A Survey of TEE and its Applications

Log Creative

June 2nd, 2024

Contents

1 SOTA of TEE 1
1.1 Intel TDX 2
1.2 AMD SEV-SNP 2
1.3 Arm CCA 3
1.4 RISK-V: Penglai 4
2 Applications of TEE 4
2.1 Cloud computing: Cyptomator 4
2.2 Blockchain: Twilight 6
2.3 AI: PPFL 7
3 Vulnerabilities of TEE 8
3.1 Side channel attacks 8
3.2 Rewind attacks 8

1 SOTA of TEE

TEE (Trusted Execution Environments) provides an isolated environment (called secure enclave) that safeguards processed data by encrypting the incoming and outgoing data. Furthermore, TEE provides mechanisms to ensure the computation is correctly executed with an integrity guarantee. More importantly, TEE protects the data and computation against any potentially malicious entity residing in the system (including the kernel, hypervisor, etc.) [1].

The current state-of-the-art (SOTA) of TEE are Intel TDX, AMD SEV-SNP, Arm CCA, and RISK-V-based Penglai.

1.1 Intel TDX

Intel TDX (Trust Domain Extensions) [2] is Intel's newest confidential computing technology, shown in Figure 1. This hardware-based trusted execution environment (TEE) facilitates the deployment of trust domains (TD), which are hardware-isolated virtual machines (VM) designed to protect sensitive data and applications from unauthorized access. The key features are as follows:

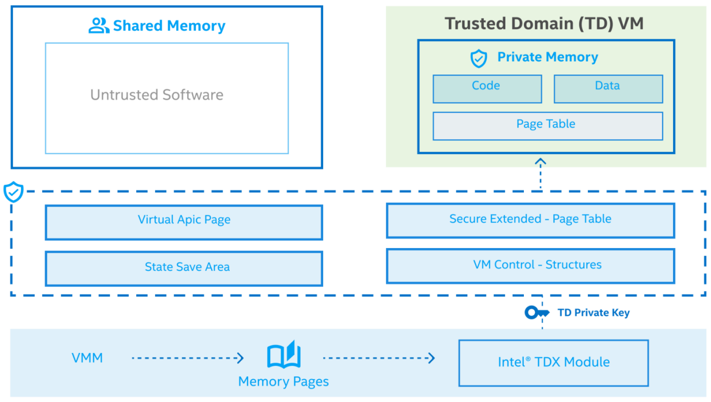

Figure 1 Intel TDX Architecture

- **IO virtualization.** It introduces a method for direct device assignment to TEEs without the need for shared buffers, enhancing both functionality and performance.
- **Security model.** The architecture ensures that only the TEE Device owner can decide which TEE-IO device interface is trustworthy, with strict access controls and data isolation.
- **Trusted execution.** Intel TDX Connect extends the TDX hardware with private MMIO and DMA access control, along with end-to-end data protection using PCIe selective IDE streams.
- **Architecture overview.** It provides a comprehensive framework for establishing trust between a TD (TEE Device) and a TDI (TEE Device Interface), securing the data path, and supporting the TDISP (TEE Device Interface Security Protocol) assignment and removal lifecycle.

1.2 AMD SEV-SNP

AMD SEV-SNP (Secure Encrypted Virtualization–Secure Nested Paging) [3] builds upon existing SEV and SEV-ES functionality while adding new hardware-based security protections. SEV-SNP adds strong memory integrity protection to help prevent malicious hypervisor-based attacks like data replay, memory re-mapping, and more in order to create an isolated execution environment. Also, SEV-SNP introduces several additional optional security enhancements designed to support additional VM use models, offer stronger protection around interrupt behavior, and offer increased protection against recently disclosed side channel attacks. The key features are as follows:

- **Memory integrity protection.** SEV-SNP enhances VM isolation by adding strong memory integrity protection to prevent malicious attacks like data replay and memory re-mapping.

- **Enhanced security features.** It offers optional protections against malicious interrupt injection, speculative side channel attacks, and rollback attacks on trusted computing base (TCB) components.
- **Flexible VM privilege levels.** The architecture introduces Virtual Machine Privilege Levels (VMPLs) that allow a VM to divide its address space into four hierarchical levels for additional security controls.
- **Advanced attestation and migration.** SEV-SNP supports flexible attestation, enabling VMs to request attestation reports at any time, and introduces an improved migration policy with the involvement of a Migration Agent (MA).

However, Paradžik *et al.* found attacks on several authentication and attestation properties, including **the compromise of attestation report integrity**. They also discover a vulnerability that **allows a cloud provider to trick a third party** into incorrectly believing that an SNP-protected guest is running on a platform with secure, up-to-date firmware. Slight modifications to the design which let third parties detect guest migrations to vulnerable platforms are proposed in [4].

1.3 Arm CCA

Arm proposed the design of Arm Confidential Compute Architecture (CCA) [5] shown in Figure 2 and its hardware security primitive — Realm Management Extensions (RME) — to support confidential computing in next-generation Arm devices. CCA introduces **realms**, the basic unit of the confidential computing environment, and the Realm Management Monitor (RMM), which behaves as a thin hypervisor for realm isolation. To mitigate the latency in realm execution, an untrusted hypervisor is delegated to schedule the realms and manage memory resources, but cannot access realms [6].

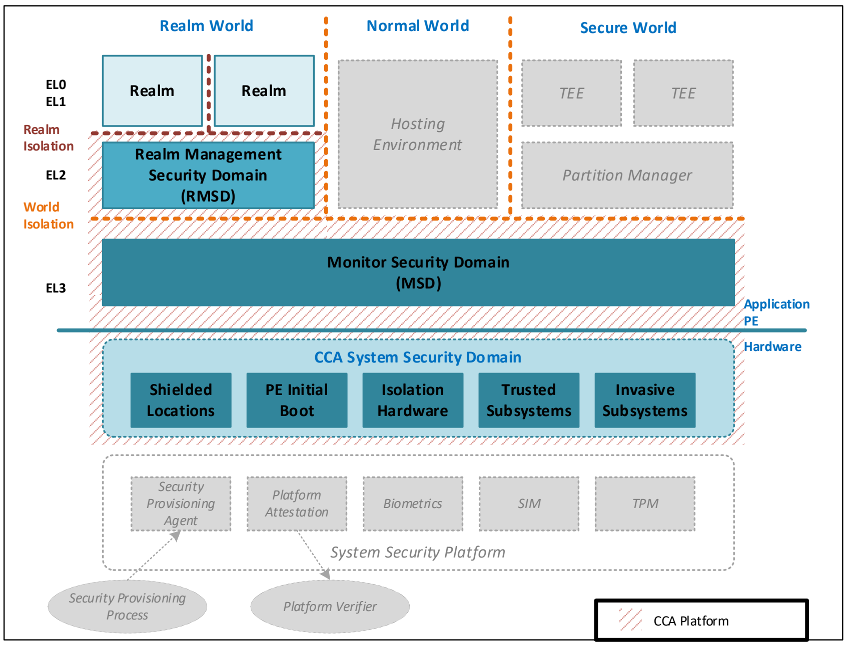

Figure 2 Elements of CCA [7]

1.4 RISK-V: Penglai

RISK-V-based Penglai [8] is an open-source RISC-V enclave system using a secure monitor to manage all the enclaves shown in Figure 3. It first introduces two **new architectural primitives**, Guarded Page Table (GPT) and Mountable Merkle Tree (MMT): GPT protects page table pages and enables memory isolation with page-level granularity, and MMT is a new abstraction to achieve on-demand and scalable memory encryption and integrity protection. Leveraging these two primitives, a lightweight secure monitor running in the most privileged mode is in charge of enclave management and maintaining security guarantees. To mitigate the high overhead of enclave creation due to costly secure memory initialization, we propose a new type of enclave, called shadow enclave, to support fork-style **fast enclave creation**.

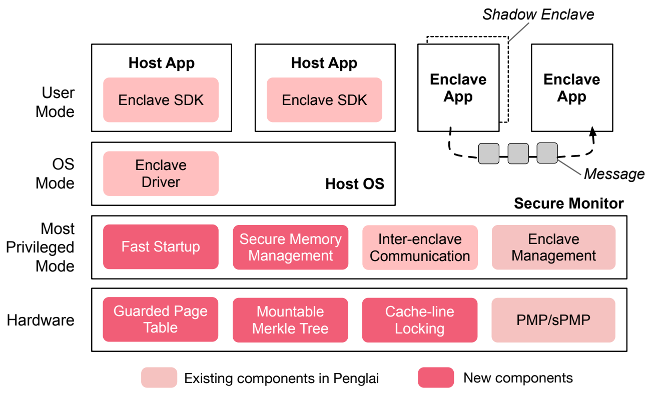

Figure 3 Overview of Penglai Architecture

2 Applications of TEE

Use cases for TEEs start evolving in the areas of cloud computing, blockchain, and artificial intelligence [9]. There are selected applications for each area: for cloud computing, Twilight for blockchain, PPFL for AI.

2.1 Cloud computing: Cyptomator

With the evolution of computer systems, the amount of sensitive data to be stored as well as the number of threats on these data grow up, making the data confidentiality increasingly important to computer users. Currently, with devices always connected to the Internet, the use of cloud data storage services has become practical and common, allowing quick access to such data wherever the user is. Such practicality brings with it a concern, precisely the confidentiality of the data which is delivered to third parties for storage. In the home environment, disk encryption tools have gained special attention from users, being used on personal computers and also having native options in some smartphone operating systems.

In [10], da Rocha. *et al.* uses the data sealing, feature provided by the Intel Software Guard Extensions (Intel SGX) technology, for file encryption. A virtual file system (**CryptoFS**) is created in which applications can store their data, keeping the security guarantees provided by the Intel SGX technology, before sending the data to a storage provider. This way, even if the storage provider is compromised, the data are safe. To validate the proposal, the **Cryptomator** software, which is a free client-side encryption tool for cloud files, was integrated with an Intel SGX application (enclave) for data sealing. The results demonstrate that the solution is feasible, in terms of performance and security, and can be expanded and refined for practical use and integration with cloud synchronization services.

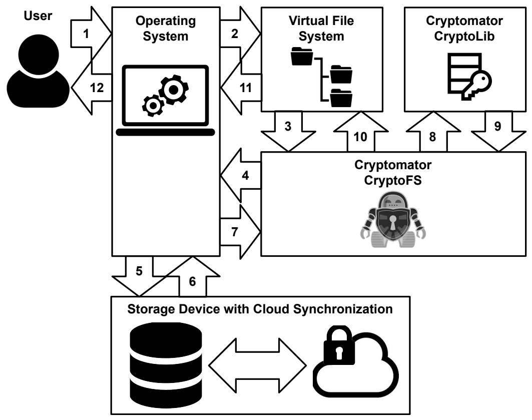

Figure 4 Workflow for reading and decrypting stored data with Cryptomator

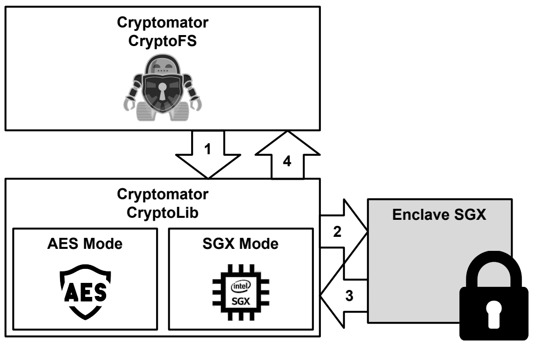

Figure 5 Workflow performed for data encryption and decryption through the Cryptomator

The process of reading the files provided by the application is shown in Fig 4: (1) The user requests the operating system to open a file; (2) The operating system requests FUSE for file data. FUSE allows the userspace applications export a filesystem to the Linux kernel, with functions to mount the file system, unmount it and communicate with kernel; (3) FUSE forwards this request to the Cryptomator, using the CryptoFS library; (4) Cryptomator requests the operating system to have the file data fetched from the storage device; (5) The operating system locates the data; (6) These data are loaded into the main memory; (7) The operating system provides these data to the Cryptomator; (8) The CryptoFS library sends the encrypted data to the CryptoLib library; (9) The CryptoLib library decrypts the received data and returns them to CryptoFS library; (10) The CryptoFS library sends the decrypted data to FUSE; (11) FUSE forwards such data to the operating system; (12) Finally, the operating system provides the user with the decrypted file.

Fig 5 describes the process of sending and receiving data between CryptoFS and CryptoLib libraries, which includes two additional steps for communication between the CryptoLib library and the SGX enclave. The shaded box indicates a trusted execution environment (SGX in this case), where **the key is manipulated** and **the data are encrypted and decrypted**. The communication flow is explained below: (1) CryptoFS library calls encrypt/decrypt functions in CryptoLib library, which contains AES and SGX modes; (2) In SGX mode, CryptoLib library creates an enclave and sends to it each data block to perform the encrypt/decrypt process; (3) Encrypt/decrypt blocks are send back to CryptoLib; (4) Encrypt/decrypt data forward to CryptoFS library.

2.2 Blockchain: Twilight

Payment channel networks (PCNs) provide a faster and cheaper alternative to transactions recorded on the blockchain. Clients can trustlessly establish payment channels with relays by locking coins and then send signed payments that shift coin balances over the network's channels. Although payments are never published, anyone can track a client's payment by monitoring changes in coin balances over the network's channels.

Twilight [11] is the first PCN that provides a **rigorous differential privacy guarantee** to its users. Relays in Twilight run a noisy payment processing mechanism that **hides the payments they carry**. This mechanism increases the relay's cost, so Twilight **combats selfish relays** that wish to avoid it using a TEE that ensures they follow its protocol. The TEE **does not store the channel's state**, which minimizes the trusted computing base. Crucially, Twilight ensures that even if a relay breaks the TEE's security, it cannot break the integrity of the PCN. Dotan *et al.* implement Twilight using Intel's SGX framework and evaluate its performance using relays deployed on two continents and show that a route consisting of 4 relays handles 820 payments/sec.

In detail, protocol messages and relay-to-TEE interface are shown in Figure 6.

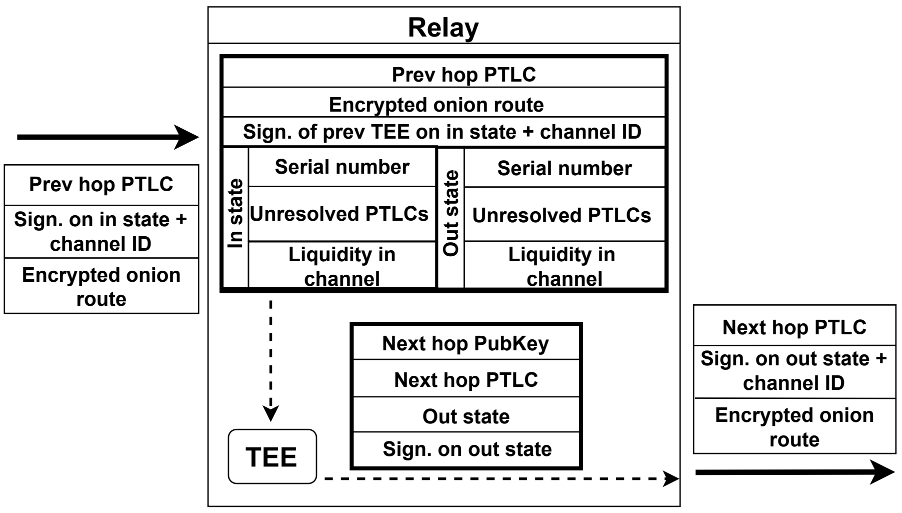

Figure 6 Protocol messages and relay-to-TEE interface in Twilight

Establishing public keys and channel IDs. TEEs generate secret keys, and the corresponding public keys are bound to payment channels. When a new payment channel is established, each party provides a public key to manage the account via a smart contract. Clients can verify the payment processing logic within the TEE using remote attestation. The smart contract's account address serves as the channel ID, ensuring that messages about the channel are unique and secure.

Processing payments inside TEEs. Relays receive and forward encrypted payments without knowing the outcome of the payment processing. Payments are conditioned on a secret known only to the payee, using

Private Time-Locked Contracts (PTLCs). The TEE decides whether to approve a payment based on the current balance and pending payments, without storing the channel's state. Payments are encrypted for the next TEE in the route, preventing relays from adjusting the noise distribution to the payment.

Channel teardown. Channels close by posting a transaction to the blockchain, splitting the locked coins between the channel endpoints. Parties exchange signed closing balance messages after each payment, which are authorized by the TEE's secret key. The smart contract allows for an appeal period where parties can dispute the closing balance with higher serial number messages. Twilight can operate over blockchains that support private transactions, hiding the channel's closing balance split.

2.3 AI: PPFL

PPFL (Privacy-preserving Federated Learning with TEE) [12] is a framework for mobile systems to limit privacy leakages in federated learning. It utilizes TEEs on clients for local training, and on servers for secure aggregation, so that model/gradient updates are hidden from adversaries. Figure 7 provides an overview of the framework and the various steps of the **greedy layer-wise training and aggregation**, which allows clients to collaboratively train a DNN model while keeping the model's layers **always inside TEEs during training**. In general, starting from the first layer, each layer is trained until convergence, before moving to the next layer. In this way, PPFL aims to achieve full privacy preservation without significantly increasing system costs.

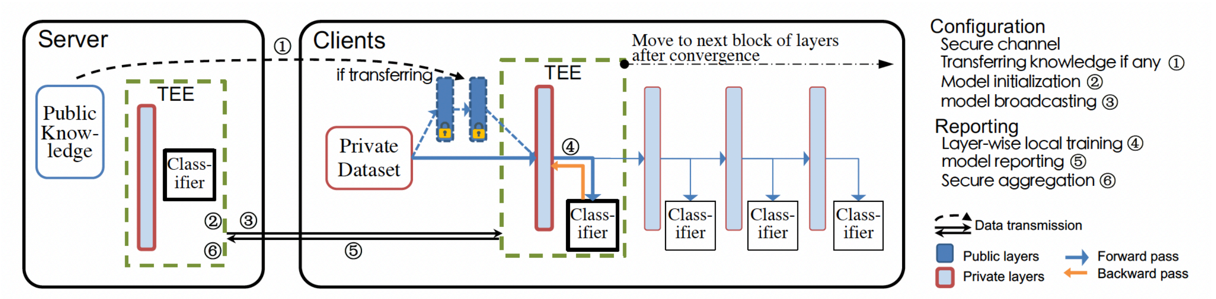

Figure 7 A schematic diagram of the PPFL framework

The brief process is as follows: In PPFL, the server can learn from public models. Thus, during initialization, the server first chooses a model pre-trained on public data that has a similar distribution with private data. The server keeps the first layers, removes the last layer(s), and assembles new layer(s) atop the reserved first ones. These **first layers** are transferred to clients and are always kept frozen (step ①). The server configures the model architecture, decides the layers to be protected by TEEs, and then **initializes model parameters** inside the TEE (step ②). The latter ensures clients' local training starts with the same weight distribution. In addition, the server **configures other training hyper-parameters** such as learning rate, batch size, and epochs, before transmitting such settings to the clients (step ③). After model transmission and configuration using secure channels, each client starts **local training** on their data on each layer via a model partitioned execution technique (step ④). Once local training of a layer is completed inside TEEs, all participating clients

report the layer parameters to the server through secure channels (step ⑤). Finally, the server securely **aggregates the received parameters** within its TEE and applies FedAvg, resulting in a new global model layer (step ⑥).

3 Vulnerabilities of TEE

Despite its appealing features, TEE is also vulnerable to several attacks, and the survey in [13] discusses these attacks in detail. There are two major attacks that would affect secure computation: side-channel attacks and rewind attacks.

3.1 Side channel attacks

First, TEE may suffer from side-channel attacks which could breach data privacy. They are mainly from timing, memory, and network side channels.

Timing side channel [14]. Given different inputs, the running time of the program in the enclave may be different too. By **monitoring the difference over the execution time**, an attacker may derive some sensitive information about the input, e.g., a secret value reflecting how many times a loop is executed. To solve this problem, one approach is to ensure the program loaded into the enclave **always takes approximately the same amount of execution time**.

Memory side channel [15]. In some programs, **the access pattern** to the non-PRM memory **may depend on the enclave's secret**. For example, if the enclave runs a binary search program, the access pattern is a path from the root node to a leaf node, which may reveal the real secret. Therefore the loaded program should **remove any correlation** between the memory access pattern and the insider secret.

Network side channel [16]. In a distributed system, **network access patterns** (e.g., sorting or hash partitioning) may reveal sensitive information. Therefore, the network access protocol should also be well-designed such that no sensitive information can be deduced from the patterns.

3.2 Rewind attacks

TEEs may also be vulnerable to rewind attacks (or replay attacks), which allows an attacker to actively **reset the state of the computation** [17]. Note that such attacks may result in catastrophic consequences for stateful computation, such as limited-attempt password checking. We cannot always expect the TEE to maintain the state securely for two reasons: (1) TEE may be stateless in practice; (2) even for stateful TEE, the limited protected memory makes it fail to correctly maintain a large state. Therefore, the designer should be careful and provides special mechanisms to counter rewind attacks.

References

[1] X. Li, B. Zhao, G. Yang, T. Xiang, J. Weng, and R. H. Deng, "A survey of secure computation using trusted execution environments," *arXiv preprint arXiv:2302.12150*, 2023.

[2] Intel, *Intel Trust Domain Extensions*, Feb. 2022. [Online]. Available: https://cdrdv2.intel.com/v1/dl/getContent/690419.

[3] AMD, *AMD SEV-SNP: Strengthening vm isolation with integrity protection and more*, Jan. 2020. [Online]. Available: https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf.

[4] P. Paradžik, A. Derek, and M. Horvat, "Formal security analysis of the amd sev-snp software interface," *arXiv preprint arXiv:2403.10296*, 2024.

[5] Arm, *Learn the architecture - introducing arm confidential compute architecture*, Jun. 2023. [Online]. Available: https://developer.arm.com/documentation/den0125/0300/.

[6] C. Wang, F. Zhang, Y. Deng, *et al.*, "Cage: Complementing arm cca with gpu extensions," in *Proceedings of the 31st Annual Network and Distributed System Security Symposium*, 2024.

[7] Arm, *Arm cca security model 1.0*, Aug. 2021. [Online]. Available: https://developer.arm.com/documentation/DEN0096/latest/.

[8] E. Feng, X. Lu, D. Du, *et al.*, "Scalable memory protection in the PENGLAI enclave," in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, USENIX Association, Jul. 2021, pp. 275–294, ISBN: 978-1-939133-22-9. [Online]. Available: https://www.usenix.org/conference/osdi21/presentation/feng.

[9] T. Geppert, S. Deml, D. Sturzenegger, and N. Ebert, "Trusted execution environments: Applications and organizational challenges," *Frontiers in Computer Science*, vol. 4, p. 930 741, 2022.

[10] M. da Rocha., D. C. G. Valadares., A. Perkusich., K. C. Gorgonio., R. T. Pagno., and N. C. Will., "Secure cloud storage with client-side encryption using a trusted execution environment," in *Proceedings of the 10th International Conference on Cloud Computing and Services Science - CLOSER*, INSTICC, SciTePress, 2020, pp. 31–43, ISBN: 978-989-758-424-4. DOI: 10.5220/0009130600310043.

[11] M. Dotan, S. Tochner, A. Zohar, and Y. Gilad, "Twilight: A differentially private payment channel network," in *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA: USENIX Association, Aug. 2022, pp. 555–570, ISBN: 978-1-939133-31-1. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/dotan.

[12] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, "Ppfl: Privacy-preserving federated learning with trusted execution environments," in *Proceedings of the 19th annual international conference on mobile systems, applications, and services*, 2021, pp. 94–108.

[13] S. Fei, Z. Yan, W. Ding, and H. Xie, "Security vulnerabilities of sgx and countermeasures: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–36, 2021.

[14] 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: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers 20*, Springer, 2016, pp. 302–318.

[15] S. Sasy, S. Gorbunov, and C. W. Fletcher, “Zerotrace: Oblivious memory primitives from intel sgx,” *Cryptology ePrint Archive*, 2017.

[16] W. Zheng, A. Dave, J. G. Beekman, R. A. Popa, J. E. Gonzalez, and I. Stoica, “Opaque: An oblivious and encrypted distributed analytics platform,” in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 283–298.

[17] M. Bellare, M. Fischlin, S. Goldwasser, and S. Micali, “Identification protocols secure against reset attacks,” in *Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings 20*, Springer, 2001, pp. 495–511.