparty computation and trusted hardware: Examining adoption challenges and opportunities," *Security and Communication Networks*, 2019.

[20] R. Li, Q. Wang, Q. Wang, D. Galindo, and M. Ryan, "Sok: TEE-assisted confidential smart contract," *arXiv preprint arXiv:2203.08548*, 2022.

[21] R. Bahmani, M. Barbosa, F. Brasser, B. Portela, A.-R. Sadeghi, G. Scerri, and B. Warinschi, "Secure multiparty computation from SGX," in *Financial Cryptography and Data Security*, A. Kiayias, Ed. Springer International Publishing, 2017, pp. 477–497.

[22] F. Alder, N. Asokan, A. Kurnikov, A. Paverd, and M. Steiner, "S-faas: Trustworthy and accountable function-as-a-service using intel SGX," in *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, ser. CCSW'19. Association for Computing Machinery, 2019, p. 185–199.

[23] P. Wu, J. Ning, J. Shen, H. Wang, and E.-C. Chang, "Hybrid trust multi-party computation with trusted execution environment," 2022.

[24] D. Gupta, B. Mood, J. Feigenbaum, K. Butler, and P. Traynor, "Using intel software guard extensions for efficient two-party secure function evaluation," in *Financial Cryptography and Data Security*, J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds. Springer Berlin Heidelberg, 2016, pp. 302–318.

[25] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, ser. Lecture Notes in Computer Science, J. Feigenbaum, Ed. Springer, 1991, pp. 420–432.

[26] B. Ngabonziza, D. Martin, A. Bailey, H. Cho, and S. Martin, "Trustzone explained: Architectural features and use cases," in *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*. IEEE, 2016, pp. 445–451.

[27] C. Senarak, "Port cyberattacks from 2011 to 2023: a literature review and discussion of selected cases," *Maritime Economics & Logistics*, pp. 1–26, 2023.

[28] D. Lee, D. Jung, I. T. Fang, C.-C. Tsai, and R. A. Popa, "An {Off-Chip} attack on hardware enclaves via the memory bus," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[29] A. Moghimi, G. Irazoqui, and T. Eisenbarth, "Cachezoom: How SGX amplifies the power of cache attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, ser. Lecture Notes in Computer Science, W. Fischer and N. Homma, Eds. Springer, 2017, pp. 69–90.

[30] M. Lentz, R. Sen, P. Druschel, and B. Bhattacharjee, "Secloak: ARM trustzone-based mobile peripheral control," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2018, Munich, Germany, June 10-15, 2018*, J. Ott, F. Dressler, S. Saroiu, and P. Dutta, Eds. ACM, 2018, pp. 1–13.

[31] Y. Jang, J. Lee, S. Lee, and T. Kim, "SGX-bomb: Locking down the processor via rowhammer attack," in *Proceedings of the 2nd Workshop on System Software for Trusted Execution, SysTEX@SOSP 2017, Shanghai, China, October 28, 2017*. ACM, 2017, pp. 5:1–5:6.

[32] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller, "Cache attacks on intel SGX," in *Proceedings of the 10th European Workshop on Systems Security, EUROSEC 2017, Belgrade, Serbia, April 23, 2017*, C. Giuffrida and A. Stavrou, Eds. ACM, 2017, pp. 2:1–2:6.

[33] K. A. Küçük, A. Paverd, A. Martin, N. Asokan, A. Simpson, and R. Ankele, "Exploring the use of intel SGX for secure many-party applications," ser. SysTEX '16. Association for Computing Machinery, 2016.

[34] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious Multi-Party machine learning on trusted processors," in *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, 2016, pp. 619–636.

[35] F. Shaon, M. Kantarcioglu, Z. Lin, and L. Khan, "SGX-bigmatrix: A practical encrypted data analytic framework with trusted processors," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. Association for Computing Machinery, 2017, p. 1211–1228.

[36] S. Felsen, A. Kiss, T. Schneider, and C. Weinert, "Secure and private function evaluation with intel SGX," 11 2019, pp. 165–181.

[37] S. Tamrakar, J. Liu, A. Paverd, J.-E. Ekberg, B. Pinkas, and N. Asokan, "The circle game: Scalable private membership test using trusted hardware," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. Association for Computing Machinery, 2017, p. 31–44.

[38] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ser. ASIACCS '18. Association for Computing Machinery, 2018, p. 707–721.

[39] Y. Lu, B. Zhang, H.-S. Zhou, W. Liu, L. Zhang, and K. Ren, "Correlated randomness teleportation via semi-trusted hardware—enabling silent multi-party computation," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 699–720.

[40] Y. Jie, Y. Ren, Q. Wang, Y. Xie, C. Zhang, L. Wei, and J. Liu, "Multi-party secure computation with intel SGX for graph neural networks," in *IEEE International Conference on Communications, ICC 2022, Seoul, Korea, May 16-20, 2022*. IEEE, 2022, pp. 528–533.

[41] J. I. Choi, D. J. Tian, G. Hernandez, C. Patton, B. Mood, T. Shrimpton, K. R. B. Butler, and P. Traynor, "A hybrid approach to secure function evaluation using SGX," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, ser. Asia CCS '19, 2019, p. 100–113.

[42] B. Pinkas and T. Reinman, "Oblivious ram revisited," in *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*. Springer, 2010, pp. 502–519.

[43] W. Du and M. J. Atallah, *Protocols for Secure Remote Database Access with Approximate Matching*. Springer US, 2001, pp. 87–111.

[44] Y. Lu, B. Zhang, and K. Ren, "Low communication secure computation from semi-trusted hardware," *IEEE Transactions on Information Forensics and Security*, pp. 3962–3976, 2023.

[45] G. Dessouky, T. Frassetto, and A. Sadeghi, "Hybcache: Hybrid side-channel-resilient caches for trusted execution environments," in *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, S. Capkun and F. Roesner, Eds. USENIX Association, 2020, pp. 451–468.

[46] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. Lai, "Sgxpectre: Stealing intel secrets from SGX enclaves via speculative execution," in *IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019*. IEEE, 2019, pp. 142–157.

[47] W. Huang, S. Xu, Y. Cheng, and D. Lie, "Aion attacks: Manipulating software timers in trusted execution environment," in *Detection of Intrusions and Malware, and Vulnerability Assessment - 18th International Conference, DIMVA 2021, Virtual EuroS&P Event, July 14-16, 2021, Proceedings*, ser. Lecture Notes in Computer Science, L. Bilge, L. Cavallaro, G. Pellegrino, and N. Neves, Eds. Springer, 2021, pp. 173–193.

[48] Y. Wang, R. Paccagnella, E. T. He, H. Shacham, C. W. Fletcher, and D. Kohlbrenner, "Hertzbleed: Turning power Side-Channel attacks into remote timing attacks on x86," in *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, 2022, pp. 679–697.

[49] S. Johnson, V. Scarlata, C. Rozas, E. Brickell, and F. Mckeen, "Intel software guard extensions: Epid provisioning and attestation services," *White Paper*, p. 119, 2016.

[50] B. Parno, J. R. Lorch, J. R. Douceur, J. W. Mickens, and J. M. McCune, "Memoir: Practical state continuity for protected modules," in *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*. IEEE Computer Society, 2011, pp. 379–394.

[51] R. Strackx and F. Piessens, "Ariadne: A minimal approach to state continuity," in *25th USENIX Security Symposium (USENIX Security)*, 2016, pp. 875–892.

[52] B. Parno, J. R. Lorch, J. R. Douceur, J. Mickens, and J. M. McCune, "Memoir: Practical state continuity for protected modules," in *2011 IEEE Symposium on Security and Privacy*, 2011, pp. 379–394.

[53] R. Strackx, B. Jacobs, and F. Piessens, "Ice: A passive, high-speed, state-continuity scheme," in *Proceedings of the 30th Annual Computer Security Applications Conference*, ser. ACSAC '14. Association for Computing Machinery, 2014, p. 106–115.

[54] S. Matetic, M. Ahmed, K. Kostiainen, A. Dhar, D. Sommer, A. Gervais, A. Juels, and S. Capkun, "ROTE: Rollback protection for trusted execution," in *26th USENIX Security Symposium (USENIX Security)*, 2017, pp. 1289–1306.

[55] J. Niu, W. Peng, X. Zhang, and Y. Zhang, "Narrator: Secure and practical state continuity for trusted execution in the cloud," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. Association for Computing Machinery, 2022, p. 2385–2399.

[56] W. Wang, S. Deng, J. Niu, M. K. Reiter, and Y. Zhang, "Engraft: Enclave-guarded raft on byzantine faulty nodes," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22, New York, NY, USA, 2022, p. 2841–2855.

[57] J. Lind, I. Eyal, F. Kelbert, O. Naor, P. R. Pietzuch, and E. G. Sirer, "Teechain: Scalable blockchain payments using trusted execution environments," *CoRR*, vol. abs/1707.05454, 2017.

[58] A. Oprea and M. K. Reiter, "Integrity checking in cryptographic file systems with constant trusted storage," in *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007*, N. Provos, Ed. USENIX Association, 2007.

[59] G. Kaptchuk, M. Green, and I. Miers, "Giving state to the stateless: Augmenting trustworthy computation with ledgers," in *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[60] R. Strackx and F. Piessens, "Ariadne: A minimal approach to state continuity," in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, T. Holz and S. Savage, Eds. USENIX Association, 2016, pp. 875–892.

[61] M. K. Jangid, G. Chen, Y. Zhang, and Z. Lin, "Towards formal verification of state continuity for enclave programs," in *30th USENIX Security Symposium (USENIX Security)*, 2021, pp. 573–590.

[62] B. Dinis, P. Druschel, and R. Rodrigues, "RR: A fault model for efficient TEE replication," in *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023.

[63] S. Angel, A. Basu, W. Cui, T. Jaeger, S. Lau, S. Setty, and S. Singanamalla, "Nimble: Rollback protection for confidential cloud services," in *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*. USENIX Association, 2023, pp. 193–208.

[64] M. Brossard, G. Bryant, B. E. Gaabouri, X. Fan, A. Ferreira, E. Grimley-Evans, C. Haster, E. Johnson, D. Miller, F. Mo, D. P. Mulligan, N. Spinale, E. V. Hensbergen, H. J. M. Vincent, and S. Xiong, "Private delegated computations using strong isolation," *CoRR*, 2022.

[65] "Confidential Computing: Hardware-Based Trusted Execution for Applications and Data," Confidential Computing Consortium, Tech. Rep., 2022.

[66] A. Erwig, S. Faust, S. Riahi, and T. Stöckert, "Commitee : An efficient and secure commit-chain protocol using TEEs," in *8th IEEE European Symposium on Security and Privacy, EuroS&P 2023, Delft, Netherlands, July 3-7, 2023*. IEEE, 2023, pp. 429–448.

[67] R. Khalil, A. Zamyatin, G. Felley, P. Moreno-Sanchez, and A. Gervais, "Commit-chains: Secure, scalable off-chain payments," *Cryptology ePrint Archive*, 2018.

[68] D. Natarajan, W. Dai, and R. G. Dreslinski, "CHEX-MIX: combining homomorphic encryption with trusted execution environments for two-party oblivious inference in the cloud," *IACR Cryptol. ePrint Arch.*, p. 1603, 2021.

[69] B. Fisch, D. Vinayagamurthy, D. Boneh, and S. Gorbunov, "IRON: functional encryption using intel SGX," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM, 2017, pp. 765–782.