Towards Shielding 5G Control Plane Functions

Sudip Maitra*, Tolga O. Atalay*, Angelos Stavrou*†, Haining Wang*
*Department of Electrical and Computer Engineering, Virginia Tech, USA
†Kryptowire, LLC, McLean, VA, USA
Email: {smaitra, tolgaoa, angelos, hnw}@vt.edu

***Abstract*—Network Functions Virtualization (NFV) enables flexible and scalable 5G core deployment but it also introduces new attack vectors into the mobile network ecosystem, especially when network functions are deployed on public cloud infrastructure. To address this issue, Third Generation Partnership Project (3GPP) standardization body recommends isolating critical 5G core functionalities inside Hardware Mediated Execution Enclaves (HMEEs). However, the use of HMEEs can incur debilitating QoS degradation in control plane functions including Authentication and Key Agreement (AKA) protocol. In this paper, we design and implement network slices with HMEE-enforced isolation for sensitive AKA functions and characterize their performance. Our findings reveal that the use of HMEE leads to 1.2 to 1.5× increase in function execution time and 2.2 to 2.9× increase in response time for the isolated containers. While appearing very large, this overhead is a small fraction of the end-to-end session setup latency. To evaluate the feasibility of HMEE, we use a real commercial User Equipment (UE) to register with the 5G core network through the isolated AKA functions. Finally, we discuss the role of HMEEs in addressing the key issues introduced by NFV.**

***Index Terms*—5G Core Security, VNF Security, HMEE, SGX**

I. Introduction

In legacy Long Term Evolution (LTE) networks, mobile core network functions were packaged as Physical Network Functions (PNFs) and deployed on proprietary hardware. By contrast, 5G deployments leverage Network Functions Virtualization (NFV) as a key enabler to deliver a flexible and scalable infrastructure. In the context of 3GPP networks, NFV refers to the deployment of softwarized Network Functions (NFs) as Virtual Network Functions (VNFs) on Commercial Off-the-Shelf (COTS) hardware. Service-chaining these VNFs leads to the creation of the network slice [1], a logically isolated network fragment tailored to accommodate a specific set of Quality of Service (QoS) requirements.

While NFV is a powerful approach to enabling a high degree of flexibility in 5G network slicing, it expands the attack surface of the mobile network ecosystem [2]–[4]. Legacy deployments rely on discrete physical devices which implicitly provide security and separation. Consequently, legacy core network security models assume that threats apply only at the edge where the network is exposed to external interfaces. Moreover, unlike PNFs in the legacy core network, VNFs operate in shared virtualization environments. Therefore, the legacy security models cannot account for the new threat vectors introduced by virtualization [5].

As the computational demands of deploying a 5G core network evolve, network operators, especially smaller Virtual Network Operators (VNOs), may increasingly need to utilize cloud infrastructure. This is in addition to larger operators managing their own virtualization infrastructure, while smaller VNOs are already inclined towards third-party hosted solutions. Consequently, cloud service providers such as Microsoft Azure [6] and Amazon Web Services [7] have started partnering with network operators to offer integrated 5G deployments on top of their infrastructure.

The deployment of 5G core VNFs on such infrastructure results in the placement of critical functions next to untrusted third-party applications, which has raised some security concerns [8]. For example, attackers can exploit memory corruption vulnerabilities to escape virtualization boundaries [9], and attackers with local admin privileges can use out-of-bounds write vulnerability to execute code on the host machine [10]. Vulnerabilities in the Linux kernel can be exploited to escalate privilege and execute arbitrary code in the kernel [11] and gain root access [12]. Adversaries in a public cloud can utilize the aforementioned NFV attack vectors to compromise the confidentiality and integrity of the critical 5G core functions.

To tackle these emerging attack vectors in the 5G core network introduced as a result of NFV adoption, 3GPP outlines several Key Issues (KIs) and possible solutions in [5]. One such solution is the use of Hardware Mediated Execution Enclaves (HMEEs). According to the European Telecommunications Standards Institute (ETSI), HMEE is defined as a secure process space hardened against any type of eavesdropping and data alteration attacks from the rest of the system environment [13]. This includes privileged entities such as hypervisors and container engines as well as other kernel-level modules. HMEE-enabled hosts are marked as higher trust domains by 3GPP which are designated for the deployment of critical functions. To that end, Intel Software Guard Extensions (SGX) is a Trusted Execution Environment (TEE) [14] conforming to the fundamental HMEE requirements set forth by 3GPP and ETSI. SGX has garnered much interest in academia [15] and has commercial presences as well [16], [17]. For secure operation in the SGX framework, any code

This material is based on research sponsored by Defense Advanced Research Projects Agency (DARPA) under agreement number HR001120C0155. The views, opinions, and/or findings contained in this article are those of the author(s) and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

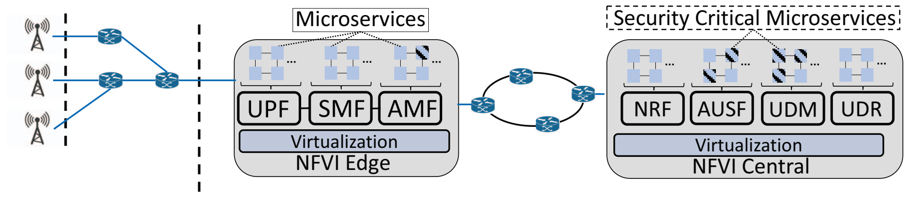


Fig. 1: Deployment of the 5G core in a cloud environment with each VNF comprised of multiple microservices with certain critical functionalities

and data belonging to the application of interest is encrypted by the Central Processing Unit (CPU) and executed inside a protected memory region denoted as an enclave. However, the security guarantees come at the cost of performance. SGX can impose overheads as high as 79% that can ultimately deteriorate the performance of time-sensitive tasks [18], [19].

To the best of our knowledge, no study has investigated the cost of deploying and operating critical 5G core functions inside hardware enclaves. To address this gap, we aim to build and characterize 5G network slices that offer higher security assurances through HMEEs and evaluate the feasibility of such a deployment with real COTS User Equipment (UE). Characterizing the performance of HMEEs and testing their feasibility in the context of 3GPP networks is essential. HMEE is not only offered as one of the solutions by 3GPP to mitigate NFV attack vectors but also a crucial standardized component for realizing trust domains in the future 3GPP networks [5].

In Figure 1, we depict a tentative 5G deployment on top of a cloud infrastructure. The core VNFs comprise multiple microservices to decentralize the physical deployment of the 5G network. Although each VNF has an indispensable role in the network slice service chain, 3GPP recognizes that some functions in the VNFs are more sensitive and require special security assurances [5]. Within this scope, the 5G Authentication and Key Agreement (AKA) cryptography functions, secret key materials, and authentication credentials [20] in the control plane are of the utmost sensitivity. The infiltration of the 5G-AKA protocol would be detrimental to user privacy as unauthorized parties can gain access to sensitive credentials. To characterize the performance of these functions in HMEEs and evaluate the feasibility of 3GPP's proposed solution, we deploy the critical 5G-AKA microservices of the Unified Data Management (UDM), Authentication Server Function (AUSF), and the Access and Mobility Management Function (AMF) in SGX enclaves using Gramine-SGX (formerly Graphene-SGX [21]). We utilize the OpenAirInterface (OAI) 5G core [22], Radio Access Network (RAN) gNB [23], and the gNBSIM RAN entity [24] to create our 5G core network testbed. Our main contributions are summarized below.

- We extract the AKA functions from the three monolithic OAI 5G core VNFs (i.e., UDM, AUSF, and AMF) into three container-based microservices.
- The extracted 5G-AKA microservices are encapsulated inside SGX enclaves to protect them against NFV attack vectors.
- We provide the first comprehensive performance characterization of a real-life network slice implementation with HMEE support by conducting mass experiments with gNBSIM to collect enclave initialization time, operational latency, and SGX-specific metrics to measure the overhead introduced as a result of SGX isolation. Our results show that SGX introduces 2.2 to 2.9× increase in response time across the three isolated containers. We compare this overhead with the end-to-end session setup delay and find that SGX isolation accounts for only 5.58% of the overall session setup delay.
- To evaluate the feasibility of using HMEEs for protecting sensitive 3GPP functions, we conduct an Over-the-Air (OTA) test using a Software-defined Radio (SDR) as the 5G gNB and a OnePlus 8 as the UE. We demonstrate that the UE successfully registers with the 5G core through the isolated AKA functions.
- Finally, we discuss the SGX attributes that meet the HMEE requirements in resolving the key issues highlighted by 3GPP. We also unveil that HMEEs can partially resolve some additional issues, for which 3GPP does not offer any solutions.

The rest of the paper is organized as follows. Section II provides background information on 5G AKA protocol and Intel SGX. The threat model of this work is described in Section III. We detail the modified AKA message flow and deployment of critical functions inside SGX enclaves in Section IV. We present our experimental setup, and evaluation methodology along with our findings in Section V. Section VI provides a discussion regarding the role of HMEE in mitigating NFV attack vectors. Section VII surveys related work, and finally, Section VIII concludes the paper.

II. Background

A. 5G and AKA Overview

The 5G core is comprised of a series of service-chained VNFs as shown in Figure 2. These VNFs communicate over service-based interfaces using Representational State Transfer (REST) APIs standardized by 3GPP according to the Common API Framework (CAPIF). The VNFs are made up of multiple microservices, each one responsible for a specific flow within the control plane. The Unified Data Repository (UDR) serves as the credential storage unit for the users. Fetching credentials from the UDR, the Unified Data Management (UDM) interacts with the Authentication Server Function (AUSF) to generate the authentication vectors of the 5G- Authentication and Key Agreement (AKA) procedure in the Home Network (HN).

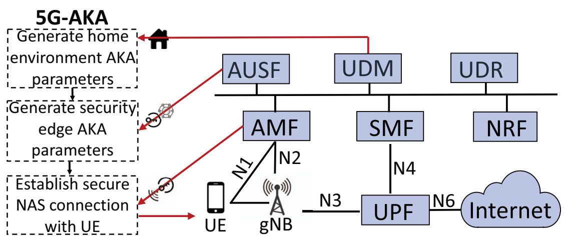

5G-AKA
Generate home environment AKA parameters
Generate security edge AKA parameters
Establish secure NAS connection with UE
AUSF
UDM
UDR
AMF
SMF
NRF
N1
N2
N4
N3
N6
UE
gNB
UPF
Internet

Fig. 2: 5G core service-based-architecture with AKA overview

Moving into the Serving Network (SN), the Access and Mobility Management Function (AMF) establishes a connection with the UE and forwards Non-Access Stratum (NAS) signaling messages between the Access Network (AN) and the core. The Session Management Function (SMF) and the User Plane Function (UPF) constitute the data session anchors for the client. Last but not least, the Network Functions Repository Function (NRF) stores metadata for each VNF and orchestrates mutual discovery procedures between them.

5G AKA. The AKA procedure is crucial in establishing secure communication between the UE and the network. The UE initiates the AKA procedure by sending an authentication request along with its Globally Unique Temporary Identity (GUTI) or Subscription Concealed Identifier (SUCI) to the network. SUCI is the encrypted form of Subscription Permanent Identifier (SUPI). AUSF verifies the SN authentication service authorization and forwards the request to UDM, which then starts the 5G-AKA procedure. If the authentication is successful, the HN issues a GUTI to the UE for the session. In this paper, our goal is to secure the critical microservices of the 5G-AKA procedure summarized in Figure 2 and test the feasibility of such a deployment. When an authentication request from the UE propagates back to the core, UDM generates the Home Environment (HE) Authentication Vector (AV) and sends it to the AUSF. The HE-AV is the concatenation of:

- *RAND:* random number generated by the core network,
- *AUTN:* the authentication token sent from the core to the UE as the mutual authentication challenge,
- *XRES*:* the expected response for the generated authentication challenge,
- *KAUSF:* key shared between UDM and AUSF in the 3GPP key derivation hierarchy [20].

Upon receiving the HE AV, AUSF generates and sends the Security Edge (SE) AV to the AMF, comprised of:

- *RAND:* random number,
- *AUTN:* authentication token,
- *HXRES*:* hashed expected response derived from *XRES**.

When the AMF receives the SE AV, it will issue the challenge to the UE and proceed to establish a secure Non-Access Stratum (NAS) connection.

B. HMEE and SGX

According to ETSI, HMEE refers to *'an area of process space and memory within a system environment within a computer host which delivers confidentiality and integrity of instructions and data associated with that enclave. This enclave is protected from eavesdropping, replay and alteration attacks as the programs within the enclave are executed'* [5], [13]. SGX is a hardware-based TEE developed by Intel to safeguard critical components of an application by utilizing hardware sandboxes called enclaves. TEE is a tamper-resistant processing environment within the CPU, providing hardware-based isolation, code and data integrity, and confidentiality. A third party can also verify the trustworthiness of the secure environment through remote attestation. TEE implementations aim to resist software attacks and physical attacks against the main memory [14]. SGX trusted computing base (TCB) only includes the CPU package and the code running inside the enclave. The attack surface is reduced by considering the host operating system (OS), hypervisor, BIOS, and other system software as untrusted. During boot-up, SGX reserves a portion of the system memory known as Processor Reserved Memory (PRM), which is further divided into two sections: enclave metadata storage and user application data and code storage called Enclave Page Cache (EPC). The contents of the EPC are encrypted and can only be decrypted when they are loaded inside the processor at the Last Level Cache (LLC). This approach prevents any access to the code and data in EPC by software outside the enclave. Since the OS is not part of the SGX TCB, applications running inside the enclave cannot directly issue system calls. Instead, they enter enclaves by invoking an `ECALL` and exit the enclave by `OCALL`. To issue a system call, the application must issue an `OCALL` to exit the enclave and then perform the operation. Once the system call is served by the OS, the application issues an `ECALL` to re-enter the enclave. Before using external data, shielding code inside the enclave performs verification on the fidelity of the results [21], [25]. SGX also has provisions for remote attestation to enable secure deployment of applications on untrusted cloud infrastructure. To summarize, SGX offers confidentiality and integrity of the application code and data by:

- encrypting application code and data when it leaves the CPU package,
- integrity checking the memory contents of an enclave at start time,
- providing remote attestation signed by the hardware,
- restricting control flow to specific entry points during code execution.

Therefore, SGX meets the security attribute requirements of HMEE set by 3GPP and ETSI. However, SGX offers security guarantees at the expense of performance degradation. The SGX design imposes limitations that can result in significant overhead for applications due to tasks such as data and code encryption/decryption, integrity checks, and context switching between enclaves and the untrusted environment [15], [18]. The enclave transitions are especially responsible for the high overheads as each context switch can cost 10,000 to 18,000 cycles [19] which can produce a negative impact on server-based workloads [18], [26]. So, it is critical to assess the usability of such a security mechanism in the context of critical 3GPP network functions.

III. Threat Model

The 5G mobile core network is one of the largest ecosystems to be deployed on top of COTS hardware using virtual components. However, the deployment of 5G core VNFs on public cloud infrastructure results in the placement of critical microservices next to untrusted third-party applications. To understand the security implications in this context, it is critical to identify the relevant actors and attack vectors targeting the critical microservices of sensitive 5G functions. Our threat model is illustrated in Figure 3 highlighting the attack vector of malicious actors to compromise the 5G-AKA flow.

A. Assumptions

In this threat model, side-channel attacks have not been taken into account due to the reasonable assumption that this work employs SGX as an illustrative example of HMEE. Numerous works have shown that competing HMEE solution, AMD SEV [27], is also vulnerable to side-channel attacks [28]–[30]. Indeed, there is no commercial TEE implementation that is completely impervious to side-channel attacks. It is also worth noting that, over time, these vulnerabilities are anticipated to be addressed with architectural improvements and microcode patches. The scope of this work excludes denial-of-service (DoS) attacks, as other works have proposed solutions to address such types of attacks [31]–[33] and incorporating defenses against it would not introduce any original contributions and could potentially divert focus from the primary objectives of this work.

Environment: The deployment environment consists of COTS hardware on the infrastructure shared with third-party application providers. Compute resources are offered in the form of virtual sandboxes where the underlying host is vulnerable to attacks capable of breaking the virtualization boundaries [34]–[38]. This includes malicious co-residents executing attacks such as privilege escalation into infrastructure managing entities and code injections [9]–[12], [39], [40].

Trusted Entities:

- The 5G Radio Access Network (RAN) *gNB* in charge of relaying messages between the core and UE.
- Physical host *Central Processing Unit (CPU)* where critical microservices are loaded into enclaves.
- Gramine in-enclave *bootloader* and *shielding module*.
- *Architectural Enclave Service Manager Daemon* or `aesmd`, provided by Intel's SGX SDK that manages enclaves.

Untrusted Entities:

- The virtualization infrastructure managing entities such as *container engines* and *hypervisors* can be compromised.
- *Third-party applications* deployed next to 5G core VNFs.
- *Host OS, BIOS, off-chip hardware, and other system software* of the COTS server.
- *Other enclaves* running on the same machine.

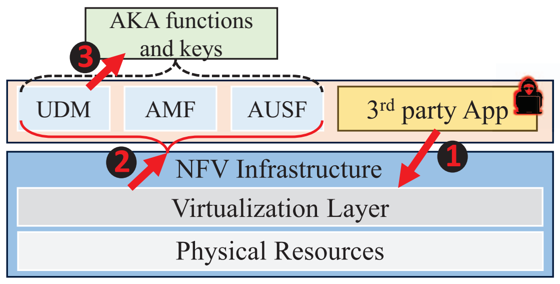


Fig. 3: Attack vector of malicious applications compromising critical microservices of VNFs involved in 5G-AKA

- *Platform Adaptation Layer* or `pal-sgx`, which enables SGX driver to initialize enclaves.

B. Attacker Model

To explain the attacker model, we walk through a hypothetical attack example as shown in Figure 3. The attacker could be a malicious third-party application, which has gained co-residency with the VNO 5G core deployment on a public cloud infrastructure. Previous works have shown that an attacker can achieve co-residency with the target with over 90% success rate [35]. The attack can also originate from a benign application that is compromised by the attacker by exploiting software vulnerabilities. After achieving co-residency, the attacker utilizes a vulnerability in the underlying container engine or VM monitor to gain root privileges or orchestrate a VM escape❶ as shown in [9]–[12]. Once the attacker has compromised the isolation boundaries and gained privilege access, it can move horizontally to other VMs or containers sharing the same virtualization infrastructure❷, thus compromising the confidentiality and integrity of the critical 5G-AKA functions and keys❸.

IV. Testbed Overview

Ensuring that attackers cannot gain control over the cryptographic functions and access sensitive key materials is crucial to securing the 5G core services. Our goal for designing the testbed is to enhance the security of critical functions in the 5G core control plane using SGX and report its performance. This section details the design elements in our testbed, including our approach to isolating AKA functions with SGX and the modified AKA protocol to accommodate the isolated functions.

A. Isolation Overview

To secure the 5G-AKA service chain, we start by identifying the AKA functions in OAI VNFs. These functions were re-engineered as external modules running inside separate Docker containers where each module corresponds to its parent VNF. The testbed overview is illustrated in Figure 4 with the isolated functions summarized in Table I. Each module is implemented as an HTTPs server written in C++17 using OpenSSL and Pistache library. The containers communicate over TLS using Representational State Transfer (REST) APIs

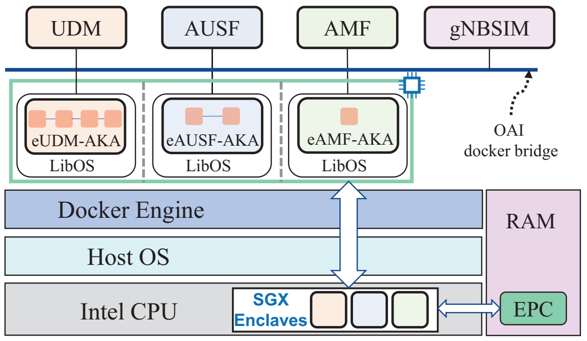


Fig. 4: Testbed design overview

via the OAI Docker bridge. The modules expose REST API endpoints where each AKA function is mapped to an endpoint handler. The monolithic OAI VNFs (UDM, AUSF, AMF) are modified so that during UE registration, the VNFs offload the sensitive functionality to their respective external AKA modules (i.e., eUDM-AKA, eAUSF-AKA, eAMF-AKA) as shown in Figure 4. However, containers do not offer sufficient isolation to thwart attacks described in Section III. Next, these external modules are deployed inside SGX enclaves to safeguard the functions against these attack vectors. While refactoring the VNFs, we notice that some specific protocol libraries (e.g., Stream Control Transmission Protocol) are not supported by the Gramine abstraction layer. Thus, we extract AKA functionalities from the VNFs without including any dependencies for unsupported protocols. Nevertheless, given that the 5G core is comprised of software-based components, porting the critical pieces into SGX enclave does not pose any 5G domain-specific challenges other than working around very specific libraries that are problematic for SGX implementation. Overall, this refactoring results in the secure isolation of the AKA functions, henceforth referred to as **Protected-AKA or P-AKA modules** (i.e., eUDM P-AKA, eAUSF P-AKA, eAMF P-AKA). SGX ensures the confidentiality and integrity of the P-AKA modules by restricting unauthorized access to AKA functions and secrets. The details regarding SGX implementation and deployment are described in Section IV-C.

B. *Modified 5G Message Flow*

The information flow for registering a UE with the core network using P-AKA modules is illustrated in Figure 5. It is important to note that cryptographic parameters are confined to the same physical host. Thus, they are never sent over the network. 3GPP requires that long-term keys used for authentication must remain in the secure environment of the UDM [20]. Therefore, the P-AKA modules are deployed on the same host as the corresponding VNFs. Operators must enforce strict deployment policies to ensure these services are co-located on the same host. Moreover, before any communication between VNFs, 3GPP specifications have provisions for TLS session establishment and mutual authentication [41]. So even within the same host, operators can encrypt the communication between VNFs and P-AKA modules. In fact,

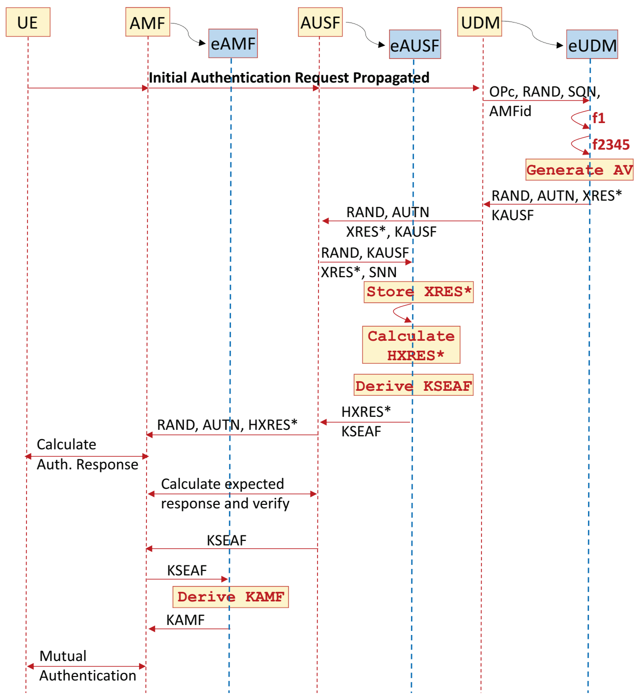


Fig. 5: Modified 5G-AKA message exchange with critical functionalities offloaded into secure external modules deployed inside SGX enclaves

SGX remote attestation and secret sealing can be utilized to achieve these measures as we discussed in Section VI.

The parameters that are sent to and from the modules are summarized in Table I. The *OPc* is the operator key used in the cryptographic algorithms. The *RAND* is a random number generated by the UDM to ensure that the authentication messages for the specific instance of 5G-AKA are unique and up-to-date. The *SQN* is the sequence number incremented in each subsequent authentication message exchanged between the core and the UE, whose purpose is to prevent replay attacks. The *AMFid* is the unique identifier of the AMF. All these parameters are used by the UDM to generate the HE AV, which is a concatenation of the Authentication Token (*AUTN*), Expected Response (*XRES**), *RAND*, and a Message Authentication Code (MAC). When the HE AV is received by the AUSF, the XRES* is hashed to create the Hashed Expected Response (*HXRES**). The *HXRES** is independently calculated by both the UE and core, which forms the foundation of the mutual authentication in 5G-AKA. Comparing the *HXRES** calculated by the UE with its own value, the core can validate the authenticity of the UE. The *KAUSF*, *KSEAF*, and *KAMF* are different sets of keys in the 5G key derivation hierarchy [20], each one responsible for securing different communication links. More details about these parameters and their roles can be found in [20].

The registration of a UE to the core network using the P-AKA modules involves the following steps:

1) UE sends an initial authentication request to AMF,

TABLE I: 5G-AKA functions and parameters loaded into SGX enclaves and the relevant enclave input/output as well as the derivation and/or execution performed inside

P-AKA	Enclave Input		Enclave Output		Derive/
Modules	Param.	Bytes	Param.	Bytes	Execute
eUDM	OPc	16	RAND	16	f1
	RAND	16	XRES*	16	f2345
	SQN	6	KAUSF	32	KAUSF
	AMFid	2	AUTN	16	AUTN
eAUSF	RAND	16	KSEAF	32	KSEAF
	XRES*	16			
	SNN	2	HXRES*	8	HXRES*
	KAUSF	32			
eAMF	KSEAF	32	KAMF	32	KAMF

which is forwarded to AUSF and ultimately to UDM. UDM sends the required parameters (i.e., `OPc, RAND, SQN, AMFid`) to our isolated eUDM P-AKA module, that is running inside an SGX enclave. The eUDM P-AKA module generates the cryptographic parameters `AUTN, XRES*, KAUSF` and sends them to UDM.

2) UDM forwards the generated parameters to AUSF, which in turn calculates the `HXRES*` and derives `KSEAF` within the eAUSF P-AKA module.
3) The AUSF proceeds to send the `RAND`, `AUTN`, and the `HXRES*` to the AMF, triggering the calculation of the authentication response within the UE. After the response is verified by the network, AUSF will forward `KSEAF` to the AMF.
4) AMF sends `KSEAF` to the eAMF P-AKA module where it is used to derive the `KAMF` for securing the NAS signaling with the UE.
5) Finally, AMF mutually authenticates with the UE and registers it with the core.

It is worth noting that a number of these exchanges depicted in Figure 5 could be reduced if the P-AKA modules directly communicated with each other. However, we made a design decision to restrict the communication of P-AKA modules only to their parent VNFs. The reason behind this decision is twofold. Firstly, our goal was to preserve the autonomy of the P-AKA modules, enabling them to be deployed in accordance with specific security requirements and scalability demands. Secondly, we did not want P-AKA modules to alter the regular UE registration flow significantly so that HMEE capability could be seamlessly integrated with OAI's implementation.

C. Deployment in SGX enclave

Applications need to be refactored to deploy inside SGX enclaves because SGX prohibits system calls inside enclaves. However, refactoring existing applications to run inside an SGX enclave requires considerable engineering expertise, time, and effort [42], [43]. AMD SEV [27] on the other hand, provides hardware-isolated VMs without requiring any special changes to the target application. Intel has also developed their version of hardware-isolated VMs called Trust Domain Extensions (TDX) [44] for major cloud service providers.

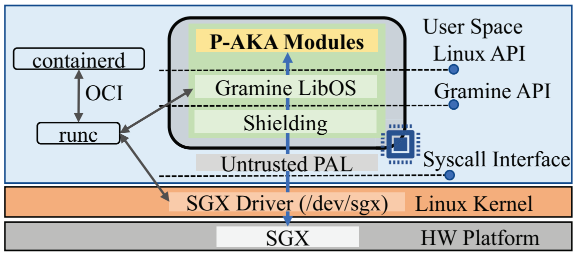


Fig. 6: P-AKA microservices deployed in SGX enclaves using Gramine

Therefore, it is reasonable to assume that vendors will be reluctant to refactor existing software just for SGX. However, secure VMs like SEV and TDX have large TCB and may potentially increase the attack surface, rendering them unsuitable for certain applications [45]. An optimal solution would be to combine the best of both worlds, easing the development and deployment process while keeping the TCB small.

In order to reduce the development costs and effort associated with SGX development, researchers have proposed shim layers and library operating systems (LibOS) [46] that can execute unmodified binaries on SGX, such as GrapheneSGX (now Gramine-SGX) [21], Panoply [47], and SCONE [48]. Introducing another such software layer further increases the TCB and performance overhead. However, it has been shown that a carefully partitioned application using such abstraction layers may not contribute to significant performance costs compared to native SGX port of the application. Thus, the overhead introduced by such intermediate layers is justified by the reduction in development and verification effort [21], [48]. Moreover, one of the design goals for our testbed is to be compatible with different TEE implementations so that one HMEE instance can be easily replaced with another. Furthermore, deploying containers inside the enclave enables us to extend the role of HMEEs in mitigating NFV attack vectors as discussed in Section VI. Hence, we opted to use Gramine to deploy our modules. Out of the available solutions, we chose Gramine because it is open source, faster than Panoply [47], and more feature-rich than the closed source SCONE libc [21].

The detailed overview of the P-AKA module implementation is illustrated in Figure 6. Gramine shielded containers or GSC CLI tool transforms regular Docker images to run inside SGX enclaves using Gramine LibOS. A config and manifest file has to be prepared before using the GSC build tool. The config file contains information regarding the base image, Gramine repository, and SGX driver. The manifest file is a JSON file that specifies configurations of the LibOS and other SGX-related settings and features, dependencies, and trusted files. The GSC signer tool is used to sign the image with a user-provided key.

The P-AKA modules were built using GSC v1.4-1-ga60a499 with preheating enabled (`sgx.preheat_enclave=true`), four allowed threads (`sgx.max_threads=4`), and 512MB of EPC size. To collect SGX-related statistics reported by Gramine, we enabled the `stats` option in the manifest file and built the image with the debug option enabled. Enabling `preheat` option directs Gramine to pre-fault all heap pages during initialization, which results in an initial delay when deploying the server. However, it shifts the cost of EPC page faults to the initialization phase, which is beneficial when a server is expected to start and receive connections after some time [49]. The rationale for selecting the number of threads and EPC size are explained in Section V-B2.

V. Performance Characterization

In this section, we first describe the experimental setup and methodology for characterizing the performance the P-AKA modules. Then we present our findings and analyze the results. Finally, we test the feasibility of P-AKA modules by conducting an OTA test using a real UE.

A. Experimental Setup and Methodology

1) Experimental Setup: For our experiments, we used a Dell PowerEdge R450 server with two SGXv2 capable Intel Xeon Silver 4314 CPUs with 32 physical cores running at 2.40GHz. The server has 512GB of DDR4 RAM, and 16GB of combined EPC size. The experiments are carried out on Ubuntu 20.04 using the 5.15.0-67-generic Linux kernel with the in-kernel SGX driver. We modified the OAI 5G core v1.5.0 VNFs to accommodate P-AKA modules. For the actual deployment, docker-compose version 1.29.2 and containerd runtime version 1.5.11 are used. We utilized *gNBSIM* to establish mass gNB-UE connections with core on a large scale to conduct our experiments. During the OTA proof-of-concept, the COTS OnePlus 8 UE is connected to a Universal Software Radio Peripheral (USRP) x310 acting as the OAI gNB.

2) Methodology: The deployment and metrics collection were carried out using automation scripts for maximum consistency and repeatability. We devised the following experiments for a comprehensive evaluation of P-AKA modules:

1) *Enclave load time:* Measures time for P-AKA modules to become operational. This experiment provides insight into the overhead of the initial deployment but has no bearing on the operational performance of the P-AKA modules.
2) *Number of threads and EPC size:* Determines the optimum number of threads and EPC size required for the proper operation of the P-AKA modules.
3) *Functional and total latency:* Quantifies the latency overheads introduced by SGX compared to container deployment. The functional latency (L_F) measures the execution time of the 5G AKA functions while the total latency (L_T) accounts for the network overhead (L_N) as well (i.e., $L_T = L_F + L_N$).
4) *End-to-end response latency:* Measures the duration from when a request is sent to the P-AKA module (i.e., from the OAI VNF) until the reception of a response.

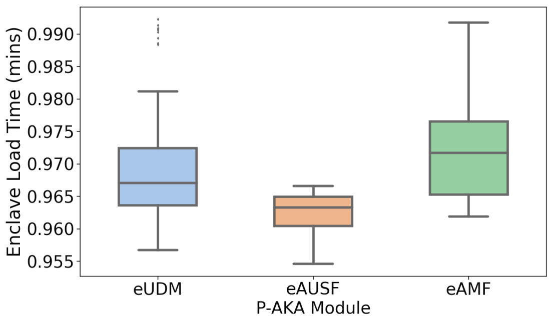


Fig. 7: Enclave load time for the P-AKA modules

5) *SGX-specific metrics:* We register 1 to 10 UEs for 100 iterations and gather SGX-related metrics, including EENTER, EEXIT, and AEX instructions. EENTER and EEXIT are low-level instructions for `ECALL` and `OCALL` system calls. The processor issues asynchronous enclave exit (AEX) instructions to service in-enclave exceptions, faults, and interrupts [25]. We especially focus on EENTER and EEXIT instructions as these incur the highest performance penalty on server-based applications [18]. These metrics can serve as a platform-agnostic basis for comparison with other proposed solutions. We register UEs back to back and measure the number of SGX-related operations. Taking the difference in the number of `ECALLs` and `OCALLs` for consecutive registrations, we get the occurrence of SGX-related operations per UE registration.
6) *OTA test:* Finally, we register a real UE to our testbed through the P-AKA modules (i.e., the isolated AKA functions operating from within the SGX enclaves) to test the feasibility of HMEEs in 3GPP networks.

To ensure the reliability and validity of our collected data, we repeated each experiment 500 times, unless specified otherwise. We noted less than 5% outliers in our measurements.

B. Results and Performance Analysis

1) Slice creation time: For the first experiment, we focus on the enclave load time of the P-AKA modules. When a P-AKA module is first deployed, Gramine and glibc initialize by opening and reading the manifest file, trusted files, and loading shared libraries. The initialization of Gramine and glibc invokes several hundred `OCALLs` which introduce added delay to the enclave deployment. This is especially noticeable when using GSC as it appends the majority of the root directory files (excluding some platform-specific directories e.g., `/boot`, `/dev`, `/etc/mtab`, `/proc`, `/sys`) to the trusted list in the manifest file. This is due to a design decision made by the Gramine team to achieve generality. Moreover, the P-AKA images were built with the pre-heat option enabled, which pre-faults all heap pages during initialization, thereby, further increasing the enclave load time. The enclave load times for the P-AKA modules are presented in Figure 7, where it can be seen that the modules take almost a minute to become

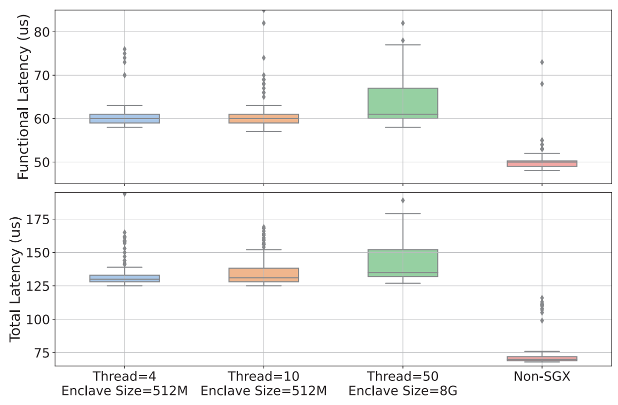


Fig. 8: Effect of varying the number of threads and EPC size on eUDM P-AKA module

operational. However, it is important to note that this delay does not affect the operational performance of the P-AKA modules. This delay only occurs when a new slice is created or migrated to a new host. Therefore, this metric is important to take into account when considering slice creation or migration time. Although enclave load time may not contribute to the latency of the P-AKA services, it is a crucial metric in understanding the deployment characteristics of an HMEE-enabled service. Unlike AKA services, this is particularly important for ephemeral services that are redeployed frequently. The enclave load time of such services may affect the feasibility of using HMEEs for those services.

2) Impact of Threads and EPC size: For the second experiment, we varied the maximum number of threads allowed inside the enclave and the EPC size. Figure 8 presents the functional and total latency overhead as a result of using the eUDM P-AKA module running inside the SGX enclave. The functional latency (L_F) is the amount of time it takes for the UDM-specific 5G AKA function to execute. The total latency (L_T) is the duration between receiving a request from the UDM VNF and sending back a corresponding response. Therefore, L_T accounts for the network overhead in addition to L_F. More details on L_F and L_T are discussed in Section V-B3.

As shown in Figure 8, the number of threads was increased from 4 to 50 and the EPC size was increased from 512MB to 8GB (maximum for a single CPU in our experimental setup). We observed that setting the thread count below 4 and EPC size below 512MB causes the P-AKA modules to behave inconsistently. This is because Gramine uses 3 helper threads to facilitate inter-process communication, implement timer and asynchronous events, and the TLS handshake for new pipe creation. Therefore, a minimum of four threads is required for the P-AKA modules to perform consistently.

In this experiment, P-AKA modules are single-threaded applications and we register one UE at a time. Increasing the number of concurrent clients without impacting the performance of the modules would require changing the maximum allowed number of threads. Given the single-threaded P-AKA

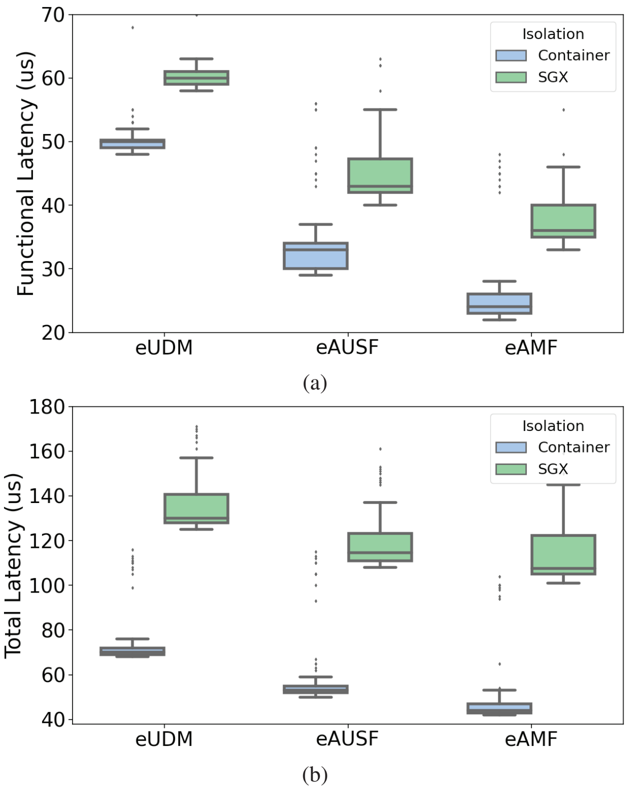


Fig. 9: (a) Functional (L_F) and (b) Total (L_T) latency of the P-AKA modules

servers, results in Figure 8 show that increasing the number of threads and EPC size beyond 4 and 512MB respectively, does not improve the performance of the module for a given client. In this case, the maximum number of threads denotes the number of threads the enclave is allowed to spawn at a time. It is not surprising that the performance did not improve by increasing the number of threads as the server would only spawn new threads in response to more UE registration flows.

The EPC size depends on the application, the shared libraries, and other dependencies. Increasing the EPC size from 512MB to 2GB does not have any effect on the performance of the modules. However, increasing the EPC size to 8GB results in a slight decrease in performance and a wider interquartile range, suggesting greater variability compared to smaller EPC sizes. This is due to the increase in paging which is the process of moving contents between EPC and main memory [25], [50]. Identical behavior was observed in the eAUSF P-AKA and eAMF P-AKA modules in response to increasing the thread count and EPC size. Based on these observations, we configured the enclaves with 4 threads and 512MB of EPC for the remaining experiments.

3) Overheads introduced by SGX: We observe that the difference in latency between non-SGX container deployment and monolithic deployment is negligible. Therefore, we focus on the functional (L_F) and total (L_T) latency of the external

TABLE II: SGX overhead across the isolated modules

Module	L_F	L_T	$R^C \rightarrow R_S^{SGX}$	$R_S^{SGX} \rightarrow R_I^{SGX}$
eUDM	1.2	1.86	2.2	19.04
eAUSF	1.3	2.15	2.5	18.37
eAMF	1.5	2.43	2.9	21.42

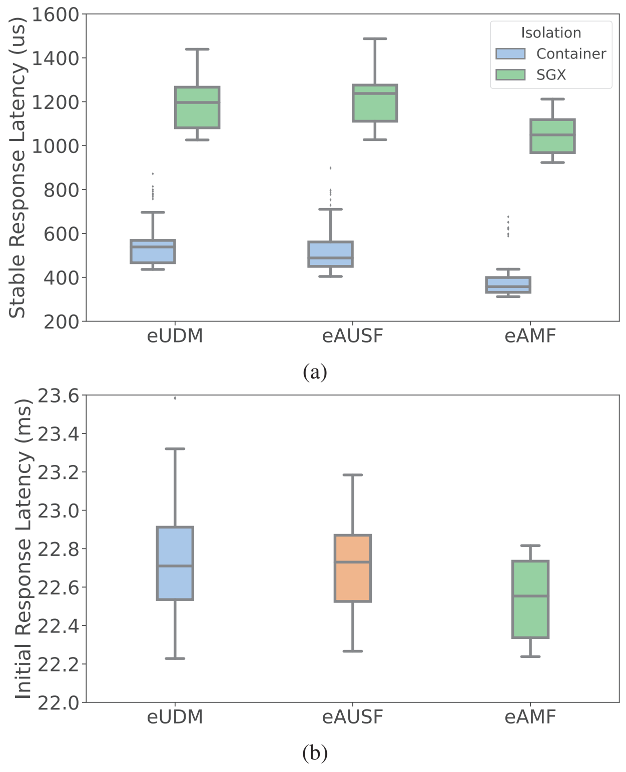


Fig. 10: (a) Stable (R_S^{SGX}) and (b) Initial (R_I^{SGX}) response time of the external P-AKA modules

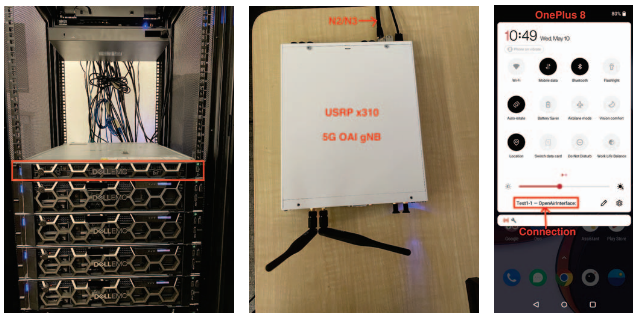
modules and compare them to the non-SGX container deployments of the same modules. Figure 9 shows that SGX incurs a 1.2× overhead for L_F and around 1.86× overhead for L_T, relative to the unprotected eUDM-AKA module. The overheads for the other modules are summarized in Table II. It is worth noting that the eUDM P-AKA module exchanges the highest number of bytes (described in Table I) among the three and therefore, introduces the highest latency followed by eAUSF and eAMF P-AKA modules. The table shows that the overhead for L_F across all three modules is within 1.2 to 1.5 times compared to the unprotected deployment, while the overhead for L_T is higher and more variable. This is because L_T depends on the size of data sent and received by the P-AKA modules. Furthermore, the network overhead is much more pronounced in SGX than local computations. Network I/O operations introduce overheads due to processing required for data encryption and decryption. They involve untrusted operations and therefore trigger `OCALLs` and `ECALLs` to transfer data from the untrusted environment to the enclave and vice versa. Last but not least, the Pistache HTTP server uses `epoll_wait` system calls to monitor sockets for incoming requests, incurring more overhead.

TABLE III: SGX specific operational statistics for the external P-AKA modules

P-AKA Modules	# of UEs	EENTERs	EEXITs	AEXs
eUDM	3	1689	1595	140370
	2	1599	1505	140371
	1	1508	1414	140320
eAUSF	3	1721	1627	140496
	2	1630	1536	140336
	1	1539	1445	140380
eAMF	3	1720	1626	140630
	2	1629	1535	140426
	1	1537	1443	140354
Empty workload	-	762	680	49674

4) Response Time of the P-AKA Modules: In this experiment, we focus on the response time of the P-AKA modules from the VNF perspective. Stable response latency (R_S^{SGX}) is compared with the response latency of unprotected container deployments of the 5G-AKA modules (R^C). The overheads are summarized in Table II, where it can be seen that the R_S^{SGX} varies from 2.2× to 2.9× across the P-AKA modules when compared to R^C. Although the overheads appear large, we measure that end-to-end UE session setup time in our testbed is around 62.38 ms. The cumulative delay added due to SGX isolation is around 3.48 ms, which is only 5.58% of the overall session setup latency. Figure 10 shows the stable (R_S^{SGX}) and initial response latency (R_I^{SGX}) of the P-AKA modules. As seen in Table II, R_I^{SGX} is around 20× that of the R_S^{SGX}, because when a module is first deployed in an enclave, the initial request coming from a VNF invokes several `OCALLs` and `ECALLs` to load drivers and other network stack dependencies. Once these have been cached, subsequent requests are served faster.

5) SGX specific metrics: In this experiment, we analyze the SGX-specific metrics reported by the P-AKA modules. As described in Section V-A, we registered one to ten UEs (each 100 times) and recorded the number of EENTER and EEXIT instructions. For the sake of brevity, a sample of up to three UE registration statistics is shown in Table III, where we can see the total number of EENTERs and EEXITs reported by P-AKA modules. The majority of these are due to `OCALLs` invoking an EEXIT instruction to exit the enclave and EENTER instruction to re-enter the enclave. We observe that the number of AEX instructions is not affected by the number of UE registrations, while the total number of EENTERs is higher than EEXITs. This is the result of `ECALL` handling by Gramine as it performs a single `ECALL` for the whole process and one per new thread [21]. It is worth noting that if an application exits the enclave through AEX instruction, it does not re-enter the enclave using the EENTER but the ERESUME instruction. Therefore, AEX instructions do not contribute to the number of EENTERs. Taking the difference of subsequent registrations of up to ten UEs, we observe that the number of EENTERs and EEXITs for registering one UE is around 90. We were able to confirm that the local computation of



(a) 5G core server (b) OAI gNB (c) Connection

Fig. 11: OTA testbed with USRP x310 the OAI gNB and a OnePlus 8 UE

TABLE IV: Hardware and Software Used for Testbed; Mobile Country Code (MCC), Mobile Network Code (MNC), Physical Resource Blocks (PRBs)

Server	**Configuration**
2 x Intel Xeon Silver 4314 CPUs 512GB DDR4 RAM - 16GB EPC Ubuntu 20.04 OS 5.15.0-67-generic kernel	MCC - 001 MNC - 01 PRBs - 106 Frequency - 3.6192 GHz
gNB Radio Unit	**COTS UE**
USRP x310 [51] OAI develop branch	OnePlus 8 - Android 11 Oxygen 11.0.11.11.IN21DA

the 5G AKA functions does not contribute to the number of `OCALLs` and `ECALLs` and these calls are only invoked during network I/O operations. Therefore, we see a similar number of EENTERs and EEXITs across all three modules. This implies that the performance of the P-AKA modules can be improved by optimizing network operations. Table III also shows the SGX-specific statistics for an empty workload, which can be considered the cost of using GSC. Subtracting the cost of an empty workload, we observe that deploying the Pistache server inside an SGX enclave contributes to around 650 EENTER and EEXIT instructions.

6) HMEE feasibility test with OTA: To collect data at a large scale for the performance characterization of the isolated modules, our primary experimental setup described in Section V-A relied on the gNBSIM entity to simulate the RAN. In the OTA experiment, we construct the testbed illustrated in Figure 11 using an SDR serving as the OAI gNB and a OnePlus 8 device as the UE.

We use the same Dell PowerEdge R450 described in Section V-A and shown in Figure 11a to host OAI 5G core. Connected to this server is the Universal Software Radio Peripheral (USRP) x310 in Figure 11b acting as the OAI gNB. An OpenCells SIM card is programmed to the test Public Land Mobile Network (PLMN) 00101 to ensure that the COTS UE can detect the OAI as an operator. We observed that if custom mobile country or network codes were used, the device would be unable to detect the OAI gNB. Furthermore, the specific Android Oxygen OS version in Table IV had to be installed onto the OnePlus 8 for a successful end-to-end connection. For other commercial UE models, different OS versions may be required. Despite the overheads introduced by the use of HMEE, the OnePlus 8 COTS mobile phone in Figure 11c successfully establishes a data session with the gNB after registering with 5G core network utilizing P-AKA modules, resulting in the *Test1-1 – OpenAirInterface* connection. The relevant software and hardware configuration information for successfully connecting this phone to the OTA testbed is summarized in Table IV.

7) Optimizations and future of SGX in securing network services: While the overhead introduced by SGX is acceptable, there are several potential optimizations that can further improve the performance.

- As noted in [52], much of the overhead introduced by SGX can be reduced by optimizing SGX-specific libraries.
- System call overhead can be reduced by including user-level TCP stack (mTCP) [53] in the enclave, and DPDK [54] can improve packet processing performance as shown in [55]. While using a user-level socket API pulls in more functionality inside the enclave, which further increases the TCB, we recognize its potential benefits in improving the performance of P-AKA modules.
- Gramine supports exitless feature that offloads OCALL execution to an untrusted helper thread that performs system calls on behalf of the enclave thread, thereby improving OCALL performance [56], [57]. However, if this feature is enabled, the helper threads are occupied listening for OCALL requests from the enclave thread. Moreover, this feature is insecure for production usage as of now.
- Since our design is microservice-based, it inherently supports horizontal scaling. Therefore, network operators can scale the enclave worker nodes and SGX-capable host pools on demand to secure more services.

Intel continues to support SGX in their latest 5th Gen Intel Xeon Scalable Processors [58]. Intel has partnered with Fortanix to ease the deployment of services into SGX for network operators [59], [60]. As of Release 18, the standardized definition of sensitive functions and sub-functions within a virtualized environment is a work in progress [5], [20]. 3GPP recommends deploying services in appropriate trust domains based on the sensitivity of workloads, risk appetite, and use case. The physical hosts are categorized into trust domains based on the security features of a host [5]. Once the definition of sensitive services is standardized in future releases, we expect SGX and HMEEs as a whole, to play a crucial role in securing future generations of cellular networks.

VI. Discussion

3GPP outlines several Key Issues (KIs) arising from NFV and possible solutions in [5]. The KIs relevant to this work are depicted in Table V and the KI identifiers and descriptions are taken directly from [5]. 3GPP recommends HMEE as a

TABLE V: Key Issues Summary (●: HMEE applicable KIs identified by 3GPP; ✦: full and ❒: partial solutions)

KI #	Description	Solution
2	Confidentiality of sensitive data	✦
5	Data location and lifecycle	❒
6	Function isolation	●
7	Memory introspection	●
11	Where are my keys and confidential data	❒
12	Where is my function	❒
13	Attestation at 3GPP function level	✦
15	Encrypted data processing	●
20	3rd party hosting environments	❒
21	VM and hypervisor breakout	❒
25	Container security	●
26	Container breakout	❒
27	Secrets in NF container images	✦

solution for **KIs 6, 7, 15, and 25** (marked with ● in the table). However, we believe HMEE has the potential to have a bigger impact in solving NFV attack vectors. In fact, we argue that HMEE can play a crucial role in resolving **KI 20, 21, 26, and 27**, for which 3GPP offers no solution. To that end, we identified nine additional KIs that can be either fully (marked with ✦) or partially (marked with ❒) mitigated with HMEE. Some of the KIs are marked as partially solved with HMEE because the full mitigation strategy for these KIs are contingent on additional security requirements that are out of scope. In this section, we briefly explain the SGX attributes that meet the HMEE requirements in mitigating the KIs identified by 3GPP. We also present our arguments to explain why the HMEE-based solution is applicable for nine KIs, in addition to the four noted by 3GPP.

KI 6. When NFs share resources (e.g., hypervisor, compute, and memory) secure protocols are reduced to protecting information traversing different memory locations in a single memory block [5]. To thwart such attack vectors, 3GPP recommends providing confidentiality protection for information traveling between different memory locations. In SGX framework, the data and code stored in the memory are encrypted, thereby providing confidentiality.

KI 7. An attacker who has access to the hypervisor or container management engine, may inspect and manipulate the memory of other functions. **KI 15** is an extension to KI 7, where sensitive key material could be stolen by an attacker through memory introspection in an insecure environment. 3GPP requires that data-in-use should be inaccessible by other VNFs and the virtualization layer, and the sensitive functions should be executed in HMEE. With SGX, the hypervisor or other co-residents cannot access EPC as it is encrypted and only decrypted inside LLC of the CPU. Therefore, the attacker with privileged access cannot gain any sensitive information by inspecting the memory of functions protected by SGX.

KI 25. NFV implementations are recently moving towards container-based architecture due to swifter deployment and resource efficiency compared to VMs. However, containers do not provide the same level of isolation that VMs offer. With SGX and Gramine, we were able to deploy the containerized AKA functions in the SGX enclave, which provides hardware-based isolation for the containers.

KI 2. NFV introduces new attack vectors to the 5G core network and a level of assurance is required to store sensitive cryptographic keys in the virtualized environments [61]–[63]. The security-critical data could be stolen by an attacker with access to the virtualization layer. 3GPP requires the increased assurance that sensitive information should not be exposed through the virtualization layers. This requirement can be met by processing sensitive information in an HMEE-enabled host infrastructure.

KI 5. One of the issues described in this KI stems from the fact that virtual functions can be moved to other hosts for efficient resource utilization. However, privacy-sensitive data may be exposed when storage resources are reused in a cloud environment. 3GPP requires all privacy-sensitive data to be encrypted at rest and in transit and the resources used by a VNF to be cleared when it is moved or terminated. SGX partially mitigates KI 5 as it ensures the sensitive cryptographic materials are encrypted when they leave the CPU package, and the cache should be flushed once the enclave is torn down.

KI 11. VM or container's view of the physical resources are abstracted by the hypervisor or the container engine. Certain NFs may require tamper-proof hardware key-storage and an attacker may present a virtual key-storage instance to the NF and compromise cryptographic keys. 3GPP requires a mechanism to ensure that VNFs can trust the security provisions offered by the host environment. **KI 12.** NFs might be instantiated or migrated to less secure hosts. To address this issue, 3GPP requires that the deployment of NFs should be preceded by a validation process utilizing secure hardware-backed attestation to verify the security posture of the hosting environment. **KI 13.** Without attestation rooted in hardware, VNFs have limited means of verifying the trustworthiness of other VNFs or the underlying infrastructure.

Hardware-based remote attestation can partially (i.e., KI 11, 12) or fully (i.e., KI 13) address these issues. SGX remote attestation can be used to verify the authenticity and integrity of a remote enclave. Virtual security resources (e.g., key storage) can be implemented inside SGX enclave and NFs requiring specific hardware security resources can verify the attestation report before deployment. SGX can also be used to generate and verify attestation reports that span from the hardware to the 3GPP function level.

KI 20. Sensitive information can be leaked if VNFs are hosted by a third party. Smaller VNOs do not possess the financial backing to build their own network infrastructure and will instead use 3rd party hosting environments for deploying their services. However, since the infrastructure is shared with other applications, sensitive information may be compromised by malicious co-tenants. To partially resolve this issue, 3GPP requires the infrastructure operator to provide confidentiality to the sensitive information of the virtualized NFs. As discussed earlier, HMEE can provide a secure enclave for confidentiality, which can be verified using attestation reports.

KI 21 and 26. Integrity of a virtual NF can be compromised by an attacker orchestrating Hyperjacking [63], VM or container breakout [34], [39], [40]. Although HMEE can not prevent these exploits, it can minimize the impact of such a breach. These exploits enable attackers to traverse the virtualization layer and steal sensitive information or manipulate the functions. The use of HMEE can prevent attackers from doing so because the code and data in an enclave are confidentiality and integrity protected.

KI 27. Container or VM images may contain credentials that are used for authentication purposes to create secure communication channels between NFs. However, attackers can gain copies of these images and extract or manipulate the secrets, thereby compromising the trustworthiness of the credentials. 3GPP requires the critical information in VNF images to be properly protected. The secrets in VNF images can be securely stored with the use of secret-sealing and attestation mechanisms offered by SGX [64]. Instead of storing plaintext secrets in the image, an encrypted secret can be provisioned to the NF image, which can only be unsealed when the enclave environment can be verified with an attestation report.

In Section III, we outline a potential attack scenario where an attacker gains co-residency with the VNFs involved in the 5G-AKA procedure. The attacker can gain root privileges and escape the virtualization boundary by exploiting vulnerabilities in the underlying software and infrastructure. The attacker then can exfiltrate sensitive user credentials (e.g., subscriber SUPI) or tamper the AKA primitives, thereby compromising the communication between the UE and the 5G network. SGX offers hardware-based isolation mechanisms such that any entity other than the CPU cannot access or manipulate the data and code residing in an enclave. Given that the AKA services are deployed in SGX enclaves as discussed in IV-C, the attacker cannot access or tamper the cryptographic parameters or functions, thus preventing such an attack scenario.

In addition to the KIs discussed above, HMEE can play a crucial role in realizing trust domains in the future 3GPP networks. Although nearly all 5G NFs contain sensitive information, some of them require additional security assurances. HMEE-enabled NFVIs offer superior security assurances and can be utilized to establish higher trust domains for the deployment of sensitive NFs. The relevance of HMEE in establishing trust domains is exemplified by the fact that 3GPP assesses the trustworthiness of an NFVI based on its HMEE capabilities.

VII. Related Work

A multiple mobile network operator (MNO) cooperation scheme using SGX was proposed in [65]. An implementation was presented to evaluate the proposed scheme, where UE belonging to one operator is registered to another MNO while preserving user privacy. The reported overhead introduced by SGX for UE registration varied from 5 to 10×. To the best of our knowledge, this is the only study that employed an SGX-based scheme to protect 5G services. However, the focus of this work is on establishing a trusted and secure communication channel between collaborating MNOs, while our focus is to secure the sensitive microservices within the 5G core VNFs. In [66], the authors discussed using TEEs to securely offload critical 5G services to third-party infrastructures. They compared several isolation mechanisms and proposed SCONE as a TEE-as-a-Service (TEEaaS) solution for securing 5G services. However, the framework was not implemented or assessed in terms of performance. SGX remote attestation services can be utilized to verify the integrity of the P-AKA modules before deployment as demonstrated in [67], [68], which address the issue of key provisioning and TLS session establishment.

The majority of the relevant studies in hardware-based isolation are generic frameworks for securing network functions. SafeBricks [69] presents a system to protect network functions in untrusted cloud environments by utilizing SGX. To avoid inter-enclave communication penalties inherent in VNF service-chaining, Safe-Bricks deploys multiple VNFs in the same enclave while maintaining isolation. LightBox [70] and ShieldBox [71] pursue a similar premise and present frameworks for deploying network traffic processing VNFs using SGX. SafeLib [55] provides a framework for adopting stateful VNFs to be securely outsourced to third-party service providers using SGX. The proposed solution adapts libVNF [72] to ease VNF development and Gramine to avoid expensive enclave transitions. However, it is challenging to measure the effectiveness of their solution, due to the limited scope of their evaluation strategy. Trusted Click [52] presents an architecture for integrating SGX into NFV application models to enhance application data privacy. The solution is evaluated by extending Click, a modular router [73], to perform secure packet processing inside an SGX enclave. Another similar solution is presented in [31] to protect VNFs from DDoS attacks by deploying elements of Click inside SGX enclaves.

VIII. Conclusion

NFV has enabled flexible and efficient deployment of 5G network functions on COTS hardware. But deployment of VNFs on cloud infrastructure exposes them to various co-tenancy attacks. Although HMEE can offer confidentiality and integrity protection to sensitive 3GPP functions and secrets, it can impose significant performance penalties which may degrade the QoS of sensitive control plane functions. We characterize the performance of HMEE isolated AKA functions and observe 1.2 to 1.5× overhead for function execution time and 2.2 to 2.9× overhead in overall response latency. However, the overhead introduced due to HMEE accounts for only 5.58% of the overall UE session setup delay. Despite the overheads, we successfully register a real UE to the 5G core network through the isolated AKA functions. While discussing the role of HMEE as highlighted by 3GPP, we discover that HMEE as a solution can have a bigger impact in mitigating the key issues of NFV.

References

[1] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[2] R. Khan, P. Kumar, D. N. K. Jayakody, and M. Liyanage, "A Survey on Security and Privacy of 5G Technologies: Potential Solutions, Recent Advancements, and Future Directions," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 196–248, 2020.

[3] J. H. Park, S. Rathore, S. K. Singh, M. M. Salim, A. Azzaoui, T. W. Kim, Y. Pan, and J. H. Park, "A Comprehensive Survey on Core Technologies and Services for 5G Security: Taxonomies, Issues, and Solutions," *Hum.-Cent. Comput. Inf. Sci*, 2021.

[4] S. Lal, T. Taleb, and A. Dutta, "NFV: Security Threats and Best Practices," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 211–217, 2017.

[5] 3GPP, "Study on Security Impacts of Virtualisation (Release 18)," 3rd Generation Partnership Project (3GPP), TR 33.848 V18.0.0, Sep. 2023.

[6] Microsoft, "Azure Private 5G Core – Manage 5G Networks — Microsoft Azure," https://azure.microsoft.com/en-us/products/private-5g-core, 2023, (accessed November 30, 2023).

[7] AWS, "Wavelength Zone Locations," https://aws.amazon.com/wavelength/locations/, 2023, (accessed November 30, 2023).

[8] Y.-L. Huang, B. Chen, M.-W. Shih, and C.-Y. Lai, "Security Impacts of Virtualization on a Network Testbed," in *2012 IEEE Sixth International Conference on Software Security and Reliability*. IEEE, 2012, pp. 71–77.

[9] "CVE-2022-31696," https://nvd.nist.gov/vuln/detail/CVE-2022-31696, August 2022, (accessed November 30, 2023).

[10] "CVE-2022-31705," https://nvd.nist.gov/vuln/detail/CVE-2022-31705, December 2022, (accessed November 30, 2023).

[11] "CVE-2021-31440," https://nvd.nist.gov/vuln/detail/cve-2021-31440, May 2021, (accessed November 30, 2023).

[12] "CVE-2020-14386," https://nvd.nist.gov/vuln/detail/CVE-2020-14386, September 2020, (accessed November 30, 2023).

[13] ETSI, "Network Functions Virtualisation (NFV);NFV Security;Report on use cases and technical approaches for multi-layer host administration," ETSI, GS NFV-SEC 009 V1.1.1, Dec. 2015.

[14] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted Execution Environment: What it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/Ispa*, vol. 1. IEEE, 2015, pp. 57–64.

[15] N. C. Will and C. A. Maziero, "Intel software guard extensions applications: A survey," *ACM Computing Surveys*, 2023.

[16] M. Russinovich, "Azure Confidential Computing," https://azure.microsoft.com/en-us/blog/azure-confidential-computing/, May 2018, (accessed November 30, 2023).

[17] Fortanix. Intel and Fortanix Confidential Computing Manager - Joint Solution Brief. [Online]. Available: https://resources.fortanix.com/intel-and-fortanix-confidential-computing-manager-joint-solution-brief

[18] O. Weisse, V. Bertacco, and T. Austin, "Regaining lost cycles with HotCalls: A fast interface for SGX secure enclaves," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2, pp. 81–93, 2017.

[19] T. Dinh Ngoc, B. Bui, S. Bitchebe, A. Tchana, V. Schiavoni, P. Felber, and D. Hagimont, "Everything You Should Know about Intel SGX Performance on Virtualized Systems," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 1, pp. 1–21, 2019.

[20] 3GPP, "Security Architecture and Procedures for 5G System," 3rd Generation Partnership Project (3GPP), TS 33.501 V18.1.0 , Mar. 2023.

[21] C.-C. Tsai, D. E. Porter, and M. Vij, "Graphene-sgx: A Practical Library OS for Unmodified Applications on SGX." in *USENIX Annual Technical Conference*, 2017, pp. 645–658.

[22] OAI, "5G Core Network – OpenAirInterface," https://openairinterface.org/oai-5g-core-network-project/, 2023, (accessed November 30, 2023).

[23] ——, "5G RAN - OpenAirInterface5g," https://gitlab.eurecom.fr/oai/openairinterface5g, (accessed November 30, 2023).

[24] Rohan, "Rohan - gnbsim - gitlab," https://gitlab.eurecom.fr/kharade/gnbsim, 2023, (accessed November 30, 2023).

[25] V. Costan and S. Devadas, "Intel SGX Explained," *Cryptology ePrint Archive*, 2016.

[26] P.-L. Aublin, F. Kelbert, D. O'Keffe, D. Muthukumaran, C. Priebe, J. Lind, R. Krahn, C. Fetzer, D. Eyers, and P. Pietzuch, "TaLoS: Secure and transparent TLS termination inside SGX enclaves," Imperial College London, Tech. Rep., 2017.

[27] AMD, "AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More," *White Paper, January*, vol. 53, pp. 1450–1465, 2020.

[28] M. Li, Y. Zhang, H. Wang, K. Li, and Y. Cheng, "{CIPHERLEAKS}: Breaking constant-time cryptography on {AMD}{SEV} via the ciphertext side channel," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 717–732.

[29] M. Li, L. Wilke, J. Wichelmann, T. Eisenbarth, R. Teodorescu, and Y. Zhang, "A Systematic Look at Ciphertext Side Channels on AMD SEV-SNP," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 337–351.

[30] P. Jauernig, A.-R. Sadeghi, and E. Stapf, "Trusted Execution Environments: Properties, Applications, and Challenges," *IEEE Security & Privacy*, vol. 18, no. 2, pp. 56–60, 2020.

[31] J. Wang, S. Hao, Y. Li, C. Fan, J. Wang, L. Han, Z. Hong, and H. Hu, "Challenges Towards Protecting VNF with SGX," in *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2018, pp. 39–42.

[32] 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.

[33] D. Gong, M. Tran, S. Shinde, H. Jin, V. Sekar, P. Saxena, and M. S. Kang, "Practical verifiable in-network filtering for ddos defense," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1161–1174.

[34] H. Zhao, Y. Zhang, K. Yang, and T. Kim, "Breaking Turtles All the Way Down: An Exploitation Chain to Break out of {VMware}{ESXi}," in *13th USENIX Workshop on Offensive Technologies (WOOT 19)*, 2019.

[35] S. Shringarputale, P. McDaniel, K. Butler, and T. La Porta, "Co-residency Attacks on Containers are Real," in *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, 2020, pp. 53–66.

[36] A. O. F. Atya, Z. Qian, S. V. Krishnamurthy, T. La Porta, P. McDaniel, and L. Marvel, "Malicious co-residency on the cloud: Attacks and defense," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[37] X. Gao, B. Steenkamer, Z. Gu, M. Kayaalp, D. Pendarakis, and H. Wang, "A Study on the Security Implications of Information Leakages in Container Clouds," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 174–191, 2021.

[38] Z. Wang, R. Yang, X. Fu, X. Du, and B. Luo, "A Shared Memory based Cross-VM Side Channel Attacks in Iaas Cloud," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2016, pp. 181–186.

[39] "CVE-2016-3134," https://nvd.nist.gov/vuln/detail/CVE-2014-4699, April 2016, (accessed November 30, 2023).

[40] "CVE-2014-4699," https://nvd.nist.gov/vuln/detail/CVE-2014-4699, July 2014, (accessed November 30, 2023).

[41] 3GPP, "Network Domain Security (NDS); IP network layer security," 3rd Generation Partnership Project (3GPP), TS 33.210 V17.1.0 , Sep. 2022.

[42] A. Hasan, R. Riley, and D. Ponomarev, "Port or Shim? Stress Testing Application Performance on Intel SGX," in *2020 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 2020, pp. 123–133.

[43] J. Han, S. Kim, J. Ha, and D. Han, "SGX-Box: Enabling Visibility on Encrypted Traffic using a Secure Middlebox Module," in *Proceedings of the First Asia-Pacific Workshop on Networking*, 2017, pp. 99–105.

[44] Intel, "Intel Trust Domain Extensions," Intel, White Paper, Feb. 2023.

[45] S. Mofrad, F. Zhang, S. Lu, and W. Shi, "A Comparison Study of Intel SGX and AMD Memory Encryption Technology," in *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, 2018, pp. 1–8.

[46] A. Baumann, M. Peinado, and G. Hunt, "Shielding Applications from an Untrusted Cloud with Haven," *ACM Transactions on Computer Systems (TOCS)*, pp. 1–26, 2015.

[47] S. Shinde, D. Le Tien, S. Tople, and P. Saxena, "Panoply: Low-TCB Linux Applications with SGX Enclaves." in *NDSS*, 2017.

[48] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. L. Stillwell, D. Goltzsche, D. Eyers, R. Kapitza, P. Pietzuch, and C. Fetzer, "SCONE: Secure Linux Containers with Intel SGX," in *12th USENIX Symposium on*

Operating Systems Design and Implementation (OSDI 16). Savannah, GA: USENIX Association, Nov. 2016, pp. 689–703.

[49] Gramine, "Manifest syntax - Gramine documentation," https://gramine.readthedocs.io/en/stable/manifest-syntax.html, (accessed November 30, 2023).

[50] N. Weichbrodt, P.-L. Aublin, and R. Kapitza, "sgx-perf: A Performance Analysis Tool for Intel SGX Enclaves," in *Proceedings of the 19th International Middleware Conference*, 2018, pp. 201–213.

[51] "OAI - OpenAirInterface5G," https://gitlab.eurecom.fr/oai/openairinterface5g, 2023, (accessed November 30, 2023).

[52] M. Coughlin, E. Keller, and E. Wustrow, "Trusted Click: Overcoming Security issues of NFV in the Cloud," in *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2017, pp. 31–36.

[53] E. Jeong, S. Wood, M. Jamshed, H. Jeong, S. Ihm, D. Han, and K. Park, "mTCP: a Highly Scalable User-level TCP Stack for Multicore Systems," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, 2014, pp. 489–502.

[54] "DPDK - Data Plane Development Kit," https://www.dpdk.org/, 2023, (accessed November 30, 2023).

[55] E. Marku, G. Biczók, and C. Boyd, "SafeLib: A Practical Library for Outsourcing Stateful Network Functions Securely," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, 2021, pp. 244–252.

[56] Gramine. Gramine Exitless Feature. [Online]. Available: https://gramine.readthedocs.io/en/stable/performance.html#exitless-feature

[57] S. Kim, "An Optimization Methodology for Adapting Legacy SGX Applications to Use Switchless Calls," *Applied Sciences*, vol. 11, no. 18, p. 8379, 2021.

[58] Intel. Intel Processors Supporting Intel SGX. [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions-processors.html

[59] ——. Confidential Computing for 5G Networks. [Online]. Available: https://www.intel.com/content/www/us/en/wireless-network/5g-technology/confidential-computing.html

[60] ——. Migrate to a 5G Core (5GC) Network. [Online]. Available: https://www.intel.com/content/www/us/en/wireless-network/core-network.html

[61] F. Reynaud, F.-X. Aguessy, O. Bettan, M. Bouet, and V. Conan, "Attacks against network functions virtualization and software-defined networking: State-of-the-art," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. IEEE, 2016, pp. 471–476.

[62] I. Farris, T. Taleb, Y. Khettab, and J. Song, "A Survey on Emerging SDN and NFV Security Mechanisms for IoT Systems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 812–837, 2018.

[63] M. Pattaranantakul, R. He, Q. Song, Z. Zhang, and A. Meddahi, "Nfv security survey: From use case driven threat analysis to state-of-the-art countermeasures," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3330–3368, 2018.

[64] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative Technology for CPU based Attestation and Sealing," in *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, vol. 13, no. 7. ACM New York, NY, USA, 2013.

[65] J. Y. Muhammad, M. Wang, Z. Yan, and F. Khan, "Trusted Network Slicing among Multiple Mobile Network Operators," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2020, pp. 1135–1140.

[66] J. M. J. Valero, P. M. S. Sánchez, A. Lekidis, P. Martins, P. Diogo, M. G. Pérez, A. H. Celdrán, and G. M. Pérez, "Trusted Execution Environment-Enabled Platform for 5G Security and Privacy Enhancement," *Security and Privacy Preserving for IoT and 5G Networks: Techniques, Challenges, and New Directions*, pp. 203–223, 2022.

[67] N. Paladi and L. Karlsson, "Safeguarding VNF credentials with Intel SGX," in *Proceedings of the SIGCOMM Posters and Demos*, 2017, pp. 144–146.

[68] E. Norberg, "Evaluation of using secure enclaves in virtualized radio environments," 2019.

[69] R. Poddar, C. Lan, R. A. Popa, and S. Ratnasamy, "SafeBricks: Shielding Network Functions in the Cloud," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 201–216.

[70] H. Duan, C. Wang, X. Yuan, Y. Zhou, Q. Wang, and K. Ren, "LightBox: Full-stack Protected Stateful Middlebox at Lightning Speed," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2351–2367.

[71] B. Trach, A. Krohmer, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer, "ShieldBox: Secure Middleboxes using Shielded Execution," in *Proceedings of the Symposium on SDN Research*, 2018, pp. 1–14.

[72] P. Naik, A. Kanase, T. Patel, and M. Vutukuru, "libVNF: Building Virtual Network Functions Made Easy," in *Proceedings of the ACM symposium on cloud computing*, 2018, pp. 212–224.

[73] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click Modular Router," *ACM Transactions on Computer Systems (TOCS)*, vol. 18, no. 3, pp. 263–297, 2000.