Network Applications of Trusted Execution Environments

Tim Kruse, Florian Wiedner*, Marcel Kempf*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: tim.kruse@tum.de, wiedner@net.in.tum.de, kempfm@net.in.tum.de

***Abstract*—As the interest in security in the networking community steadily increases, so does the interest in applying Trusted Execution Environments (TEE). However, despite the increased usage of TEEs, there is little information on how they are actually employed. To shed more light on an important tool for securing networking and its applications, we will present usages of the technology with a focus on networking. We found several proposals to utilize TEEs for networking applications in research, such as TrustedGateway, where the entire traffic is routed through the TEE. There are usages that enhance privacy by encrypting and decrypting data directly in the TEE. Finally, there are uses in cloud computing. We find that the usage of TEEs in networking applications is not that common at this time but seems to be a topic of active research.**

***Index Terms*—trusted execution environment, cloud computing, networking**

1. Introduction

Security in software products is of increasing importance, with bad actors aiming to gain access to data. There are many approaches to fortify different services and programs running on the network. One of these approaches is to employ Trusted Execution Environments (TEE), which aim to separate selected processes from the other parts of the machine. The processes are isolated from the system, which ensures that neither their code and data nor their execution can be compromised. Since the world is very connected, we want to survey the usage of TEEs to find how widely and for what purpose they are used in scientific projects in networking or networked applications. We will also determine whether there are any approaches that use TEEs in such a context in the field. For these approaches, we will show what the impact of the inclusion of a TEE is, with a focus on networking aspects, such as the impact on performance, e.g., latency or bandwidth, that can be expected. In this paper, we will first give a brief explanation of TEE, which will be followed by a summary of some interesting and diverse applications of TEEs in the space of networking and distributed systems. One interesting approach is to utilize a TEE to secure a gateway, which is called **TrustedGateway**. In this approach, all packets are passed through the TEE to applications in the cloud space. Other approaches use TEEs as a privacy-preserving measure. Since TEEs are present in most architectures these days, as Intel, AMD, and ARM have implemented them in hardware, their use will most likely continue to increase as the formation of the **Confidential Computing Consortium (CCC)** seems to indicate. The three manufacturers, as well as large companies in the cloud space, such as Microsoft and Google, are a part of this organization. We will first start with an outlook on related work in Section 2, which will give an overview of what TEEs are and also provide some information on the CCC. This will be followed by Section 3, where we will showcase applications that we found, such as TrustedGateway, which runs all traffic through a TEE. We will also take a look at applications that utilize a TEE to provide privacy. We will give information about the current usage of TEEs in cloud computing. In Section 4, a summary of the usages we found will be given, alongside our opinion on which topics warrant further research interest.

2. Related Work

In this section, we will present key components that will be relevant to this survey in order to enable the reader to follow the applications and reasons for their usage. Furthermore, we provide a starting point to conduct further research into the field.

2.1. Trusted Execution Environment

A TEE is a tamper-resistant software environment that is part of the processor. It aims to provide trust that a piece of code is executed as it should. According to Sabt et al. [1], it relies on a chain of trust that is established during the boot process in order to ensure that the environment can establish the authenticity and confidentiality of the code executed within it. Furthermore, the TEE provides integrity protection and an attestation mechanism to provide proof for the execution. Both the TEE and the rich environment, which houses the operating system, run on a separation kernel to isolate them from each other. To establish the TEE, Sabt et al. [1] propose the following five key building blocks, as shown in Figure 1:

- **Secure Boot** Ensures that if code is modified, it is detected and the chain of trust is seen as broken, provides a **Trusted Computing Base (TCB)**, which encompasses all security-critical hardware and software of a system
- **Secure Scheduling** Provides Scheduling to ensure that the rich environment is responsive
- **Inter-Environment Communication** Exchanges data between the rich environment and the TEE

 doi: 10.2313/NET-2024-04-1_12

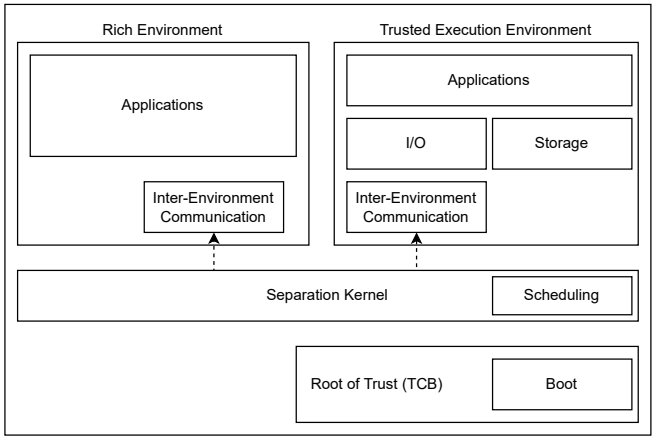

Figure 1: Building Blocks of a TEE, adapted from [1]

- **Secure Storage** Utilizes authenticated encryption, can only be used by the TEE
- **Trusted I/O Path** Provides trust for the peripherals to ensure that their input is not changed

There are many different TEEs available at this time, but the most important ones are Intel SGX [2], and its extension Intel TDX [3], AMD's SEV [4], and ARM's TrustZone [5]. The first two are mostly found in devices such as servers and PCs, while the ARM implementation is found in mobile phones and commodity gateways, among others [6]. These products are in continuous development as attack vectors are found over time. This led to the extension of Intel SGX with Intel TDX. The products are dependent on specific hardware, which leads to issues with the portability of code, as the interfaces differ between vendors. This resulted in efforts to standardize these interfaces [7], though different approaches, such as Open-TEE [8], which offers a virtual TEE to develop code and compile it for different TEEs, were also developed. TEEs aim to shield from all software and hardware attacks, but there are some attack vectors that exist despite these efforts, such as side-channel attacks, as shown by Wang et al. [9]. Despite that, the number of applications for TEEs seems to be ever-increasing, as research by Geppert et al. [10] shows. They tried to ascertain use cases and challenges for the use of TEEs, with a focus on cloud computing use cases.

2.2. Confidential Computing

In 2019, the Confidential Computing Consortium was created to advance the security of user data in the cloud [11], to describe different means to ensure the security and privacy of programs and data by utilizing TEE, as outlined in their report [12]. One of their aims is to provide secure and private ways to allow multiple physical nodes to compute tasks, as is common in the cloud space, for which they use TEEs to provide confidentiality as well as integrity. The consortium consists of large companies in the space. Key among them are the big hardware vendors Intel, AMD, and ARM, together with cloud operators such as Microsoft and Google [13]. They have outlined multiple use cases in their paper [12], which they want to employ to improve the security and privacy of "data in use" [12]. They aim to utilize the advantages of TEEs not only for normal cloud operations but also to facilitate the increased use of multi-party computation. This is aimed at data that can not be given to another party due to privacy concerns. In this setup, computation can still be done efficiently without compromising the data's security and privacy.

3. Applications of TEE in Networking

In the following section, we will give an overview of some of the uses of TEEs in a networking context.

3.1. TrustedGateway

One application by Schwarz [6] aims to increase the security of gateways. They deem the security of commodity gateways under threat, since most gateways have an increasing number of additional, less secure services running that offer services like FTP or a VPN into the network guarded by the gateway. Since it is possible to compromise the gateway via these additional services, these attack vectors could be used to compromise the core tasks of the gateway, as described in their paper. As a solution to this problem, they move all network traffic through a TEE. To achieve their goal, they implemented the minimum needed for securely networking traffic through the ARM TrustZone TEE. In order to keep performance as good as possible and limit the amount of TCB required, they only implemented switching, routing, and the firewall into the TEE.

3.1.1. Design of TrustedGateway. Schwarz [6] proposes two parts to realize their goal. The first is a networking utility called **NetTrug**, which performs the networking tasks. The second is called **ConfigService**, which is used to configure NetTrug. Since we are primarily interested in networking and not secure policy configuration, we will focus on the NetTrug utility. The entire project is implemented using **Open-TEE (OP-TEE)**. It is composed of two major parts, a partial networking driver we trust and its untrusted counterpart running in the rich environment. The second part is the I/O workers, which remove the need for context switches by handling network interrupts. NetTrug is the only application of the system that has direct access to the network interface cards (NIC), which are then considered trusted interfaces. There can be untrusted interfaces, but these will still have to run through NetTrug. The traffic is filtered by a stateful L3/L4 firewall, in their case NPF [14], but the utility itself has no complete protocol stack. To make the filtering transparent for the applications running in the rich environment, a virtual network interface card (VNIC), which is based on **virtio-net** and **virtio-mmio**, with input and output queues, is used to connect NetTrug to the rich environment. To prevent spoofing from both a system or outside attacker, the following mechanism is in place: The IP addresses for all NICs are static. Whenever a packet is sent or received, the MAC address of the NIC is used. This limits inbound traffic as well, as only traffic sent to one of the IP addresses of the TrustedGateway is accepted. The network stack deployed in the TEE is minimal. It offers Address Resolution Protocol (ARP) resolution for itself and also checks the traffic for ARP spoofing attacks originating

 doi: 10.2313/NET-2024-04-1_12

from both the rich environment and an outside attacker. It also replaces the MAC for the VNIC, which it swaps into the destination spot when forwarding the traffic. Since NetTrug does not offer an IP or UDP stack, to keep the TCB small, it also does not offer DHCP or DNS services, and the NIC's IP addresses are statically assigned. The security of the configuration of NetTrug must also be ensured, as otherwise, the entire effort would be for naught. This is done by the second part of the TrustedGateway, the ConfigService. It is an OP-TEE application, which is binary-signed, integrity-checked, and protected against version rollbacks. In order to protect it against attackers, it only allows trusted devices to configure it, which is done by the HTTPS client in the TEE. This client only performs the TLS portion, while the TCP portion is performed in the rich environment. It also utilizes custom request headers to avoid cross-site request forgery attacks. There is a master certificate that is uploaded at a physically attached interface when first configuring the service, and the service itself creates one when it is first started. The master admin is the only entity that can add or remove further admin certificates. Within this utility, it is possible to define rules for the firewall or assign different IP addresses for the NICs.

3.1.2. Performance of TrustedGateway. Schwarz poses in their paper [6] that due to TrustedGateway extracting only the strictly necessary services into the trusted environment, the TCB required is reduced compared to a full stack in an operating system, as it offers less functionality. This, in turn, reduces the likelihood of vulnerabilities. Furthermore, the performance impact of the setup on a TCP connection was measured. To have a comparable baseline setup for experimentation, they disabled NIC offloading, as it is not implemented in TrustedGateway. They set up three different experiments using iPerf3 [15], with the first two running a receiver (1) and a sender (2) in the gateway's rich environment, respectively. The third experiment was set up such that the gateway had to forward the traffic to a host outside the network connected to the gateway or receive traffic through the gateway. The key performance metrics they observed were the network throughput, the network latency, and the overhead of the firewall inside the TEE. In experiment (1), they noticed that the throughput was roughly 90% of the normal throughput with 385 Mbit/s, with experiment (3) delivering roughly the same results. In experiment (2), in some cases, higher throughput than the baseline of 369 Mbit/s could be observed. In experiment (3), the performance with the sender inside the network was at around 93% of the baseline, which equates to 236 Mbit/s, and when receiving, reached between 101.9% - 103.5% of the baseline with 221 Mbit/s. The firewall overhead was also observed in the same experiment setup and was in the range of 0.5% to 1%, though the baseline value is not mentioned, besides the result being called small. The author attributes it to NPF's static code. Another very important metric was latency overhead, as it is critical for low-latency networked applications, such as phone calls. To test the latency in a user-like manner, they loaded different websites from the Tranco List [16], which is a list of websites that should be used in research, as they are aimed to be hardened against manipulation. The latency overhead was around 3.4% on average, peaking at 4.95%, though no baseline latency is given. The latency is attributed to the TrustedGateway workers having to be woken up, which the author tried to avoid by employing an idle grace period. To evaluate low latency performance, they sent ping packets from a host inside the network to an outside host via the gateway. In this case, the average overhead was even lower, with around 2.7%, which equates to 0.37ms of added latency when utilizing the TrustedGateway compared to the stock setup. The config service is also responsive despite the fact that it runs in both the rich and trusted execution environment, with a load time of around 1-2 seconds. Since the TCB is rather small, with around 110k lines of code (LOC), rather than the 4523k LOC included in the router's default operating system, including both the OP-TEE and its cryptography library, the memory it needs is fairly small, with 32 MB of which 20 MB are for trusted user apps, which could offer other functionalities. This small code base should increase security, as larger code bases are more susceptible to issues simply due to their size. Overall, the usage of a TEE for this purpose seems sensible, especially given the fact that dedicated gateways are expensive, and this allows for secure operation of the main purpose, namely forwarding and protecting the network behind it.

3.2. TEE for Privacy

Since TEEs can provide confidentiality for the data that they use as well as prove the integrity of their computation, they are very useful for maintaining privacy. Multiple proposals in the networking space, therefore, utilize TEEs to provide privacy. We are going to focus on two specific applications that show the breadth that TEEs can be used for. In the first section, we are going to present a TEE utilized by Risdianto et al. [17] to deploy traffic policies across organizations. The second section focuses on the use to provide secure communication between Android devices which was proposed by Wang et al. [18].

3.2.1. TEE-based Collaborative Traffic Policy. Since traffic between a company's different sites tends to be forwarded via the same routers and firewalls, there is a possibility for policies to erroneously direct traffic via a public link rather than a preferred private link. In order to alleviate this problem and also allow different companies to collaborate in a manner that does not require them to exchange their policies, Risdianto et al. [17] propose the use of a TEE-based approach. They aim to use it for programmable network switches, as are in use at many larger commercial operations. They propose a way to compile their policies such that the result can be combined with an organization's own rules. Both parties need to input their data into the enclave, which refers to SGX's per-program TEE space, and then the data can be exchanged via an SSL connection. Each connection is only usable once and unidirectional, meaning that party A's enclave transfers its policy to B's enclave, and B's transfers it to A's. Both sides can then compile their policies into rules. During this phase, there are also two steps to check for overlaps between the rules. The first part is the inter-policy check, which checks for exact matches between the rules. The second portion is an intra-policy check, which employs

a binary trie to find overlapping addresses and discards addresses such as 0.0.0.0/0 as they can lead to leaks. The rules are then compiled such that they can be installed on a P4 switch, which is a software-defined network device that can be programmed to perform on all layers of the network stack using the language P4 [19].
The use of a TEE in this case is very interesting as it allows for secure multi-party computation, enabling a shared configuration without exposing your own policies to a less trusted party. According to Risdianto et al. [17], it does not have an impact on the performance of the routers and firewalls, as this process only aims at the compilation of rules rather than the actual operation. It also requires the use of P4 routers. Furthermore, it makes use of the remote attestation mechanism of SGX, as the policy compilation is done in the TEE, which can be used to prove that the same configuration was used by both parties. This has the major advantage of making it simpler to create a communication link between two organizations. It is, however, important to add that several parts of SGX have been proven to be vulnerable to attacks, even the remote attestation, as summarized by Fei et al. [20].

3.2.2. TEE-Based Communication on Android. In the second approach, Wang et al. [18] propose means to have a secure communication channel between Android devices, for which they leverage ARM's TEE called Trustzone. In order to secure communication between parties, each device uses the Elliptic-Curve Diffie-Hellman (ECDH) key agreement to create keys, which are stored inside the TEEs and never leave them, to ensure that they can not be exposed. The system is separated into two parts inside the TEE, namely the key sender and the key receiver. The key sender facilitates the public key generation and the generation of the digital signature that is used to provide integrity proof for the session key negotiation messages. The key receiver is responsible for the public key authentication, the session key generation, and key storage. Both entities run in OP-TEE OS and are involved in sending and receiving operations. Whenever a user wants to send something to another party, they first type the data in the rich environment, then send the data into the TEE to encrypt it using the appropriate keying material, which was negotiated before. Then, the encrypted message is sent to the other party, which can only decrypt the data in the TEE.
It is important to note that in their paper Wang et al. [18] did not actually implement the networked portion. Therefore, their performance analysis is not necessarily correct, but they did analyze the amount of time that their additional measures took and estimated the networking portion. The ECDH key agreement took around 0.563s, from which they assumed the transmission to take place with 128KB/s, which results in a total transmission time of 0.059s. However, this does not need to take place each time two parties want to communicate, as the keys can be kept for a longer time frame. Their method of handling encryption with a TEE also has an impact on performance compared to the standard Android Keystore technology [21], but with an impact of around 11%, it should be manageable as the agreement does not need to be performed each time the two parties communicate. Wang et al. [18] pose that the advantage of this approach is that there is no need for a trusted third party while still allowing two parties to ensure the integrity and confidentiality of their communication. The TEE also decreases the risk of the key being stolen, which can be a problem in Keystore, as it stores the keys in a file. Overall, the performance of an application is worse if a TEE is employed, as it requires additional communication within a system, but it provides more security.

3.3. Usage of TEE in the Cloud

Another interesting use of a TEE is one that seems at the center of the CCC: The usage of TEEs in cloud applications. Since "the cloud" refers to a large distributed system that handles computation to offload tasks from a local device, it is important that they can maintain the confidentiality of their data as well as authenticate that they processed it correctly. The three largest cloud providers, AWS [22], Azure [23], and Google Cloud [24], all offer services that give you access to a TEE. For both Microsoft and Google, who are part of the CCC, this is offered under confidential computing, with Microsoft offering customers access to VMs with Intel SGX and AMD SEV. However, the access to the TDX extension is limited at this point. Google, on the other hand, only offers SEV [25]. Both, however, also offer other types of confidential computing, which offer different pieces to ensure secure operation. Amazon offers its own service that is built on top of Intel and AMD processors, as well as its own architecture, Graviton. Intel TDX itself is also built on top of SGX but is meant to offer capabilities that enable cloud computation. Cheng et al. [26] have offered an overview of the TDX technology. Intel TDX is built for cloud computation, as it runs on top of Intel VT, their virtualization technology. In the work by Geppert et al. [10], they offer multiple use cases for TEEs in the cloud, such as the ability to move data from on-site into the cloud while maintaining protection. Another case is multi-party computation, where the parties need to be able to rely on each other to have computed correctly, which can be proven by the attestation mechanisms offered by TEEs.
Overall, the usefulness of TEEs in cloud applications seems very high, and they must be in use, as the three biggest cloud providers offer them to their customers. However, it is important to note that there is little actual data on the manner in which TEEs are actually used in cloud applications.

4. Conclusion

In this paper, we aimed to give a broad overview of how networking and networked applications are realized in the context of TEEs. We showed different avenues, from directly running the network traffic through the TEE in the case of TrustedGateway to using it as a means to enhance privacy from both a network administration perspective and to secure messages between two Android devices. We also gave an overview of the current state of TEEs in the cloud space, where we showed that they seem to be in demand as the major operators offer services but could not find concrete data on the subject. In summary, there is a lot of interest in the technology, though mostly as a means to increase privacy, for which TEEs are very well suited.

However, there is not a lot of information about how to create applications that use networking directly in the TEE without leveraging the rich environment to perform the Network I/O. Nonetheless, this area is interesting, and the TrustedGateway approach offers a good idea of how that could be done. Since it is a highly specific application, it remains to be seen if the approach can be used in other cases as well. In future research, it would be interesting to see if more applications take an approach similar to the one employed by TrustedGateway and how the usage of TEEs in cloud applications develops.

References

[1] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted Execution Environment: What It is, and What It is Not," in *2015 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 2015.

[2] Intel, "Intel® Software Guard Extensions (Intel® SGX)," 15/11/2023. [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html

[3] ——, "Intel® Trust Domain Extensions (Intel® TDX)," 04/11/2023. [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/overview.html

[4] AMD, "AMD Secure Encrypted Virtualization (SEV)," 15/11/2023. [Online]. Available: https://www.amd.com/en/developer/sev.html

[5] Arm Ltd., "TrustZone for Cortex-A – Arm®," 16/11/2023. [Online]. Available: https://www.arm.com/technologies/trustzone-for-cortex-a

[6] F. Schwarz, "TrustedGateway: TEE-Assisted Routing and Firewall Enforcement Using ARM TrustZone," in *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, ser. RAID '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 56–71.

[7] GlobalPlatform, "Specifications Archive - GlobalPlatform," 15/11/2023. [Online]. Available: https://globalplatform.wpengine.com/specs-library/?filter-committee=tee&utm_source=iseepr&utm_medium=Website&utm_campaign=TEE

[8] B. McGillion, T. Dettenborn, T. Nyman, and N. Asokan, "Open-TEE – An Open Virtual Trusted Execution Environment," in *2015 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 2015.

[9] J. Wang, Y. Cheng, Q. Li, and Y. Jiang, "Interface-Based Side Channel Attack Against Intel SGX." [Online]. Available: https://arxiv.org/pdf/1811.05378.pdf

[10] T. Geppert, S. Deml, D. Sturzenegger, and N. Ebert, "Trusted execution environments: Applications and organizational challenges," *Frontiers in Computer Science*, vol. 4, p. 930741, 2022.

[11] Redmondmag, "Confidential Computing Consortium Formed To Protect Processed Data – Redmondmag.com," 05/12/2023. [Online]. Available: https://redmondmag.com/articles/2019/08/21/confidential-computing-consortium.aspx

[12] Confidential Computing Consortium, "Confidential computing: Hardware-based trusted execution for applications and data v1.3," 11/2022. [Online]. Available: https://confidentialcomputing.io/wp-content/uploads/sites/10/2023/03/CCC-A-Technical-Analysis-of-Confidential-Computing-v1.3_unlocked.pdf

[13] "Members – Confidential Computing Consortium," 15/11/2023. [Online]. Available: https://confidentialcomputing.io/about/members/

[14] GitHub, "rmind/npf: NPF: packet filter with stateful inspection, NAT, IP sets, etc," 05/12/2023. [Online]. Available: https://github.com/rmind/npf

[15] V. Gueant, "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool," 26/02/2024. [Online]. Available: https://iperf.fr

[16] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation," in *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, 2019.

[17] A. C. Risdianto and E.-C. Chang, "TEE-Based Privacy-Preserve in Collaborative Traffic Policy Compilation for Programmable Devices," in *Proceedings of the 2021 ACM International Workshop on Software Defined Networks & Network Function Virtualization Security*, ser. SDN-NFV Sec'21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 19–22.

[18] Y. Wang, W. Gao, X. Hei, I. Mungwarama, and J. Ren, "Independent credible: Secure communication architecture of Android devices based on TrustZone," in *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, 2020, pp. 85–92.

[19] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[20] S. Fei, Z. Yan, W. Ding, and H. Xie, "Security Vulnerabilities of SGX and Countermeasures," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–36, 2022.

[21] Google, "Android Enterprise Security Paper 2023," Google, 2023.

[22] Amazon Web Services, Inc., "Nitro Enclaves," 01/12/2023. [Online]. Available: https://aws.amazon.com/ec2/nitro/nitro-enclaves/

[23] Microsoft, "Azure confidential computing products," 01/12/2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/confidential-computing/overview-azure-products

[24] Google Cloud, "Confidential Computing | Google Cloud," 01/12/2023. [Online]. Available: https://cloud.google.com/confidential-computing

[25] ——, "Confidential Computing concepts," 01/12/2023. [Online]. Available: https://cloud.google.com/confidential-computing/confidential-vm/docs/about-cvm

[26] P.-C. Cheng, W. Ozga, E. Valdez, S. Ahmed, Z. Gu, H. Jamjoom, H. Franke, and J. Bottomley, "Intel TDX Demystified: A Top-Down Approach," 2023.