EnclaveVPN: Toward Optimized Utilization of Enclave Page Cache and Practical Performance of Data Plane for Security-Enhanced Cloud VPN

Jaemin Park
The Affiliated Institute of ETRI
Daejeon, Republic of Korea
jmpark@nsr.re.kr

Brent Byunghoon Kang[*]
KAIST
Daejeon, Republic of Korea
brentkang@kaist.ac.kr

ABSTRACT

A cloud Virtual Private Network (VPN) is an essential infrastructure for tenants to connect their on-premise networks with a cloud network. However, tenants are often reluctant to adopt the cloud VPN because of security concerns, such as key disclosure, impersonation, and packet sniffing. Software Guard Extensions (SGX) is a good candidate to address the security concerns because it can create enclaves in the isolated memory (i.e., Enclave Page Cache (EPC)) to protect security-sensitive code and data from malicious access. In this paper, we propose EnclaveVPN, which supports a security-enhanced IPsec gateway using SGX with optimized EPC utilization and practical performance of the data plane. EnclaveVPN leverages enclaves to manage cryptographic keys and execute cryptographic operations for the IPsec gateway. EnclaveVPN allows only encrypted packets to be transmitted within and to/from the cloud network and presents features for optimizing EPC utilization and minimizing overhead in the data plane. We implemented a prototype on a real SGX v1.0 machine (Xeon E-2286M 2.40GHz 8-core CPU). The experiment and benchmark results showed that EnclaveVPN saved the EPC up to 62.5% and achieved approximately 87% of the data plane performance of the non-SGX IPsec gateway.

CCS CONCEPTS

• **Security and privacy** → **Network security**; • **Networks** → **Cloud computing**.

KEYWORDS

SGX, Cloud VPN, IKE, IPsec

ACM Reference Format:
Jaemin Park and Brent Byunghoon Kang. 2023. EnclaveVPN: Toward Optimized Utilization of Enclave Page Cache and Practical Performance of Data Plane for Security-Enhanced Cloud VPN. In *The 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23), October 16–18, 2023, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3607199.3607210

[*] Corresponding author

RAID '23, October 16–18, 2023, Hong Kong, Hong Kong

ACM ISBN 979-8-4007-0765-0/23/10…$15.00
https://doi.org/10.1145/3607199.3607210

1 INTRODUCTION

A cloud Virtual Private Network (VPN) is an essential cloud network infrastructure that enables tenants to connect their on-premise networks with cloud networks using Internet Protocol security (IPsec) [47]. Major cloud service providers offer tenants the cloud VPN [4, 29, 58] by dedicating virtual IPsec gateways (hereafter, "IPsec gateway") for the tenants. The tenants can leverage their Virtual Machines (VMs) to operate third-party IPsec gateways (e.g., Cisco CSR 1000V series [13], Juniper vSRX [62]) available in cloud marketplaces. Using the cloud VPN, the tenants establish site-to-site VPN connections between their on-premise networks and the cloud networks to protect traffic and isolate the private IP address space in the multi-tenant environment.

Although a cloud VPN is widely used and has become indispensable, security administrators are reluctant to introduce the cloud VPN into organizations because of security concerns, such as *key disclosure*, *impersonation*, and *packet sniffing*.

- An attacker can access keys (e.g., authentication keys, ephemeral keys, and session keys) after compromising IPsec gateways through privilege escalation via known vulnerabilities [96–98]. In a multi-tenant environment, an attacker can also access the keys that reside in the IPsec gateways of other guests because of the vulnerabilities of hypervisors that manage the IPsec gateways [94, 95, 99].
- Malicious insiders can intentionally access the keys owned by the IPsec gateway of a victim using cloud management operations. For example, administrators can obtain session keys used in the IPsec gateway of a guest via cloud management operations such as taking a snapshot of the IPsec gateway containing session keys.
- With the leaked authentication keys, an attacker can impersonate a tenant and establish a VPN connection with a cloud VPN. The attacker could use this connection to obtain unauthorized access to the tenant's VMs.
- With the leaked session keys, an attacker can eavesdrop on VPN connections between tenants' on-premise networks and a cloud VPN. Moreover, malicious insiders can sniff packets without authorization within a cloud network if the encryption on packets is optional [28].

Therefore, it is crucial to address the security problems by protecting both keys and packets in a cloud VPN.

Prior works on IPsec gateways in the cloud focus on operational aspects such as user mobility support [54] and efficient resource utilization [80] instead of security enhancement. Studies on general IPsec gateways [70, 71] leverage a hypervisor or TPM [84] as a trust anchor for security enhancement. However, these solutions require further study about the underlying platforms (e.g., multi-tenant environment, recent trust anchors, etc.) for cloud deployment.

Software Guard Extensions (SGX) is a good candidate for addressing the security problems raised in IPsec gateways for a cloud VPN because SGX creates *enclaves* that provide applications with hardware-based protection for security-sensitive data and its associated code. Moreover, Intel continuously releases Xeon processors for data centers, which are supporting SGX and are suitable for hosting the cloud VPN [42]. The enclaves reside in a part of DRAM invisible to other software known as the Enclave Page Cache (EPC). Non-enclave code cannot access the enclaves because the CPU enforces access control on the EPC and fetches the contents of the enclaves from the EPC in an encrypted form. This isolated execution prevents higher-privileged software (e.g., operating systems (OSes) and hypervisors) from accessing sensitive data in the enclaves. Therefore, privileged software cannot access the contents of the enclaves even if the privileged software has information leak vulnerabilities.

However, there are several challenges to the adoption of SGX into IPsec gateways:

1) Limited EPC size for multi-tenant IPsec gateways: The EPC is a restricted resource whose maximum size for a single machine is limited. For example, in [19], the maximum EPC size in SGX v1.0 is 128MB, of which 93MB are usable by the enclave code and data. While the maximum EPC size is drastically increased in SGX v2.0, the EPC is still a restricted resource because allocating the EPC more than the maximum size incurs a significant amount of performance overhead [22]. This limitation inevitably increases the price of SGX-enabled VMs. (e.g., a Confidential Computing VM in Azure is twice as expensive as a general one [59].) Moreover, even in SGX v2.0, a smaller trusted computing base (TCB) is still beneficial with respect to security. To secure IPsec gateways with SGX, the whole or part of the code and data of IPsec gateways must reside in the EPC. Thus, providing multiple tenants with SGX is relatively unrealistic for cloud service providers without addressing this limitation.

2) Performance degradation of data plane caused by the overhead in enclave transitions: Recent studies [87, 88, 101] have shown that the overhead in enclave transitions is over 8,000 CPU cycles. A switchless enclave transition [87] still introduces over 600 to 1,400 CPU cycles [103]. The overhead in the enclave transitions overwhelms the overhead (150 cycles) [103] of a typical system call. Adopting SGX to process each packet cannot avoid the overhead caused by enclave transitions. Thus, this adoption of SGX should not drastically degrade the performance metrics (i.e., throughput, packet per second, and average latency) of the data plane for the IPsec gateways.

To this end, various SGX-based network applications including Tor [49], network function virtualization (NFV) applications [15, 66, 77], middleboxes [6, 21, 27, 34, 89], WireGuard [67], and DTLS [73] have been presented to enhance network security while minimizing overhead in the data plane. However, the existing solutions do not cover the security problems of IPsec gateways in the cloud. Moreover, the solutions do not focus on efficient EPC utilization, which is an essential criterion for cloud service providers to serve tenants with network applications.

Therefore, in this paper, we propose EnclaveVPN, which supports the security-enhanced IPsec gateway using SGX along with the optimization of EPC usage and practical performance of the data plane.

To enhance the security of the IPsec gateway, EnclaveVPN leverages enclaves to manage cryptographic keys (e.g., authentication keys, shared secrets, and session keys) and execute cryptographic operations for Internet Key Exchange (IKE) [45] and Encapsulating Security Payload (ESP) [46]. To prevent attackers from sniffing packets, EnclaveVPN introduces *In-Enclave Forwarding*, which transmits only ESP packets within and to/from a cloud network. In-Enclave Forwarding forwards an ESP packet to its actual destination by replacing the destination IP address of the ESP packet with the IP address of the actual destination (address replacement) inside the enclave.

To optimize the EPC utilization, EnclaveVPN introduces two features; *crypto-partitioning* and *clear-and-seal*. EnclaveVPN splits the components of the IPsec gateway into two planes; IKE for the control plane and ESP for the data plane. Crypto-partitioning locates only cryptographic code and data on the enclaves for each plane. Clear-and-seal nullifies sensitive-but-unnecessary data and seals sensitive-and-usable data using SGX sealing [1] for the control plane. The sensitive-and-usable data are necessary only when rekeying launches before the established VPN sessions expire.

To minimize overhead in the data plane, EnclaveVPN presents Authenticated Decryption-merging (*AD-merging*). AD-merging merges cryptographic operations (authentication & decryption) and address replacement for In-Enclave Forwarding into a single ECall (an invocation of functions located inside an enclave) per each ESP packet.

Accordingly, not only cloud service providers but also tenants benefit from EnclaveVPN in terms of the optimized utilization of limited resources and practical performance of the data plane while enhancing the security of a cloud VPN.

To the best of our knowledge, EnclaveVPN is the first attempt to leverage SGX to improve the security of both planes of the IPsec gateway while supporting optimized EPC utilization and the practical performance of the data plane.

The contributions of EnclaveVPN toward a security-enhanced cloud VPN are listed below:

- **Design of a security-enhanced IPsec gateway.** EnclaveVPN presents enclave control plane and enclave data plane that locate sensitive code and data for the IPsec gateway inside enclaves for security enhancements. (§ 4.3) In-Enclave Forwarding protects traffic within and to/from a cloud network to defend against attackers (including malicious insiders). (§ 4.4) The isolated execution (the enclaves) and In-Enclave Forwarding prevent the attackers from accessing both keys and packets.
- **Optimized EPC utilization.** EnclaveVPN introduces EPC optimization features (crypto-partitioning and clear-and-seal) (§4.5) that enable cloud service providers to save the EPC by up to 65% and to lead ≈88% TCB reduction compared to them for the basic case where the entire IPsec implementation resides in the EPC of a single SGX machine.

- **Practical performance of data plane.** EnclaveVPN merges ECalls for In-Enclave Forwarding operations per a single packet into only one ECall (AD-merging) (§4.4.2) to minimize the overhead caused by enclave transitions. This approach achieved reasonable throughput and packet per second (≥ 87%) compared to the non-SGX setting for AES-256-CBC and HMAC-SHA256-128, which can be considered practical.
- **Prototype implementation on a real SGX machine.** We implemented a prototype of EnclaveVPN on a real SGX v1.0 machine (Xeon E-2286M 2.40GHz 8-core CPU) to measure network performance. This prototype leveraged the Data Plane Development Kit (DPDK) [68] as the network I/O stack and Vector Packet Processing (VPP) [25] as the packet processing framework.

The remainder of this study is organized as follows. Section 2 describes SGX and IPsec. We define the research problem in Section 3, and provide the details of EnclaveVPN in Section 4. We evaluate EnclaveVPN in Section 5, and mention related work in Section 6. We discuss limitations and future work in Section 7.2, and conclude in Section 8.

2 BACKGROUND

2.1 Software Guard Extensions (SGX)

The SGX is an extended set of instructions that supports *enclaves* where security-sensitive code and data are protected by an SGX-enabled processor. The SGX-enabled processor guarantees the confidentiality and integrity of an enclave using an isolated memory area, i.e., the EPC, which cannot be accessed from outside the enclave. When the enclave is loaded and initialized, the SGX platform detects whether the enclave has been altered by comparing the calculated measurement of the enclave with the pre-produced one. Remote attestation allows a remote entity to verify that the enclave is running inside the SGX-enabled processor and that it is trustworthy.

2.1.1 *Isolated Execution Environment.*

The SGX supports a secure container called an enclave, which executes code in an isolated environment. The enclave is isolated from all the other software, including its application process. The SGX realizes this isolation using the EPC, and it adds the Memory Encryption Engine (MEE) [32] to the uncore of the processor to protect the EPC against physical attacks. The MEE is a hardware unit that protects the confidentiality, integrity, and freshness of the traffic communicated between the CPU and the EPC. Cryptographic keys used by the MEE for this protection are generated uniformly at random during boot and never leave the CPU. Using the MEE and mechanisms implemented in the SGX-enabled processor, SGX can achieve an isolated execution environment.

2.1.2 *SGX Attestation.*

The SGX supports two attestation mechanisms, including local and remote attestation. *Local attestation* is a cryptographic approach for internal enclaves to attest other enclaves that reside inside the processor to enable higher-level functions such as remote attestation. An enclave can prove its identity to other enclaves by producing attestation evidence, which includes cryptographic proof that the enclave exists on the same platform. *Remote attestation* is a cryptographic approach for remote entities to attest to the trustworthiness of the underlying hardware platform and to the running enclaves. Intel provides an enclave for remote attestation called Quoting Enclave (QE), which is a privileged enclave in the SGX framework. The QE produces attestation evidence, and the remote entities verify the signature block of the evidence using the public key from the Intel Enhanced Privacy ID (EPID) group key.

2.2 IPsec (Internet Protocol security)

IPsec [47] is a secure network protocol suite designed to protect the IP layer. IKE [35, 45] is a key exchange protocol that provides automatic key management for IPsec. In a cloud VPN, IPsec gateways leverage IKEv2 [45] and ESP (Encapsulating Security Payload) [46] in tunnel mode.

2.2.1 *IKE (Internet Key Exchange).*

The IKE is an automatic key exchange protocol for IPsec that establishes Security Associations (SAs), secure communication sessions between two IKE peers. IKE negotiates IKE SAs and ESP SAs, which define cryptographic algorithms and contain session keys along with other parameters. An IKE SA protects IKE messages, and an ESP SA protects the actual IP packets. The Security Parameter Index (SPI) is an identification of these SAs.

When establishing an IKE SA, IKE peers generate a shared secret from an ephemeral Diffie-Hellman (DH) [18] exchange. Using a master key generated from the shared secret, the IKE peers derive session keys for the IKE SA, including a key for deriving session keys of ESP SAs, keys for authenticating and encrypting/decrypting subsequent IKE messages, and keys for the IKE peer authentication. Therefore, if adversaries access the shared secret or the master key, the adversaries can collapse the entire IPsec security protocol [16].

The IKE launches rekeying for SAs proactively before the lifetime metrics of the SAs are reached. Only parts of the session keys in an IKE SA are required before the rekeying of the SAs begins. Therefore, this part can remain unavailable until the expiration of the SAs is imminent.

2.2.2 *ESP (Encapsulating Security Payload).*

The ESP is one of the IPsec protocols, which supports data-origin authentication, data integrity, and data confidentiality. The ESP in the tunnel mode encrypts and authenticates (Integrity Check Value (ICV) calculation) the entire IP packet including the IP header, and encapsulates it into a new IP packet. Owing to this property, the ESP in the tunnel mode is widely used in constructing VPN connections for a cloud VPN.

2.2.3 *DPDK (Data Plane Development Kit) and VPP (Vector Packet Processing).*

DPDK [68] is a programming framework for high-speed data plane applications in user mode. A DPDK application should access all devices via polling to reduce the performance overhead imposed by interrupt processing. Moreover, DPDK provides a pipeline model by passing packets or messages between cores via the rings for better efficiency.

FD.io's VPP [25] supports a layer 2-4 network stack in Linux userspace and leverages DPDK for fast I/O. VPP processes multiple packets (vectors) at a time instead of processing one packet. VPP processes the vectors via a set of functions, decomposed into a packet processing graph.

3 PROBLEM DEFINITION

3.1 System Model

We consider a tenant operates an IPsec appliance in its on-premise network, and the appliance connects to a cloud VPN to manage the tenant's VMs in a cloud network. (Figure 1)

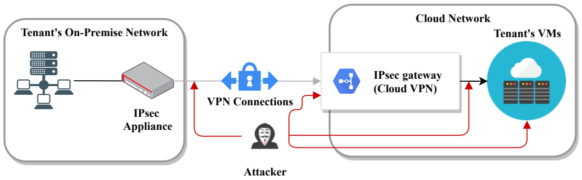


Figure 1: System and Threat Models

A cloud VPN is either an IPsec gateway managed by cloud service providers or a third-party IPsec gateway running in a tenant's VM. Hypervisors, where the cloud VPN and VMs are running, run on commodity servers that support SGX-enabled processors [37] and DPDK-compatible network interfaces. The IPsec appliance operates in an on-premise network under the tenant's control. Thus, we assume that the IPsec appliance runs as intended. We assume that both the cloud VPN and the tenant's VMs are connected to Intel Attestation Service (IAS) [39]. Thus, the tenant can launch remote attestation via the protocol supported by the SGX implementation.

3.2 Threat Model

Adversaries can control entire software stacks on a cloud VPN, except the code inside enclaves. For example, the adversaries have compromised the OS, the hypervisor, or the cloud VPN. These adversaries include malicious insiders and/or remote attackers in the untrusted cloud. The adversaries access and tamper with memory where the cloud VPN runs. Then, the adversaries attempt to read the cloud VPN's keys (e.g., authentication keys, ephemeral keys, or session keys). The adversaries can also sniff packets within and to/from a cloud network by intercepting the packets or decrypting the packets with the leaked keys.

We trust only SGX-enabled processors, enclaves, and mechanisms implemented in the processors, which is the same assumption in the threat model of SGX [56]. Thus, we assume that the adversaries cannot compromise SGX components on the compromised OS, hypervisor, or cloud VPN. A tenant's VMs are supposed to support the SGX-protected library OSes or containers [2, 5, 90] for running the tenant's applications. We assume that the code located in the enclaves is correct and does not leak any key intentionally. We also assume that the tenant's on-premise network is trusted, and the adversaries cannot compromise the underlying cryptographic primitives and protocols.

Similar to other SGX-based network applications [21, 34, 49, 66, 67, 89], all hardware attacks [10, 74, 92, 104] and cache side-channel attacks [9, 23, 30, 33, 52, 61, 75, 78, 93, 102] along with associated attacks such as cross-tenant attacks (e.g., data leakage, side-channels, etc.) are beyond the scope of this study. Thus, orthogonal to our approach, we assume that these attacks can be detected and addressed using known countermeasures [3, 8, 11, 12, 14, 31, 64, 76, 79]. Similarly, denial-of-service (DoS) attacks are beyond the scope of this study because adversaries (e.g., privileged attackers) can simply discard key exchange messages and packets or the co-resident tenants prevent non-SGX portion of code to be properly executed.

3.3 Design Goals

To enhance the security of the cloud VPN, EnclaveVPN should prevent adversaries from accessing keys of the cloud VPN and sniffing packets within and to/from a cloud network Further, EnclaveVPN should save limited resources and guarantee a practical performance of the data plane for feasibility.

The goals for EnclaveVPN are defined as follows.

G1: Secrecy of cryptographic keys. EnclaveVPN should guarantee the secrecy of cryptographic keys, including authentication keys, shared secrets, and session keys, by restricting access to these keys only to trustworthy parties. Adversaries should be unable to access any cryptographic key to disguise themselves as a legitimate entity, falsify key exchange messages, or eavesdrop on packets.

G2: Protection of packets within and to/from a cloud network. EnclaveVPN should protect all packets within and to/from the cloud network by transforming all packets into ESP packets. Thus, adversaries should not be able to extort any valuable content from the packets.

G3: Isolated execution of IPsec gateway. EnclaveVPN should feature the isolated execution of key exchange and packet processing for each tenant. In a multi-tenant environment, it is crucial to thwarting attack propagation to other tenants. Thus, even if a cloud VPN is compromised, adversaries should not be able to access any data of other tenants by accessing the memory pages of the cloud VPN.

G4: Feasible IPsec gateway for a cloud VPN. EnclaveVPN should support the optimized EPC utilization and provide a practical performance of the data plane for better feasibility. Otherwise, EnclaveVPN is impractical for both cloud service providers and tenants in terms of efficiency in the EPC utilization and the data plane performance. This impracticality results from the fact that the EPC has a limited size and enclave transitions incur substantial performance overhead in packet processing.

4 DESIGN

4.1 Overview

EnclaveVPN is a security-enhanced IPsec gateway for a cloud VPN that leverages enclaves to execute IKE and ESP securely. EnclaveVPN consists of enclave control plane and enclave data plane, and they each operate their enclaves (EnclaveIKE and EnclaveESP). This plane separation in EnclaveVPN observes that the usual implementations of IPsec gateways manage IKE SAs and ESP SAs separately. An IKE daemon (e.g., strongSwan [83]) is a typical userland implementation, and it manages the IKE SAs. The daemon communicates with an IPsec implementation in the kernel (e.g., XFRM in Linux) to deliver the negotiated ESP SAs.

A tenant's IPsec appliance in the on-premise network negotiates SAs including IKE and ESP SAs (VPN connections) with EnclaveVPN. EnclaveVPN extends these VPN connections to protect packets within a cloud network by In-Enclave Forwarding. To this end, EnclaveVPN delivers the ESP SAs to the IPsec implementations of the tenant's VMs. Note that the tenant's VMs support the

SGX-protected library OSes or containers [2, 5, 90] where the tenant's applications and the IPsec implementations run. The tenant can then connect to its VMs via the established VPN connections (Figure 2). Figure 3 depicts the internal architecture of EnclaveVPN.

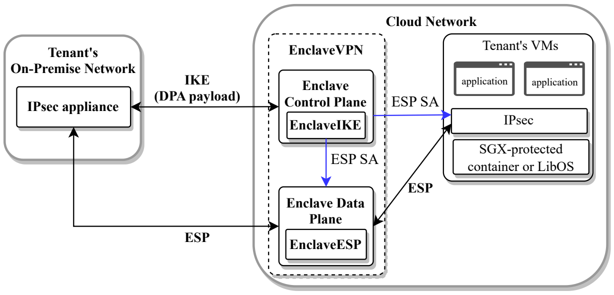


Figure 2: Overall system based on EnclaveVPN

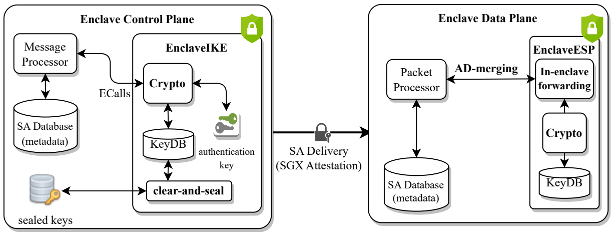


Figure 3: Internal architecture of EnclaveVPN

EnclaveVPN supports In-Enclave Forwarding to prevent adversaries from sniffing packets within and to/from a cloud network. Rather than forwarding a decrypted packet to a destination, EnclaveESP replaces the destination address of the outer IP header of the ESP packet with the IP address of the inner IP header. The IP address of the inner IP header is an actual destination address and can be revealed only after decrypting the ESP packet. Thus, only ESP packets travel within the cloud network, as shown in Figure 2. EnclaveVPN leverages AD-merging to minimize the overhead in the data plane caused by enclave transitions for In-Enclave Forwarding. AD-merging performs cryptographic operations (ICV verification & decryption) with the address replacement for a single ESP packet at only one ECall.

For In-Enclave Forwarding, EnclaveVPN proposes a new IKE payload, i.e., Data Plane Address (DPA), which contains a list of IP addresses (destinations of ESP SAs). After negotiating SAs (IKE SAs and ESP SAs) for VPN connections, enclave control plane delivers the ESP SAs to enclave data plane and the IPsec implementation of the tenant's VMs via secure channels established by local (inside EnclaveVPN) and remote (for tenant's VMs) attestation in SGX [1].

By utilizing crypto-partitioning, EnclaveVPN loads only cryptographic operations and keys to the two enclaves to save the limited EPC. EnclaveVPN supports clear-and-seal to clear sensitive-but-unnecessary data (shared secrets, master keys, the part of session keys in the IKE SAs, the derived session keys for the ESP SAs) from the EPC and seal the sensitive-and-usable data (one of the session keys in the IKE SAs for rekeying) to the untrusted memory to save more EPC space.

4.2 Bootstrapping

Before launching EnclaveVPN, a tenant needs a bootstrapping that provisions authentication keys to EnclaveIKE. We suppose that the signed enclaves (EnclaveIKE and EnclaveESP) along with the EnclaveVPN implementation have been deployed to the cloud securely as in [72].

The tenant establishes a secure channel with EnclaveIKE via the standard remote attestation [1]. The tenant requests a measurement of EnclaveIKE signed by the SGX-enabled processor and communicates with the IAS for verification. This attestation ensures that EnclaveIKE is not altered and that it runs in a genuine SGX-enabled processor. Moreover, this attestation derives session keys for the secure channel between EnclaveIKE and the tenant. Then, the tenant transmits the authentication keys to EnclaveIKE through the channel. This SGX-backed procedure ensures that the authentication keys are only visible inside EnclaveIKE, and prevents adversaries from accessing the keys in the cloud.

4.3 Security Enhancement for Planes

EnclaveVPN leverages two enclaves (EnclaveIKE, EnclaveESP), the SGX-enabled processors, and SGX implementations as trust anchors to secure enclave control plane and enclave data plane. Instead of executing all IKE and ESP operations in the untrusted memory, EnclaveVPN loads and executes them in the EPC.

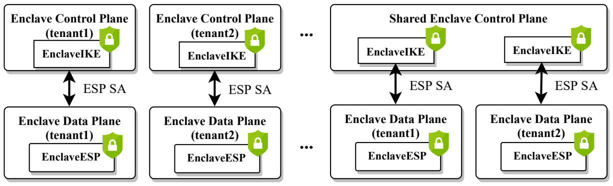
4.3.1 Securing enclave control plane by EnclaveIKE. EnclaveIKE provides the message processor of enclave control plane with cryptographic operations. The cryptographic operations include the DH computation, the session key derivation, the authentication calculation, and the IKE message protection (encryption/decryption, and integrity calculation). Inside the EPC, EnclaveIKE creates cryptographic contexts, initializes the contexts, and uses the contexts to fulfill the cryptographic operations for IKE. Because the keys reside inside EnclaveIKE and do not leave EnclaveIKE during the operations, adversaries cannot access these keys illegally.

4.3.2 Securing enclave data plane by EnclaveESP. A packet processor of enclave data plane leverages EnclaveESP to perform In-Enclave Forwarding. The packet processor invokes ECalls to process the cryptographic operations and the address replacement against inbound ESP packets using the session keys of an ESP SA. Once receiving the session keys of the ESP SA from enclave control plane (EnclaveIKE), EnclaveESP creates cryptographic contexts and initializes the contexts with the session keys. The session keys reside inside the EnclaveESP without leaving EnclaveESP during In-Enclave Forwarding because EnclaveIKE distributes the session keys of the ESP SA to EnclaveESP via a secure channel established by remote attestation (§ 4.3.3) and the session keys do not leave EnclaveESP to fulfill In-Enclave Forwarding. Moreover, In-Enclave Forwarding allows only ESP packets to be placed in the untrusted memory because EnclaveESP performs all operations without leaving itself. Thus, adversaries can access neither the session keys nor the processing plaintext.

4.3.3 SA (Security Association) delivery to enclave data plane and tenant's VMs. To establish an ESP SA with an enclave control plane, a tenant's IPsec appliance inserts the IP addresses of both enclave data plane and the tenant's VMs to the `DPA` payload. The `DPA` payload allows the tenant to include a list of IP addresses where enclave control plane delivers the ESP SAs. Besides the IKE generic components (e.g., IKE header, payload length, etc.), the `DPA` payload consists of a number of IP addresses and two or more individual IP addresses.

Enclave control plane delivers the ESP SAs to EnclaveESP in enclave data plane and the IPsec implementations of the tenant's VMs as specified in the `DPA` payload via a secure channel established by the standard SGX attestation [1], which is similar to 4.2. EnclaveESP creates cryptographic contexts with session keys of the received ESP SA. Then, EnclaveESP returns the metadata (indices) of the contexts to the untrusted memory. Enclave data plane passes this metadata to EnclaveESP to indicate the contexts for In-Enclave Forwarding. Similarly, the IPsec implementations in the tenant's VMs leverage the received ESP SA to configure the VPN connection.

4.3.4 Plane dedication for tenant's isolated execution. EnclaveVPN provides each tenant with its dedicated enclave control plane and enclave data plane for the isolated execution of the IPsec gateway, as depicted in Figure 4a. For better efficiency in resource utilization, EnclaveVPN can share an enclave control plane among tenants like Protego[80]. However, EnclaveVPN should dedicate individual EnclaveIKE to each tenant in the shared enclave control plane, as shown in Figure 4b. Then, the tenant provisions its authentication key to the dedicated EnclaveIKE via a secure channel established by SGX attestation during the bootstrapping. This isolated execution for each tenant prevents adversaries from accessing the keys of other tenants, even if the underlying software (e.g., hypervisor) is compromised. Moreover, the unprotected shared component cannot access the keys of other tenants because all keys including ephemeral ones reside in the EPC.


Enclave Control Plane (tenant1)
EnclaveIKE
ESP SA
Enclave Data Plane (tenant1)
EnclaveESP
Enclave Control Plane (tenant2)
EnclaveIKE
ESP SA
Enclave Data Plane (tenant2)
EnclaveESP
...
Shared Enclave Control Plane
EnclaveIKE
EnclaveIKE
ESP SA
ESP SA
Enclave Data Plane (tenant1)
EnclaveESP
Enclave Data Plane (tenant2)
EnclaveESP
...

(a) Non-shared enclave control plane **(b) Shared enclave control plane**

Figure 4: Dedication of planes and enclaves

4.4 Packet protection with practical performance of data plane

EnclaveVPN offers In-Enclave Forwarding to prevent adversaries from eavesdropping on packets transmitted within and to/from a cloud network. EnclaveVPN supports AD-merging to guarantee the practical performance of the data plane.

4.4.1 Packet protection by In-Enclave Forwarding. Enclave data plane transmits only ESP packets within and to/from a cloud network via In-Enclave Forwarding. After receiving an ESP packet, enclave data plane invokes EnclaveESP to perform the ICV verification and the decryption of the packet. Then, EnclaveESP replaces the destination address of the outer header of the ESP packet with the IP address of the inner header. Finally, enclave data plane forwards the ESP packet returned by EnclaveESP to its actual destination.

For this feature, a tenant's applications and IPsec implementations should run inside SGX-protected library OSes or containers. Otherwise, inside attackers can sniff packets at the upper layer after the packets are decrypted in the IP layer by compromising the underlying OSes or hypervisors. The IPsec implementation is supposed to support remote attestation and receive a negotiated ESP SA for a VPN connection.

4.4.2 Practical performance of data plane by AD-merging (Authenticated Decryption-merging). To minimize the overhead in In-Enclave Forwarding, EnclaveVPN presents AD-merging, which merges cryptographic operations (authentication & decryption) and address replacement for In-Enclave Forwarding into a single ECall per each ESP packet. Thus, EnclaveESP exports only one ECall (`ecall_ad()`), as defined in Table 1, to enclave data plane for In-Enclave Forwarding. Enclave data plane passes input parameters to EnclaveESP; *idx* is the metadata (indices) to indicate the session keys of the ESP SA; *iv* is an initialization vector (IV); the information (memory address and length) of the encrypted part (*enc*, *enc_len*), the authentication part (*auth*, *auth_len*), an ICV (*icv*, *icv_len*), and the whole output ESP packet (*esp*, *esp_len*).

ECall		`ecall_ad()`
Parameters	return value	*ret*
	index of metadata	*idx*
	IV	*iv*
	encrypted part	*enc_src*, *enc_src_len*
	authenticated part	*auth_src*, *auth_src_len*
	ICV	*icv*, *icv_len*
	output ESP packet	*esp*, *esp_len*

Table 1: ECall for AD-merging

With the input parameters, EnclaveESP processes as shown in Algorithm 1. Using *idx*, EnclaveESP searches the matched cryptographic context. EnclaveESP calculated an ICV (*auth*) over *auth_src*. EnclaveESP checks whether the computed ICV (*auth*) matches *icv*. If this verification fails, EnclaveESP returns a non-zero value (*ret*). Otherwise, EnclaveESP allocates a temporary memory (*tmp*) where EnclaveESP temporarily holds the decrypted packet inside the enclave. Because EnclaveESP has already initialized *ctx* with the session keys of an ESP SA, EnclaveESP needs to re-initialize *ctx* by feeding *iv*. EnclaveESP decrypts *enc_src* and replaces the destination address of the outer header($IP_{dst}{}^{outer}$) of *esp* with the IP address of the inner header($IP_{dst}{}^{inner}$), which is the actual address of a destination as depicted in Figure 5.

AD-merging converts any existing authenticate-then-decrypt routine into a single (atomic) ECall instead of two separate ECalls; one for ICV verification and the other for decryption. This ECall also executes the address replacement for In-Enclave Forwarding inside the enclave. Thus, AD-merging reduces the number of enclave transitions to only one.

Moreover, In-Enclave Forwarding does not introduce any additional overhead to enclave data plane compared to the case without

Algorithm 1 Pseudo-code of `ecall_ad()`

```
$ret \leftarrow 0$
$ctx \leftarrow$ lookup with $idx$
$auth \leftarrow$ calculate an ICV over $auth\_src$
if verify $auth \neq icv$ then $ret \leftarrow -1$
else
    $tmp \leftarrow$ allocate a temporary memory
    re-initialize $ctx$ with $iv$
    $tmp \leftarrow$ decrypt $enc\_src$
    $esp \leftarrow$ replace ${IP_{dst}}^{outer}$ with ${IP_{dst}}^{inner}$ in $tmp$
```

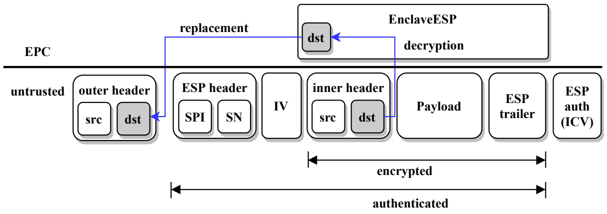


Figure 5: Address replacement inside EnclaveESP

In-Enclave Forwarding. Enclave data plane originally verifies and decrypts an ESP packet via EnclaveESP, and it then forwards a decrypted packet to its destination. Instead of forwarding the decrypted packet, In-Enclave Forwarding changes the ESP packet's destination address to the decrypted address of the inner header. Because In-Enclave Forwarding requires only one ECall, the overhead to enclave data plane is almost identical to the case without In-Enclave Forwarding.

4.4.3 Overall packet protection of enclave data plane by In-Enclave Forwarding and AD-merging. Enclave data plane works with EnclaveESP to process In-Enclave Forwarding as shown in Algorithm 2. Upon receiving the ESP packet (p), the packet processor of enclave data plane constructs a new packet structure that contains address pointers of the authentication part and the encryption part along with esp that points to p. Then, the packet processor invokes `ecall_ad()` to execute Algorithm 1. If EnclaveESP returns a non-zero value (ret), enclave data plane simply discards p. Finally, enclave data plane forwards p', a new ESP packet whose destination IP address was replaced by EnclaveESP, to its destination.

Algorithm 2 Pseudo-code of enclave data plane

```
construct a new packet structure for AD-merging
invoke EnclaveESP via ecall_ad()
if $ret \neq 0$ then drop $p$
forward $p'$ to its destination
```

To accelerate this operation, enclave data plane allocates the shared memory. With the allocated shared memory, enclave data plane constructs the input parameters with the input ESP packet before `ecall_ad()` and the output ESP packet after `ecall_ad()`. Enclave data plane supports In-Enclave Forwarding securely because EnclaveESP performs the cryptographic operations and the address replacement for In-Enclave Forwarding inside the enclave. Moreover, due to In-Enclave Forwarding, enclave data plane can access only ESP packets located in the shared memory.

4.5 EPC Saving for Planes

EnclaveVPN supports two EPC optimization features (crypto-partitioning and clear-and-seal) to save the EPC.

4.5.1 Crypto-partitioning for enclave control plane and enclave data plane. The planes in EnclaveVPN offload only cryptographic operations (code) and session keys (data) to the enclaves to minimize attack surfaces. (Figure 3) EnclaveIKE and EnclaveESP utilize KeyDBs as session key storage and implement cryptographic operations for handling IKE messages and ESP transformations.

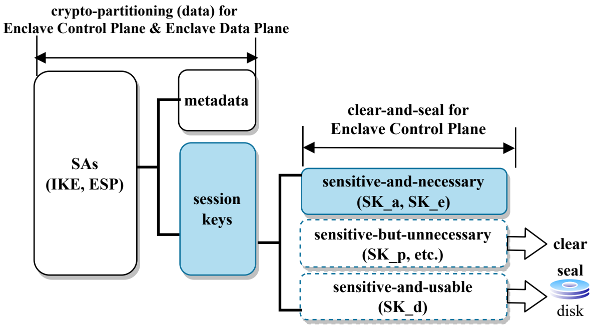


Figure 6: Crypto-partitioning (data) and clear-and-seal of EnclaveVPN; only the blue parts reside in the EPC.

The IKE and ESP SAs comprise metadata (e.g., SPIs, SA lifetimes, etc.) and session keys. In crypto-partitioning (data), EnclaveVPN stores session keys in the enclaves, and the metadata remains in the unprotected memory. (Figure 6) The information in metadata is used to manage configurations for VPN connections. However, the information in the metadata does not pose security concerns as the adversaries cannot sniff the connections without the session keys. Thus, crypto-partitioning (data) profits from the efficient EPC utilization along with the defense against malicious accesses.

In crypto-partitioning (data), EnclaveVPN utilizes enclaves to perform cryptographic operations. Crypto-partitioning (code) requires developers to partition existing IKE and ESP implementation into trusted (enclave) and untrusted parts. A possible solution is the use of automated tools (e.g., Glamdring [53]). Because usual IPsec implementations already separate two parts (cryptographic and non-cryptographic operations) and the cryptographic operations generally include routines to use API calls (e.g., SSL library's APIs), manual partitioning is another possible solution.

4.5.2 Clear-and-seal for enclave control plane. IKE manages SAs from the initial negotiations to the terminations. For this management, IKE peers exchange IKE messages for notifications and rekeying, occasionally or periodically after establishing VPN connections. The IKE peers authenticate each other using session keys for authentication (SK_p) in an IKE SA. The IKE peers protect the IKE messages using session keys for integrity-protection and encryption/decryption (SK_a, SK_e) in the IKE SA. Further, the IKE peers perform rekeying using a session key for key derivation

(SK_d) in the IKE SA to renew the session keys proactively before the expiry.

With the aforementioned characteristics of SAs, enclave control plane saves the EPC as depicted in Figure 6 by supporting clear-and-seal. During the SA establishment, EnclaveIKE generates shared secrets, master keys, and session keys of an IKE SA and ESP SAs inside the EPC. After establishing the SAs, the part of session keys (SK_d, SK_a, SK_e) are used for the SA management, and EnclaveIKE distributes the session keys of the ESP SAs to EnclaveESP. Then, EnclaveIKE nullifies the shared secrets, the master keys, SK_p, and the session keys of the ESP SA (sensitive-but-unnecessary) by clear-and-seal. Further, clear-and-seal seals SK_d (sensitive-and-usable), whose utilization is predictable (i.e., rekeying). Only the sensitive-and-necessary part (SK_a, SK_e) for the IKE message protection resides in the EPC for EnclaveIKE.

Note that SGX sealing [1] does not guarantee the freshness of the sealed data, and this mechanism can be susceptible to rollback attacks [82]. Thus, clear-and-seal can store the sealed data in non-volatile storage on the same platform with the utilization of the monotonic counters. Clear-and-seal can also leverage ROTE [55] or Narrator [63] to prevent the rollback attacks and to recover the sensitive-and-usable part (SK_d) even though adversaries force EnclaveIKE to reboot.

4.6 Implementation

We implemented the EnclaveVPN prototype [1] on an Intel NUC9VXQNX (Xeon E-2286M 2.40GHz 8-core CPU) machine running Ubuntu 16.04 LTS (64-bit). The prototype leverages DPDK 18.11 [68] and VPP 19.08 [25]. We used SGX SSL [40] and the SGX SDK for Linux [41] to implement EnclaveIKE and EnclaveESP.

We added a plugin into the vanilla VPP for enclave control plane. The implementation of enclave control plane handles IKE messages by invoking ECalls of EnclaveIKE. The implementation of enclave data plane consists of a VPP plugin that supports In-Enclave Forwarding and the porting of the packet processor into the core network stack of the VPP. Whenever a packet arrives at the prototype, enclave data plane constructs a new packet structure for AD-merging and enqueues the structure. Then, the VPP plugin of enclave data plane dequeues the structure and invokes `ecall_ad()` to process the packet. To emulate In-Enclave Forwarding, EnclaveESP implements the allocation of the temporary memory inside the enclave.

As indicated in [66, 89], the implementation of enclave data plane utilizes shared memory between the packet processor and EnclaveESP. The prototype allocates this memory from 4GB of RAM by using huge pages of size 1GB. EnclaveESP processes packets on receiving the memory addresses of the packets from the packet processor without copying the packets into its EPC pages. This approach enhances the throughput further because the huge pages can avoid expensive translation lookaside buffer flushing. This approach does not reveal any plaintext because In-Enclave Forwarding locates only ESP packets in this shared memory.

5 EVALUATION

5.1 EPC Utilization of EnclaveIKE and EnclaveESP

We evaluated two EPC optimization features: crypto-partitioning and clear-and-seal. We highlight that crypto-partitioning achieves 88% code (TCB) reduction compared to `Whole`, defined as the case where the entire IPsec implementation resides in the EPC of a single SGX machine. Crypto-partitioning and clear-and-seal save EPC (data) 62.5% more than `Whole`. `Whole` does not confine any specific IPsec implementation, but we use the IPsec implementation of VPP [24] for the comparison.

5.1.1 *LoC reduction by crypto-partitioning (code).*

EnclaveVPN presents crypto-partitioning to utilize the EPC efficiently and minimize attack surfaces (i.e., small TCB). We refer to the IPsec implementation of VPP [24], which comprises protocol and cryptographic parts. We measured lines of code (LoC) in both EnclaveVPN where only the cryptographic part resides in the EPC and `Whole` where the entire IPsec implementation resides in the EPC using Count Lines of Code (CLOC) [17]. For this measurement, we counted only the code for API calls in both EnclaveVPN and `Whole` to use SSL libraries, excluding the libraries themselves. The result in Table 2 indicates that EnclaveVPN shrinks LoC (TCB) by approximately 88% compared to `Whole`. Crypto-partitioning also benefits the small TCB compared to AMD SEV (Secure Encrypted Virtualization), one of the alternative approaches for the cloud, because AMD SEV requires the entire IPsec implementation including OS to reside in the encrypted memory [60].

	`Whole`	EnclaveVPN	
		EnclaveIKE	EnclaveESP
LoC	6758	521	268

Table 2: LoC comparison; `Whole` denotes the entire IPsec implementation resides in the EPC of a single SGX machine.

5.1.2 *Optimized EPC utilization by crypto-partitioning (data) and clear-and-seal.*

To analyze the EPC usage of EnclaveVPN, we measured the peak memory (heap) usage of two enclaves using `sgx_emmt` [41], as depicted in Table 3. We measured the memory usage of `Whole` by comparatively examining the heap variation before and after the VPN connections using VPP's command (`show_memory`). The heap utilization increases because cryptographic contexts are initialized with session keys whenever the VPN connections are established.

Table 3 depicts the EPC usage of EnclaveVPN and `Whole`. As more VPN connections are established, the EPC usages for both cases increase because each VPN connection generates a new cryptographic context initialized with session keys. EnclaveVPN utilizes the `std:map` library to store session keys inside the enclaves, whereas `Whole` defines a C-structure to store all IKE contexts. Thus, for one SA, EnclaveIKE consumes more memory due to `std:map`, but `Whole` occupies more memory than EnclaveIKE as the number of SAs increases. Before implementing clear-and-seal, EnclaveVPN saves the EPC by 56.39% more than that for `Whole`. Clear-and-seal

[1] The source code is available at https://github.com/jmpetrus/EnclaveVPN

	Whole	EnclaveVPN		
		EnclaveIKE		EnclaveESP
		w/o C&S	w/ C&S	
SAs	Used EPC for cryptographic contexts(data)			
1	72KB	120KB	≈ 120KB	4KB
10	164KB	152KB	146KB	28KB
100	1.24MB	412KB	349KB	256KB
500	6.09MB	1.48MB	1.18MB	1.28MB
1,000	12.13MB	2.80MB	2.18MB	2.55MB

Table 3: Comparison of EPC usage; C&S denotes clear-and-seal, and `Whole` denotes the case where the entire IPsec implementation resides in the EPC of a single SGX machine.

can save 3.66~5.11% further compared to EnclaveVPN without clear-and-seal. Finally, EnclaveVPN with crypto-partitioning (56.39%) and clear-and-seal (3.66~5.11%) saves the EPC by 62.5% more than that for `Whole`.

In SGX v2.0, an enclave can add new pages on-demand; however, this incurs paging if the enclave exceeds the EPC size [101]. Thus, it is still valuable to save the EPC as far as possible in SGX v2.0 because paging is expensive and has a significant effect on enclave performance.

5.2 Performance Evaluation of enclave control plane

The SGX utilization for the cloud VPN introduces inevitable performance penalties in IKE. To evaluate the IKE performance, we measured the tunnel setup time and rate. Throughout this evaluation, we denote `Native` as a vanilla implementation with no enhanced security.

	`Native`	EnclaveVPN
Tunnel setup time	20.19	26.72
Tunnel setup rate	99	60

Table 4: Comparison of tunnel setup time (ms) and tunnel setup rate (tunnels per second)

5.2.1 Tunnel setup time and rate for enclave control plane. We measured the elapsed time of tunnel setup for establishing a single tunnel. We also measured the elapsed time for establishing multiple tunnels to calculate the tunnel setup rate. We leveraged strongSwan [83] as a tenant's IPsec appliance to leverage its load tester plugin for the measurement. We configured the implementations to use AES-256-CBC for encryption and decryption, HMAC-SHA256-128 for integrity verification, SHA256 for pseudo-random function, and MODP-3072 for DH computation. We assume that the bootstrapping in 4.2 is already completed before measuring the tunnel setup time and rate. Bootstrapping happens rarely and the overhead is not significant because the underlying SGX attestation with a simple data transfer entails little overhead (56.05 ms [51]).

Table 4 shows the results of the tunnel setup time and rate for `Native` and EnclaveVPN. We measured the elapsed time for a single tunnel. EnclaveVPN records about 32.36% overhead compared to `Native`. EnclaveVPN invokes 13 ECalls, which differs from `Native`. These ECalls comprise a single ECall for the DH computation, seven for the pseudo-random functions, four for the encryption/decryption and integrity verification of IKE messages, and one for clear-and-seal.

To evaluate the tunnel setup rate, we measured the elapsed time to establish multiple tunnels. For this measurement, we increased the number of simultaneous tunnel initiations until any single tunnel failed. Then, we calculated the tunnel setup rate using the elapsed time to establish the multiple tunnels. The results indicate that EnclaveVPN establishes 39.39% fewer tunnels per second than `Native`. As mentioned in 5.2.2, the performance gap caused by the ECall overhead yields to the difference of the tunnel setup rate between `Native` (0 ECall) and EnclaveVPN (13 ECalls).

	`Native`	EnclaveVPN
DH computation	9.853	15.168
pseudo-random functions	0.063	0.137
encryption/decryption	0.005	0.474
integrity verification	0.006	0.077
clear-and-seal	-	0.032
Total	9.928	15.888

Table 5: Elapsed time for cryptographic operations (ms)

5.2.2 Detail in performance of enclave control plane. To analyze the performance in the tunnel setup time and rate, we measured the elapsed time for each cryptographic operation that EnclaveIKE supports as depicted in Table 5. Each elapsed time is the accumulated time if multiple cryptographic operations are processed. The performance gap between `Native` and EnclaveVPN (37.51%) is almost identical to the gap in the tunnel setup time and rate. Thus, it is clear that the ECall overhead (enclave transitions and cryptographic operations) causes the performance gap in the tunnel setup time and rate.

Note that the sealed SK_d should be unsealed before rekeying, whose overhead is known to be less than the sealing operation [44, 100]. The overhead of the unsealing operation is not included in this measurement because rekeying, which needs unsealing the sealed SK_d, is not part of the tunnel setup (i.e., initial tunnel establishment).

5.3 Network Performance of enclave data plane

The SGX utilization also introduces inevitable performance penalties in ESP. To evaluate the ESP performance, we measured the network performance metrics (i.e., throughput, packet per second, and average latency). Throughout this evaluation, we denote `Basic` as straightforward adoption (without AD-merging) of SGX into an implementation.

5.3.1 Practical performance of enclave data plane by AD-merging. We measured throughput, packet per second, and an average latency of enclave data plane by utilizing the RFC 2544 [7] compliant network tester, Agilent N2X [86]. Note that enclave data plane implements the emulated In-Enclave Forwarding in EnclaveESP for this experiment.

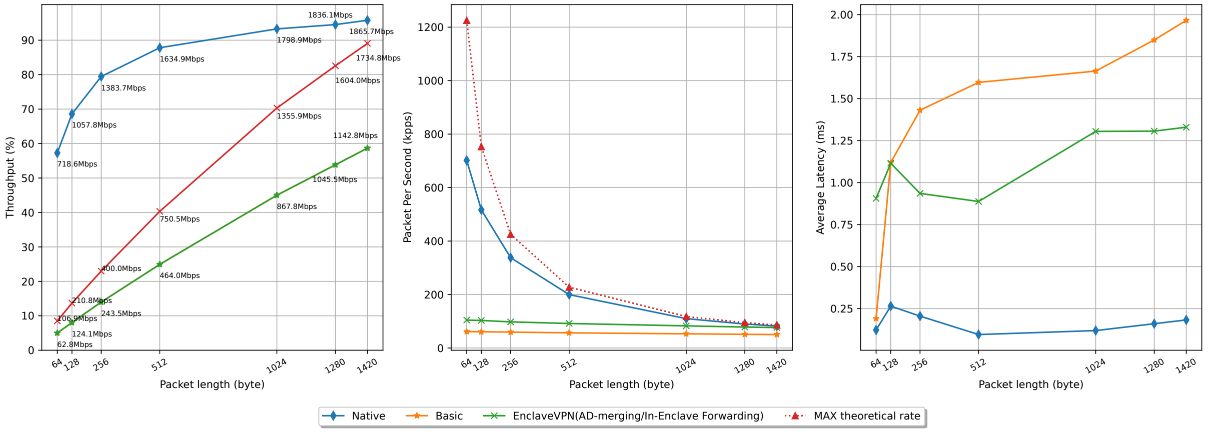


Figure 7: Performance measurement of enclave data plane (AES-256-CBC and HMAC-SHA256-128)

We define the device under test as a single tunnel between a pair of the same VPN implementations (Native, Basic, and EnclaveVPN). This configuration is used to measure the real-world IPsec performance [7, 81]. The network tester limits the traffic generation to the aggregated 2Gbps traffic for the 1Gbps links. We configured the implementations to use AES-256-CBC and HMAC-SHA256-128.

We compare the performance of EnclaveVPN against Native, which could result in the highest performance for the 1Gbps links. We also included the maximum theoretical rate of packet per second for the comparison. We measured the maximum effective throughput for each packet length, which is the fastest rate to the theoretical limit of the media per each packet length, as the throughput.

Note that EnclaveVPN is based on IPsec that provides application-independent security. Thus, we evaluate the performance against various packet sizes (the emulation of various applications) by following the same procedure used in previous works: Protego [80], PIPSEA [65], P4-IPsec [36], [57], [91].

Figure 7 shows the benchmark results on throughput (annotated with aggregated throughput (Mbps)), packet per second, and the average latency of the implementations (Native, Basic, and EnclaveVPN) for various packet lengths.

The result shows that EnclaveVPN records practical throughput and packet per second compared to Native for packets whose lengths are more than 1280-byte. SGX-enabled implementations (EnclaveVPN, Basic) output an almost uniform packet per second for all packet lengths, and this observation is distinct from that of Native. The overhead in the enclave transitions results in this uniform packet per second because the CPU cycles consumed by each enclave transition (8,000 cycles) overwhelm those caused by each packet processing in DPDK (over 100 cycles) [26]. This observation directly influences the throughput and shows similar tendencies to those of the SGX-enabled implementations.

Further, the results indicate that AD-merging improves the throughput and packet per second by 51.80%~70.32% compared to that for Basic. This enhancement is possible because AD-merging reduces the enclave transitions for each packet from two to one. The results also indicate that the adoption of SGX (Basic, EnclaveVPN) degrades the average latency compared to that for Native.

AD-merging accomplishes the practical performance of the data plane for 1280-byte and 1420-byte packets since the throughput and packet per second of EnclaveVPN output more than 87% of Native. The throughput and packet per second of EnclaveVPN decrease against small packets (≤ 512-byte). This is reasonable because the number of enclave transitions increases as the packets shorten for the given input traffic.

Packet size	Throughput (Mbps)	Average latency (ms)
1420-byte	1734.8	1.33
1280-byte	1603.9	1.31
1024-byte	1355.9	1.31
512-byte	750.5	0.89
256-byte	400.0	0.94
128-byte	210.8	1.12
64-byte	106.9	0.91

Table 6: Throughput and average latency of EnclaveVPN

Table 6 summarizes the aggregated throughput (Mbps) and average latency (ms) of EnclaveVPN. The actual numbers in terms of Megabits per second (Mbps) and milliseconds (ms) were calculated from the results depicted in Figure 7. The aggregated throughput of EnclaveVPN is the maximum throughput regardless of the established VPN connections.

5.3.2 Detail in performance of enclave data plane. Packet per second influences the performance of enclave data plane. The maximum theoretical rate for 64-byte packets is approximately 1.2M packet per second for the 1Gbps link. We calculated the consumed CPU cycles as shown in Table 7 using the benchmark results (packet per second) in Section 5.3.1. We refer to the CPU cycles (2.4GHz) where the prototype is implemented. The theoretical cycle means the maximum available cycles for the theoretical packet per second.

Each 64-byte packet should be processed within approximately 1,958 cycles. However, 8,000+ cycles are necessary only for an enclave transition per a single packet. Thus, EnclaveVPN can expect 24.48%, 39.84%, and 70.56% performance of the line speed at most for 64/128/256-byte packets.

Packet size	Theoretical	Native	EnclaveVPN
1420-byte	27,994	29,227	31,431
1280-byte	25,306	26,770	30,644
1024-byte	20,390	21,860	28,999
512-byte	10,560	12,026	26,195
256-byte	5,645	7,105	24,576
128-byte	3,187	4,646	23,312
64-byte	1,958	3,420	22,998

Table 7: CPU cycles per a single packet

5.3.3 Overhead of In-Enclave Forwarding to tenant's VMs. In-Enclave Forwarding induces the overhead to tenant's VMs by the implementation overhead of library OSes or SGX containers and the attestation overhead for receiving an ESP SA. However, the SGX attestation for receiving the ESP SA is not a frequent event because rekeying occurs in terms of time or traffic volume after the initial ESP SA is delivered.

In-Enclave Forwarding mandates the VPN connections within a cloud network. According to [43] encryption on data in transit is strongly recommended, and most cloud service providers currently support packet encryption (e.g., TLS, IPsec) within the cloud network by default or by user configuration. Thus, the packet encryption itself within the cloud network is not a huge overhead to the tenant's VMs in terms of the data plane performance. Moreover, In-Enclave Forwarding does not introduce any additional overhead to the tenant's VMs compared to the usual IPsec operations because In-Enclave Forwarding operates only in EnclaveVPN, not in the tenant's VMs. Hence, the IPsec implementations in the tenant's VMs handle the receiving of ESP packets as usual.

5.4 Analysis

We evaluate EnclaveVPN from the perspective of the goals defined in Section 3.3 (denoted by **G1-G4** as below and summarized in Table 8.

Goals	Countermeasures
[G1] Secrecy of crypto keys	Use EnclaveIKE and EnclaveESP to store crypto keys and execute crypto operations
[G2] Protection of packets within and to/from a cloud network	Support In-Enclave Forwarding
[G3] Isolated execution of IPsec gateway	Dedicate EnclaveIKE and EnclaveESP to each tenant
[G4] Feasible IPsec gateway for a cloud VPN	Support Crypto-partitioning, Clear-and-seal, and AD-merging

Table 8: Analysis against goals in Section 3.3.

[**G1**, **G3**] EnclaveVPN delegates EnclaveIKE and EnclaveESP to store cryptographic keys into the EPC and to execute cryptographic operations for IKE and ESP using the keys inside the enclaves. Further, EnclaveVPN provides each tenant with its dedicated enclaves; EnclaveIKE and EnclaveESP. Because SGX supports an isolated execution environment for enclaves and the hardware-based access control mechanism of SGX prevents any illegal access to the EPC, adversaries cannot access keys and the contents of the packets. Moreover, crypto-partitioning reduces the attack surfaces of the IPsec implementation only to the cryptographic operation, which resides in the EPC. SGX prevents the privileged software (e.g., OSes or hypervisors) from accessing the EPC directly. Thus, adversaries cannot impersonate tenants and invade other tenants' data in the EPC, even though the adversaries corrupt the privileged software with known vulnerabilities.

[**G2**] EnclaveVPN supports In-Enclave Forwarding, which promises that only ESP packets travel within and to/from a cloud network, thereby preventing eavesdropping attacks from adversaries (even including malicious insiders).

[**G4**] To save the limited resource (EPC), EnclaveVPN introduces crypto-partitioning and clear-and-seal. Crypto-partitioning for EnclaveIKE and EnclaveESP saves the EPC by up to 56.39% compared to Whole. Clear-and-seal for EnclaveIKE saves the EPC 5.11% more than the case without it. AD-merging increases the data plane performance of EnclaveVPN by 51.8~70.3% more than that for Basic in terms of the throughput. These features also benefit the average latency of approximately 45% at most compared to that of Basic. In addition, the benchmark testing of EnclaveVPN shows more than 87% throughput of Native for 1280-byte and 1420-byte packets. Thus, EnclaveVPN is feasible because EnclaveVPN achieves both efficiencies in the utilization of the limited resource and the practical performance of the data plane while enhancing the security of a cloud VPN.

6 RELATED WORK

6.1 IPsec Gateways

Prior works on IPsec gateways in the cloud include a secure tunnel establishment for user mobility [54] and efficient resource utilization [80]. Lu et al. [54] introduced a secure tunnel establishment by authentication with a key agreement scheme for a roaming user without carrying the same machine in a private cloud. Protego [80] presents an architecture of distributed IPsec gateways for multitenancy in the public cloud. Protego indicates that dedicating IPsec gateways to tenants results in inefficient resource utilization because the gateways consume fixed resources. Thus, Protego divides the gateways into a control plane and a data plane to resolve this inefficiency and shares the control plane among tenants. In addition to SGX utilization, the plane separation in EnclaveVPN differentiates it from one in Protego by introducing the IPsec-specific EPC optimization features.

We further investigate previous works about general IPsec gateways for security [70, 71] and performance [65]. For the cloud deployment, these solutions need further study about the multi-tenant environment, recent trust anchors for the cloud (e.g, SGX, SEV), etc. sVPN [71] proposes a VPN architecture for a trusted platform that utilizes a hypervisor as an isolated execution environment. Using a shared IPsec gateway, sVPN provides multiple isolated environments with dedicated logical IPsec gateways. Sadeghi et al. [70] proposed an IKEv2 extension to exchange attestation data that allows a peer to evaluate the internal state of a remote peer. This extension leverages TPM to calculate the attestation data and one of the standard IKE exchanges to exchange the attestation data. PIPSEA [65] presents a high-performance IPsec gateway using the APU that includes the CPU and GPU in a single chip. PIPSEA uses an APU's heterogeneous architecture to eliminate the data copy overhead between the CPU and GPU and improves GPU utilization via a new packet scheduling.

6.2 SGX-based Network Applications

SGX is a commercial trusted execution environment and addresses security problems in network applications running on commodity hardware such as network middleboxes and NFV [6, 15, 21, 27, 34, 48, 49, 66, 67, 77, 89].

In [48], SGX is presented as a possible solution for providing security and privacy in network applications such as software-defined inter-domain routing, Tor, and middleboxes. SGX-Tor [49] is an approach to enhance the security and privacy of Tor with SGX. SGX-Tor prevents code manipulation, limits information exposed to untrusted parties, and reduces adversaries' power to the network level.

S-NFV [77] utilizes SGX to isolate the state of NFV applications securely (e.g., Snort [69]). Trusted Click [15] presents an integration of SGX into Click [50] for supporting arbitrary NFV applications securely. SafeBricks [66] allows cloud service providers to monitor only encrypted traffic and guarantees the integrity of both traffic and the Network Functions (NFs) by executing the NFs within enclaves.

SGX-Box [34] presents a secure middlebox system along with abstraction and a high-level programming language that supports the secure inspection of encrypted traffic using SGX. LightBox [21] provides a system that supports full-stack protected stateful middleboxes at native speed. EndBox [27] is a scalable system that securely deploys and executes middlebox functions on client machines at the network edge. ShieldBox [89] is a secure middlebox framework for deploying high-performance NFs over untrusted commodity servers. AirBox [6] is a platform that supports fast, scalable, and secure onloading of edge functions (EFs) for device-cloud interactions.

SGX-LKL [67] is an SGX-based runtime that runs unmodified Linux binaries inside an enclave. SGX-LKL uses WireGuard [20], a different VPN protocol that supports host-to-host connections as well as site-to-site connections. SENG [73] is a network gateway that allows firewalls to attribute traffic to an application using attestation-based DTLS channels.

7 DISCUSSION

7.1 Consideration of larger EPC

Larger EPC (e.g., 1TB) with SGX v2.0 in very recent architecture (the 3rd generation Xeon Scalable) [38] may not motivate partitioning of whole IPsec implementations. However, the smaller TCB would enable source code verification whereas the large application would entail more vulnerabilities and manual verification is often infeasible. Thus, partitioned IPsec implementations with smaller TCB would appreciate the EnclaveVPN's crypto-partitioning, AD-merging, and clear-and-seal.

To support the larger EPC, SGX v2.0 introduces a new memory encryption, Total Memory Encryption (TME), instead of MEE [22]. Additionally, SGX v2.0 implements new protection mechanisms to prevent replay attacks and manipulations of data from a malicious enclave inside the EPC [22]. We will refine the experimental results of EnclaveVPN in SGX v2.0 to evaluate and analyze the impact of the new memory encryption and the new protection mechanisms on the performance.

7.2 Limitation and Future Work

We evaluate EnclaveVPN to measure the maximum available efficiency in EPC utilization and data plane performance. To estimate the maximum of these essential metrics, the evaluation does not consider the performance issues that are not directly related to EnclaveVPN. For example, enclave paging and performance isolation across VMs can affect the measurement of the maximum available efficiency.

Enclave paging [56] enables the privileged software to evict EPC pages into the untrusted memory when an enclave exceeds the EPC size. EnclaveVPN does not use the current EPC size limitation (it is much smaller than 93MB [19] or 1TB [38]). In an unlikely case when EnclaveVPN is forced to experience enclave paging, the performance may be affected by the paging overhead. Given that a cloud VPN is a critical service, we expect the highest priority to run inside the EPC. Performance isolation [85] means VMs should not influence the performance of the application in other VMs running on the same physical machine. Investigating the effects of enclave paging and performance isolation across VMs in the cloud would be a good future research topic for enclave usage.

8 CONCLUSION

In this paper, we presented EnclaveVPN, which leverages SGX to enhance the security of the IPsec gateway in a cloud VPN while achieving the optimized EPC utilization and practical performance of the data plane. EnclaveVPN leverages enclaves (EnclaveIKE, EnclaveESP) to manage cryptographic keys and execute cryptographic operations. EnclaveVPN introduces In-Enclave Forwarding to prevent attackers from sniffing packets within and to/from a cloud network. To save the limited resource in SGX (the EPC), EnclaveVPN presents crypto-partitioning and clear-and-seal. The experiment result showed that EnclaveVPN can save up to 62.5% of the EPC and reduce TCB size to ≈88% in comparison with the case where the entire IPsec implementation resides in the EPC of a single SGX machine. Further, EnclaveVPN introduces AD-merging that guarantees a practical performance (≈87%) of the data plane for the non-SGX IPsec gateway against the 1280+ byte packets.

ACKNOWLEDGMENTS

We sincerely thank the anonymous reviewers for their valuable comments and suggestions. KAIST was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT). (No. NRF-2020R1A2C2101134)

REFERENCES

[1] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. 2013. Innovative Technology for CPU Based Attestation and Sealing. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*. Article 13, 7 pages.

[2] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Daniel O'Keeffe, Mark L Stillwell, et al. 2016. SCONE: Secure linux containers with Intel SGX. In *12th USENIX Symp. Operating Systems Design and Implementation*, Vol. 16. 689–703.

[3] Ahmed Osama Fathy Atya, Zhiyun Qian, Srikanth V Krishnamurthy, Thomas La Porta, Patrick McDaniel, and Lisa Marvel. 2017. Malicious co-residency on the cloud: Attacks and defense. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 1–9.

[4] AWS. (n.d.). AWS Managed VPN Connections. https://docs.aws.amazon.com/vpc/latest/userguide/VPC_VPN.html, Accessed: 2023-3-20.

[5] Andrew Baumann, Marcus Peinado, and Galen Hunt. 2015. Shielding Applications from an Untrusted Cloud with Haven. *ACM Transactions on Computer Systems (TOCS)* 33, 3, Article 8 (2015), 26 pages.

[6] Ketan Bhardwaj, Ming-Wei Shih, Pragya Agarwal, Ada Gavrilovska, Taesoo Kim, and Karsten Schwan. 2016. Fast, scalable and secure onloading of edge functions using AirBox. In *Edge Computing (SEC), IEEE/ACM Symposium on*. IEEE, 14–27.

[7] Scott Bradner and Jim McQuaid. 1999. Benchmarking Methodology for Network Interconnect Devices. https://tools.ietf.org/html/rfc2544. Accessed: 2023-3-20.

[8] Ferdinand Brasser, Srdjan Capkun, Alexandra Dmitrienko, Tommaso Frassetto, Kari Kostiainen, and Ahmad-Reza Sadeghi. 2019. DR. SGX: Automated and Adjustable Side-Channel Protection for SGX using Data Location Randomization. In *Proceedings of the 35th Annual Computer Security Applications Conference*. 788–800.

[9] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. 2017. Software Grand Exposure: SGX Cache Attacks Are Practical. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. 12 pages.

[10] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. 2019. SgxPectre: Stealing Intel Secrets from SGX Enclaves Via Speculative Execution. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 142–157.

[11] Guoxing Chen, Wenhao Wang, Tianyu Chen, Sanchuan Chen, Yinqian Zhang, XiaoFeng Wang, Ten-Hwang Lai, and Dongdai Lin. 2018. Racing in Hyperspace: Closing Hyper-Threading Side Channels on SGX with Contrived Data Races. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 178–194.

[12] Sanchuan Chen, Xiaokuan Zhang, Michael K Reiter, and Yinqian Zhang. 2017. Detecting privileged side-channel attacks in shielded execution with Déjá Vu. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. 7–18.

[13] CISCO. (n.d.). Cisco Cloud Services Router 1000V Series. https://www.cisco.com/c/en/us/products/routers/cloud-services-router-1000v-series/index.html, Accessed: 2023-3-20.

[14] Victor Costan, Ilia Lebedev, and Srinivas Devadas. 2016. Sanctum: Minimal hardware extensions for strong software isolation. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 857–874.

[15] Michael Coughlin, Eric Keller, and Eric Wustrow. 2017. Trusted Click: Overcoming Security Issues of NFV in the Cloud. In *Proceedings of the 2017 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 31–36.

[16] Cas Cremers. 2011. Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2. In *European Symposium on Research in Computer Security*. Springer, 315–334.

[17] A Danial. (n.d.). Count lines of code. https://github.com/AlDanial/cloc, Accessed: 2023-3-20.

[18] Whitfield Diffie and Martin Hellman. 1976. New Directions in Cryptography. *IEEE Transaction on Information Theory* 22, 6 (1976), 644–654.

[19] Tu Dinh Ngoc, Bao Bui, Stella Bitchebe, Alain Tchana, Valerio Schiavoni, Pascal Felber, and Daniel Hagimont. 2019. Everything You Should Know About Intel SGX Performance on Virtualized Systems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 1, Article 5 (March 2019), 21 pages.

[20] Jason A Donenfeld. 2017. WireGuard: Next Generation Kernel Network Tunnel.. In *NDSS*. 1–12.

[21] Huayi Duan, Cong Wang, Xingliang Yuan, Yajin Zhou, Qian Wang, and Kui Ren. 2019. LightBox: Full-Stack Protected Stateful Middlebox at Lightning Speed. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2351–2367.

[22] Muhammad El-Hindi, Tobias Ziegler, Matthias Heinrich, Adrian Lutsch, Zheguang Zhao, and Carsten Binnig. 2022. Benchmarking the Second Generation of Intel SGX Hardware. In *Data Management on New Hardware*. 1–8.

[23] Dmitry Evtyushkin, Ryan Riley, Nael CSE Abu-Ghazaleh, Dmitry Ponomarev, et al. 2018. BranchScope: A New Side-Channel Attack on Directional Branch Predictor. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, Vol. 53. ACM, 693–707.

[24] Fast Data Project (FD.io). (n.d.). Github repository for Vector Packet Processing. https://github.com/FDio/vpp, Accessed: 2023-3-20.

[25] Fast Data Project (FD.io). (n.d.). Vector Packet Processing. https://wiki.fd.io/view/VPP, Accessed: 2023-3-20.

[26] Sebastian Gallenmüller, Paul Emmerich, Florian Wohlfart, Daniel Raumer, and Georg Carle. 2015. Comparison of Frameworks for High-Performance Packet IO. In *2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE, 29–38.

[27] David Goltzsche, Signe Rüsch, Manuel Nieke, Sébastien Vaucher, Nico Weichbrodt, Valerio Schiavoni, Pierre-Louis Aublin, Paolo Cosa, Christof Fetzer, Pascal Felber, et al. 2018. EndBox: Scalable Middlebox Functions Using Client-Side Trusted Execution. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 386–397.

[28] Google. 2022. Encryption in Transit in Google Cloud. https://cloud.google.com/security/encryption-in-transit, Accessed: 2023-3-20.

[29] Google. (n.d.). Google Cloud VPN. https://cloud.google.com/vpn/docs/concepts/overview, Accessed: 2023-3-20.

[30] Johannes Götzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Müller. 2017. Cache attacks on Intel SGX. In *Proceedings of the 10th European Workshop on Systems Security*. ACM, 2 pages.

[31] Daniel Gruss, Julian Lettner, Felix Schuster, Olya Ohrimenko, Istvan Haller, and Manuel Costa. 2017. Strong and Efficient Cache Side-Channel Protection using Hardware Transactional Memory. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 217–233.

[32] Shay Gueron. 2016. A Memory Encryption Engine Suitable for General Purpose Processors. https://eprint.iacr.org/2016/204. Accessed: 2023-3-20.

[33] Marcus Hähnel, Weidong Cui, and Marcus Peinado. 2017. High-Resolution Side Channels for Untrusted Operating Systems. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. 299–312.

[34] Juhyeng Han, Seongmin Kim, Jaehyeong Ha, and Dongsu Han. 2017. SGX-Box: Enabling Visibility on Encrypted Traffic Using a Secure Middlebox Module. In *Proceedings of the First Asia-Pacific Workshop on Networking*. ACM, 99–105. https://doi.org/10.1145/3106989.3106994

[35] Dan Harkins, Dave Carrel, et al. 1998. The Internet Key Exchange (IKE). https://tools.ietf.org/html/rfc2409. Accessed: 2023-3-20.

[36] Frederik Hauser, Marco Häberle, Mark Schmidt, and Michael Menth. 2019. P4-IPsec: Implementation of IPsec Gateways in P4 with SDN Control for Host-to-Site Scenarios. *arXiv preprint arXiv:1907.03593* (2019).

[37] Intel Corporation. 2016. SGX Virtualization. https://01.org/intel-software-guard-extensions/sgx-virtualization. Accessed: 2023-3-20.

[38] Intel Corporation. 2021. What Technology Change Enables 1 Terabyte (TB) Enclave Page Cache (EPC) size in 3rd Generation Intel Xeon Scalable Processor Platforms?, 2021. https://www.intel.com/content/www/us/en/support/articles/000059614/software/intel-security-products.html, Accessed: 2023-3-20.

[39] Intel Corporation. (n.d.). Attestation Service for Intel Software Guard Extensions (Intel SGX): API Documentation. https://software.intel.com/sites/default/files/managed/7e/3b/ias-api-spec.pdf. Accessed: 2023-3-20.

[40] Intel Corporation. (n.d.). Intel SGX SSL. https://github.com/intel/intel-sgx-ssl/. Accessed: 2023-3-20.

[41] Intel Corporation. (n.d.). Intel Software Guard Extensions for Linux OS. https://github.com/intel/linux-sgx. Accessed: 2023-3-20.

[42] Intel Corporation. (n.d.). Xeon processors supporting SGX. https://ark.intel.com/content/www/us/en/ark/search/featurefilter.html?productType=873&2_SoftwareGuardExtensions=No. Accessed: 2023-07-13.

[43] Wayne Jansen and Timothy Grance. 2011. SP 800-144: Guidelines on Security and Privacy in Public Cloud Computing. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-144.pdf. Accessed: 2023-3-20.

[44] Vishal Karande, Erick Bauman, Zhiqiang Lin, and Latifur Khan. 2017. SGX-Log: Securing System Logs with SGX. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. 19–30.

[45] C Kaufman, P Hoffman, Y Nir, and P Eronen. 2010. Internet Key Exchange (IKEv2) Protocol. https://tools.ietf.org/html/rfc5996. Accessed: 2023-3-20.

[46] Stephen Kent. 2005. IP Encapsulating Security Payload (ESP). https://tools.ietf.org/html/rfc4303. Accessed: 2023-3-20.

[47] Stephen Kent and Karen Seo. 2005. Security Architecture for the Internet Protocol. https://tools.ietf.org/html/rfc4301. Accessed: 2023-3-20.

[48] Seongmin Kim, Youjung Shin, Jaehyung Ha, Taesoo Kim, and Dongsu Han. 2015. A First Step Towards Leveraging Commodity Trusted Execution Environments for Network Applications. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*. ACM, Article 7, 7:1–7:7 pages.

[49] Seong Min Kim, Juhyeng Han, Jaehyeong Ha, Taesoo Kim, and Dongsu Han. 2017. Enhancing Security and Privacy of Tor's Ecosystem by Using Trusted

Execution Environments. In *NSDI*. 145–161.

[50] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M Frans Kaashoek. 2000. The Click modular router. *ACM Transactions on Computer Systems (TOCS)* 18, 3 (2000), 263–297.

[51] Kubilay Ahmet Küçük, Andrew Paverd, Andrew Martin, N Asokan, Andrew Simpson, and Robin Ankele. 2016. Exploring the use of Intel SGX for Secure Many-Party Applications. In *Proceedings of the 1st Workshop on System Software for Trusted Execution*. 1–6.

[52] Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. 2017. Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing. In *26th USENIX Security Symposium (USENIX Security 17)*. 557–574.

[53] Joshua Lind, Christian Priebe, Divya Muthukumaran, Dan O'Keeffe, Pierre-Louis Aublin, Florian Kelbert, Tobias Reiher, David Goltzsche, David Eyers, Rüdiger Kapitza, et al. 2017. Glamdring: Automatic application partitioning for intel {SGX}. In *2017 {USENIX} Annual Technical Conference ({USENIX}{ATC} 17)*. 285–298.

[54] Yung-Feng Lu and Chin-Fu Kuo. 2013. Robust and Flexible Tunnel Management for Secure Private Cloud. *ACM SIGAPP Applied Computing Review* 13, 1 (2013), 41–50.

[55] Sinisa Matetic, Mansoor Ahmed, Kari Kostiainen, Aritra Dhar, David Sommer, Arthur Gervais, Ari Juels, and Srdjan Capkun. 2017. ROTE: Rollback Protection for Trusted Execution. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*.

[56] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. 2013. Innovative Instructions and Software Model for Isolated Execution. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*. ACM, Article 10, 10:1–10:8 pages.

[57] Jinli Meng, Xinming Chen, Zhen Chen, Chuang Lin, Beipeng Mu, and Lingyun Ruan. 2010. Towards High-Performance IPsec on Cavium OCTEON Platform. In *International Conference on Trusted Systems*. Springer, 37–46.

[58] Microsoft. (n.d.). Azure VPN Gateway. https://azure.microsoft.com/en-us/services/vpn-gateway/, Accessed: 2023-3-20.

[59] Microsoft. (n.d.). Linux Virtual Machines Pricing in Microsoft Azure. https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/, Accessed: 2023-3-20.

[60] Saeid Mofrad, Fengwei Zhang, Shiyong Lu, and Weidong Shi. 2018. 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*. 1–8.

[61] Ahmad Moghimi, Gorka Irazoqui, and Thomas Eisenbarth. 2017. Cachezoom: How SGX amplifies the power of cache attacks. In *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 69–90.

[62] Juniper Networks. (n.d.). Juniper vSRX Virtual Firewall. https://www.juniper.net/us/en/products-services/security/srx-series/vsrx/, Accessed: 2023-3-20.

[63] Jianyu Niu, Wei Peng, Xiaokuan Zhang, and Yinqian Zhang. 2022. NARRATOR: Secure and Practical State Continuity for Trusted Execution in the Cloud. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2385–2399.

[64] Oleksii Oleksenko, Bohdan Trach, Robert Krahn, Mark Silberstein, and Christof Fetzer. 2018. Varys: Protecting {SGX} Enclaves from Practical Side-Channel Attacks. In *2018 {Usenix} Annual Technical Conference ({USENIX}{ATC} 18)*. 227–240.

[65] Jungho Park, Wookeun Jung, Gangwon Jo, Ilkoo Lee, and Jaejin Lee. 2016. PIPSEA: A Practical IPsec Gateway on Embedded APUs. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1255–1267.

[66] Rishabh Poddar, Chang Lan, Raluca Ada Popa, and Sylvia Ratnasamy. 2018. SafeBricks: Shielding Network Functions in the Cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI'18)*.

[67] Christian Priebe, Divya Muthukumaran, Joshua Lind, Huanzhou Zhu, Shujie Cui, Vasily A Sartakov, and Peter Pietzuch. 2019. SGX-LKL: Securing the Host OS Interface for Trusted Execution. *arXiv preprint arXiv:1908.11143* (2019).

[68] DPDK Project. (n.d.). Data Plane Development Kit. https://www.dpdk.org/, Accessed: 2023-3-20.

[69] Martin Roesch. 1999. Snort: Lightweight intrusion detection for networks. *Lisa* 99, 1, 229–238.

[70] Ahmad-Reza Sadeghi and Steffen Schulz. 2010. Extending IPsec for Efficient Remote Attestation. In *International Conference on Financial Cryptography and Data Security*. Springer, 150–165.

[71] Steffen Schulz and Ahmad-Reza Sadeghi. 2009. Secure VPNs for Trusted Computing Environments. In *International Conference on Trusted Computing*. Springer, 197–216.

[72] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. 2015. VC3: trustworthy data analytics in the cloud using SGX. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 38–54.

[73] Fabian Schwarz and Christian Rossow. 2020. {SENG}, the {SGX-Enforcing} Network Gateway: Authorizing Communication from Shielded Clients. In *29th USENIX Security Symposium (USENIX Security 20)*. 753–770.

[74] Michael Schwarz, Moritz Lipp, Daniel Moghimi, Jo Van Bulck, Julian Stecklina, Thomas Prescher, and Daniel Gruss. 2019. ZombieLoad: Cross-privilege-boundary data sampling. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 753–768.

[75] Michael Schwarz, Samuel Weiser, Daniel Gruss, Clémentine Maurice, and Stefan Mangard. 2017. Malware Guard Extension: Using SGX to conceal Cache Attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 3–24.

[76] Jaebaek Seo, Byoungyoung Lee, Seong Min Kim, Ming-Wei Shih, Insik Shin, Dongsu Han, and Taesoo Kim. 2017. SGX-Shield: Enabling Address Space Layout Randomization for SGX Programs. In *NDSS*.

[77] Ming-Wei Shih, Mohan Kumar, Taesoo Kim, and Ada Gavrilovska. 2016. S-NFV: Securing NFV states by using SGX. In *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 45–48.

[78] Shweta Shinde, Zheng Leong Chua, Viswesh Narayanan, and Prateek Saxena. 2016. Preventing page faults from telling your secrets. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. 317–328.

[79] Sushrut Shringarputale, Patrick McDaniel, Kevin Butler, and Thomas La Porta. 2020. Co-residency attacks on containers are real. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*. 53–66.

[80] Jeongseok Son, Yongqiang Xiong, Kun Tan, Paul Wang, Ze Gan, and Sue Moon. 2017. Protego: Cloud-Scale Multitenant IPsec Gateway. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. USENIX Association, 473–485.

[81] Spirent. 2020. Spirent TestCenter: Is it possible to run an RFC2544 throughput test on STC for IPSec? https://support.spirent.com/SC_KnowledgeView?Id=FAQ19103, Accessed: 2022-03-24.

[82] Raoul Strackx, Bart Jacobs, and Frank Piessens. 2014. ICE: A Passive, High-Speed, State-Continuity Scheme. In *Proceedings of the 30th Annual Computer Security Applications Conference*. 106–115.

[83] strongSwan Team. (n.d.). strongSwan. https://www.strongswan.org/, Accessed: 2023-3-20.

[84] TCG. (n.d.). TPM Main Specification. https://trustedcomputinggroup.org/resource/tpm-main-specification/. Accessed: 2023-3-20.

[85] Alain Tchana, Bao Bui, Boris Teabe, Vlad Nitu, and Daniel Hagimont. 2016. Mitigating performance unpredictability in the IaaS using the Kyoto principle. In *Proceedings of the 17th International Middleware Conference*. 1–10.

[86] Keysight Technologies. 2009. Agilent N2X. https://about.keysight.com/en/newsroom/imagelibrary/2009/13may-em09093/, Accessed: 2023-3-20.

[87] Hongliang Tian, Qiong Zhang, Shoumeng Yan, Alex Rudnitsky, Liron Shacham, Ron Yariv, and Noam Milshten. 2018. Switchless Calls Made Practical in Intel SGX. In *Proceedings of the 3rd Workshop on System Software for Trusted Execution*. ACM, 22–27.

[88] Hongliang Tian, Yong Zhang, Chunxiao Xing, and Shoumeng Yan. 2017. SGXKernel: A Library Operating System Optimized for Intel SGX. In *Proceedings of the Computing Frontiers Conference*. ACM, New York, NY, USA, 35–44. https://doi.org/10.1145/3075564.3075572

[89] Bohdan Trach, Alfred Krohmer, Franz Gregor, Sergei Arnautov, Pramod Bhatotia, and Christof Fetzer. 2018. ShieldBox: Secure Middleboxes using Shielded Execution. In *Proceedings of the Symposium on SDN Research*. ACM, 2.

[90] Chia-Che Tsai, Donald E Porter, and Mona Vij. 2017. Graphene-SGX: A practical library OS for unmodified applications on SGX. In *2017 USENIX Annual Technical Conference (USENIX ATC)*.

[91] Markku Vajaranta, Arto Oinonen, Timo D Hämäläinen, Vili Viitamäki, Jouni Markunmäki, and Ari Kulmala. 2019. Feasibility of FPGA accelerated IPsec on cloud. *Microprocessors and Microsystems* 71 (2019), 102861.

[92] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 991–1008.

[93] Jo Van Bulck, Nico Weichbrodt, Rüdiger Kapitza, Frank Piessens, and Raoul Strackx. 2017. Telling Your Secrets without Page Faults: Stealthy Page Table-Based Attacks on Enclaved Execution. In *26th USENIX Security Symposium (USENIX Security 17)*. 1041–1056.

[94] Common Vulnerabilities and Exposures (CVE). 2010. CVE-2010-0430. https://www.cvedetails.com/cve/CVE-2010-0430/, Accessed: 2023-3-20.

[95] Common Vulnerabilities and Exposures (CVE). 2015. CVE-2015-3340. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3340, Accessed: 2023-3-20.

[96] Common Vulnerabilities and Exposures (CVE). 2015. CVE-2015-6385. https://www.cvedetails.com/cve/CVE-2015-6385/, Accessed: 2023-3-20.

[97] Common Vulnerabilities and Exposures (CVE). 2017. CVE-2017-2341. https://www.cvedetails.com/cve/CVE-2017-2341/, Accessed: 2023-3-20.

[98] Common Vulnerabilities and Exposures (CVE). 2018. CVE-2018-0053. https://www.cvedetails.com/cve/CVE-2018-0053/, Accessed: 2023-3-20.

[99] Common Vulnerabilities and Exposures (CVE). 2019. CVE-2019-17346. https://www.cvedetails.com/cve/CVE-2019-17346/, Accessed: 2023-3-20.

[100] Juan Wang, Zhi Hong, Yuhan Zhang, and Yier Jin. 2017. Enabling Security-Enhanced Attestation with Intel SGX for Remote Terminal and IoT. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 1 (2017), 88–96.

[101] Nico Weichbrodt, Pierre-Louis Aublin, and Rüdiger Kapitza. 2018. Sgx-perf: A Performance Analysis Tool for Intel SGX Enclaves. In *Proceedings of the 19th International Middleware Conference (Middleware '18)*. ACM, 201–213.

[102] Nico Weichbrodt, Anil Kurmus, Peter Pietzuch, and Rüdiger Kapitza. 2016. AsyncShock: Exploiting synchronisation bugs in Intel SGX enclaves. In *European Symposium on Research in Computer Security*. Springer, 440–457.

[103] Ofir Weisse, Valeria Bertacco, and Todd Austin. 2017. Regaining Lost Cycles with HotCalls: A Fast Interface for SGX Secure Enclaves. *ACM SIGARCH Computer Architecture News* 45, 2 (2017), 81–93.

[104] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. 2015. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 640–656.