Toward Scaling Hardware Security Module for Emerging Cloud Services

Juhyeng Han*, Seongmin Kim*, Taesoo Kim†, Dongsu Han

KAIST †Georgia Tech

ABSTRACT

The hardware security module (HSM) has been used as a root of trust for various key management services. At the same time, rapid innovation in emerging industries, such as container-based microservices, accelerates demands for scaling security services. However, current on-premises HSMs have limitations to afford such demands due to the restricted scalability and high price of deployment. This paper presents ScaleTrust, a framework for scaling security services by utilizing HSMs with SGX-based key management service (KMS) in a collaborative, yet secure manner. Based on a hierarchical model, we design a cryptographic workload distribution between HSMs and KMS enclaves to achieve both the elasticity of cloud software and the hardware-based security of HSM appliances. We demonstrate practical implications of ScaleTrust using two case studies that require secure cryptographic operations with low latency and high scalability.

CCS CONCEPTS

• **Security and privacy → Systems security**;

1 INTRODUCTION

The hardware security module (HSM) has been utilized as a root of trust to secure numerous cloud applications and network transactions. An HSM protects cryptographic key operations such as signing certificates by supporting secure physical separation for the key management. HSMs have been widely deployed for the online services that require secure key management such as certificate authorities (CAs) in public key infrastructure (PKI) environment [2], e-commerce payment system [22], and DNSSEC [21].

At the same time, rapid innovation in emerging industries, such as container-based microservices, edge computing, and financial technology, accelerates demands for scaling security services. For example, traditional cloud monolithic applications are divided into multiple microservices to achieve high deployability and modifiability [20, 32]. Recent applications and web services take edge-friendly designs in a distributed manner to deal with numerous mobile and IoT transactions for low-latency. This trend dramatically increases both user-to-service transactions and service-to-service transactions which need to be cryptographically secured. This, however, introduces a heavy burden on HSMs.

The fundamental problem of an on-premises HSM is its limited scalability [16, 18]. Because operations related to secret keys are only conducted by the dedicated hardware, the throughput of cryptographic requests is affected by the resource utilization of an HSM and its network status, which can be a bottleneck of large-scale services. However, deploying HSMs in large-scale introduces a significant capital investment. Even a cloud-based HSM [2, 5] is quite costly: AWS Cloud HSM [2] and IBM Cloud HSM [5] charge $1,168, and $1,250 per month, respectively.

An alternative is to use a software-based key management service (KMS) by leveraging a commodity trusted execution environment (TEE) [16, 18]. Intel SGX [27] offers an isolated execution inside a secure enclave with native performance. Because the SGX functionality is available on x86 CPUs, it is cost-effective and easy to deploy at large scale, compared to the HSM. However, this approach does not provide hardware separation required by regulation [24], unlike the HSM. For example, Canadian and U.S. federal governments have regulations to validate compliance with the FIPS 140-2 standard [14] by the law. The FIPS certification above level 2 [29] mandates physical separation and enterprises often use on-premises HSMs to meet the requirement.

However, a naïve combination of HSMs and TEE-based KMS does not achieve both service scalability and security. For example, handling frequent private key operations in an HSM might introduce heavy computations, which makes

*The first two authors contributed equally to this work.

SysTEX '19, October 27, 2019, Huntsville, ON, Canada

ACM ISBN 978-1-4503-6888-9/19/10…$15.00
https://doi.org/10.1145/3342559.3365335

Level	Description	Validated Technology Examples
Level 1	Requires production-grade equipment with externally tested algorithms including DES and AES	WolfSSL-SGX [13]
Level 2	Satisfies the requirement for physical tamper-evidence and role-based authentication	Self-Encrypting Drive (SED) storage [23]
Level 3	Satisfies the requirement for physical tamper-evidence and identity-based authentication	on-premise HSMs [11]
Level 4	Has the ability to delete contents if it detects environmental attack such as supercooling	on-premise HSMs [6]

Table 1: Four levels of FIPS 140-2 standard certification.

Product	Vendor	Signing speed (RSA-2048)	Price (USD)	tps/$
SafeNet Luna SA A790	Gemalto	10,000 tps	$29,900	0.33
nShield F2 6000+	nCipher	3,000 tps	$12,560	0.24
Xeon E3-1280 v6	Intel	4,200 tps	$500	8.4
Xeon E3-1280 v6 (SGX)*	Intel	3,600 tps	$500	7.2

Table 2: Specifications of commercial HSM products and x86 CPUs in marketplace (*See §5).

the HSM becoming a performance bottleneck. Also, preserving chain-of-trust from an HSM to end users over untrusted channels is non-trivial due to the lack of validation mechanism between the HSM and the TEE-based KMS instances. Therefore, a careful design considering both application characteristics and security is required to utilize TEE with HSMs.

This paper explores practical implications of utilizing HSMs with TEE-based KMS in a collaborative, yet secure manner. We propose a workload distribution methodology between HSMs and TEE-based KMS to achieve both the elasticity of cloud software and the hardware-based security of an HSM appliance. To achieve end-to-end security between an HSM and end users, we design a secure key bootstrapping mechanism and leverage SGX remote attestation for trustworthy deployment of KMS instances. Finally, we explore the division of workload between HSMs and SGX enclaves with new emerging applications in the microservice architecture and perform preliminary evaluation of JSON web token authentication scenario.

2 BACKGROUND

HSM vs Traditional Software KMS. HSM appliances are designed and certified to provide the highest level of physical security. The hardware is intrusion-resistant and tamper-evident against physical attacks. Because cryptographic operations and key management are performed within isolated hardware, an HSM has been used as a root of trust for various key management services [2, 21, 22]. Modern HSM devices are powerful enough to operate thousands of cryptographic operations per second [11, 28] and satisfy FIPS 140-2 [14] level 3 or 4 standard shown in Table 1.

One core limitation of an HSM is the overhead of provisioning an on-premises HSM due to its high price. Table 2 shows the prices and transactions per second (tps) of commercial HSM products that meet FIPS 140-2 level 3 certification. HSMs do not scale well because the performance is restricted by hardware resource and network bandwidth. To scale up the service, significant investments are needed for deploying more HSMs. Recent network trends, such as edge computing [33], offload cryptographic operations at the edge, which may instantiate the need for a scalable infrastructure.

Software-based KMS for cloud environments [3, 12] complements the limitations of HSM appliances. It takes the same role with the HSM (e.g., centralized management of digital keys), but more flexible and highly available by providing SDK for application development and integration. Compared to the HSM, software-based KMS scales horizontally to reduce its queuing delay. Despite the above benefits, it cannot guarantee the physical separation of digital keys which is required to meet FIPS 140-2 [14] level 3 or higher standard. In fact, commodity software-based KMS solutions [3, 12] support HSM extensions as options to satisfy the demands to use on-premises HSMs for regulatory compliance policies [24].

KMS utilizing TEE. Several market products target the cloud environment and deliver key management service based on commodity TEE technology [16, 18]. Fortanix released self-defending key management service (SDKMS) [16] to achieve scalable data protection, secured by SGX enclaves. To protect against enclave code vulnerabilities, the encryption module of SDKMS is implemented in Rust. Intel extends Barbican that provides REST APIs for secure key management in the Openstack environment to support SGX crypto plugins [18]. It offers a secure multi-user key distribution among third-party SGX enclaves by leveraging remote attestation. Several studies leverage ARM TrustZone, a commoditized TEE for embedded and mobile devices for credential-based authentication [25] and key management service [26]. Existing approaches have focused on utilizing the TEE-based crypto plugins as an alternative of HSMs to address the scalability issue, which means that they do not preserve the property of physical separation that on-premises HSMs give.

3 SYSTEM DESIGN

This section presents our framework called ScaleTrust, designed to scale out cryptographic operations by utilizing HSMs and SGX-enabled KMS in a collaborative, yet secure manner. To achieve our design goal, a careful workload distribution between HSMs and SGX-enabled key management modules is required. First, we would like to avoid using HSMs for frequent cryptographic computations. For example, storing all the session keys in HSMs may incur performance

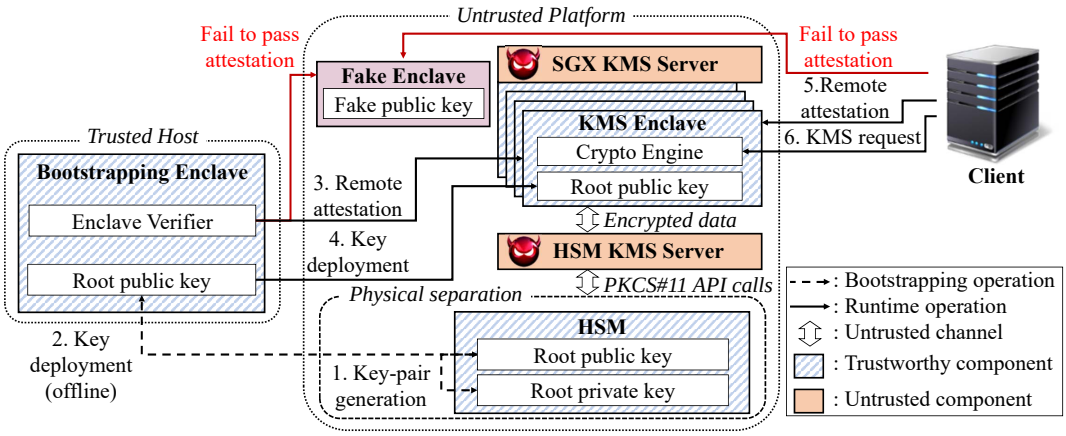

Figure 1: Design overview of ScaleTrust and its threat model.

bottleneck when many clients establish short-lived, but frequent TLS connections. This might fail to satisfy service level objectives (SLOs), such as the response time. Second, we would like to store master secrets on hardware-isolated HSMs. SGX-enabled key management modules cannot be the last line to store master secrets because they cannot preserve physical tamper-evidence, unlike the HSMs.

Threat model and deployment assumption. We assume a powerful adversary who can control any software components in untrusted platform, including OS and hypervisor. HSM hardware itself protects secrets by isolating them in separated hardware, but a communication channel with external hosts is untrusted, as an attacker takes full control over the host OS and libraries. We assume that an HSM's root public key deployment to a trusted host is trustworthy during the system bootstrapping time, and the HSM only accepts requests encrypted with the key (See §3.2 for more details). Also, we typically trust SGX enclaves, but our system does not fully rely on the enclaves for key management because they do not provide physical separation for storing keys. Finally, we assume that the SGX-enabled instances for key management service are located between an HSM and end users, following the generic deployment scenario of software-based KMS [3, 12]. Note that we focus on guaranteeing forward secrecy after key deployments, and achieving post-compromise security is beyond the scope of this paper.

3.1 Introducing Hierarchy for Scaling

To satisfy the above two requirements, we take a hierarchical approach inspired by modern CA systems. Figure 1 shows the overall architecture of ScaleTrust. ScaleTrust consists of three main components: an HSM KMS server equipped with HSM devices, a bootstrapping enclave, and an SGX KMS server that launches multiple KMS enclaves. The HSM plays the role of a root of trust and handles security-sensitive cryptographic requests that should be performed in the physically separated hardware. The bootstrapping enclave is a helper enclave that bridges trust from the HSM to the KMS enclaves through secure bootstrapping (See §3.2). Lastly, the SGX KMS server generates KMS enclaves that handle cryptographic requests from clients. Based on the hierarchical model, we explore a new design space of utilizing both HSMs and SGX-enabled KMS to scale out security services.

HSM virtualization. Our framework supports fully functional, but flexible HSM virtualization to accommodate multi-tenancy and to increase workload density. Although some commodity high-end HSM products support HSM partitioning [11], they only provide a small number of partitions with limited hardware resource (e.g., 100 partitions with 32MB memory [11]). Since SGX preserves memory protection between enclaves running in the same host, each KMS enclave is cryptographically isolated and acts as an independent HSM with its own access controls and security policies in richer resource environments.

Flexible deployability and usability. The software architecture of ScaleTrust accommodates various deployment scenarios depending on the business model. It would be applicable for various security services that rely on an SGX-enabled key management service provided by not only a single public cloud provider, but also a hybrid cloud. For example, a service provider can utilize heterogeneous KMS enclaves provided by multiple cloud providers such as Microsoft OpenEnclave [9] and Google Asylo [1]. ScaleTrust also offers better usability of key management service as the KMS enclave enables a service provider to provide crypto-agility and to utilize various software cryptographic schemes (e.g., attribute-based encryption), not supported by a legacy HSM.

Workload distribution between SGX enclaves and HSMs. ScaleTrust reduces the burden of HSMs by delegating frequent cryptographic requests to multiple KMS enclaves in a distributed manner. The HSM focuses on dealing with root

key operations such as key derivation, rotation, and revocation on the derived keys, and the KMS enclave handles user requests for cryptographic operations such as signing, encryption, decryption with the derived keys. During the execution, the bootstrapping enclave monitors the workload of each KMS enclave and launches additional KMS enclaves, if necessary. In addition, KMS enclaves can be deployed to different platforms to reduce the tail latency of distributed clients or prevent the single-point-of-failure of a KMS server. This takes advantages of software-based key management which is on-demand, fault-tolerant, and highly scalable.

3.2 Security Guarantees in ScaleTrust

ScaleTrust provides better security for end users to explicitly spell out which SGX instances they trust in practice, without breaking the security guarantee of HSMs. Under our threat model, an HSM does not have knowledge of which enclaves are reliable due to the lack of validation mechanism against SGX instances. For example, an attacker can launch an emulated, but potentially malicious enclave to acquire a derived key from an HSM, similar to man-in-the-middle attacks. Note that such attack scenario is still valid in the traditional HSM deployment once a communication channel between an end user and an HSM is compromised. To address this problem, we design a secure key bootstrapping mechanism and leverage SGX remote attestation primitive.

Secure bootstrapping. To establish secure channels between an HSM and KMS enclaves, ScaleTrust performs a secure bootstrapping procedure for sharing the HSM's root public key with the enclaves. First, a service provider generates a root asymmetric key pair in the HSM and retrieves the root public key only. Next, the service provider embeds the root public key in a bootstrapping enclave code as fixed data and launches the enclave in the trusted host. Note that the key deployment process must be done in a secure way (e.g., offline distribution). During the initialization of KMS enclaves, the bootstrapping enclave deploys the root public key only to the valid KMS enclaves that pass SGX remote attestation. Since the deployed key is only shared between the HSM and the enclaves, it enables them to perform mutual attestation for mitigating remote attacks on HSMs [15]. Finally, each KMS enclave establishes a secure channel with the HSM by encrypting session data using the root public key and decrypting it in the HSM using the root private key.

Attestation on SGX instances. ScaleTrust performs two types of remote attestation to achieve end-to-end security guarantee between an HSM and end users. First, the bootstrapping enclave verifies the integrity of KMS enclaves through remote attestation for a secure launch. This guarantees the root public key used for building secure channels between the HSM and the KMS enclaves is correctly provisioned. Another remote attestation procedure is done by clients who would request cryptographic operations to KMS enclaves to validate whether the launched enclaves are reliable. This attestation chain ensures that clients can sort out which SGX instances are trustworthy.

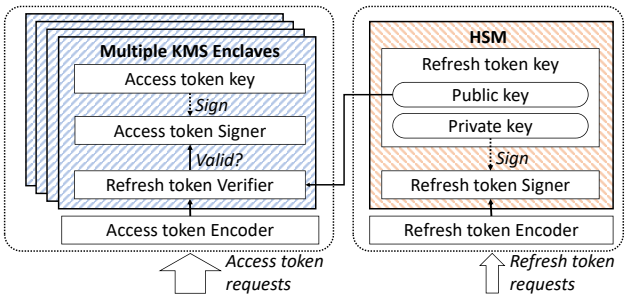

Figure 2: Applying ScaleTrust to JWT

4 APPLICATION CASE STUDIES

Achieving both elasticity of the TEE-based key management and the strong physical separation guaranteed by HSM appliances, our system accommodates diverse use cases. We present case studies for two different applications from the real world, both of which require frequent cryptographic operations with low latency and high scalability.

4.1 JSON Web Token (JWT) Management

JSON Web Token (JWT) [17] supports token-based stateless web authentication and is widely deployed to various microservices [31]. Basically, the real world authentication services provide two types of JWT: 1) an access token for authenticating a user and 2) a refresh token for obtaining a new access token when the previous access token is expired. An access token has a shorter lifetime (few hours) than refresh token lifetime (more than a week or permanent). For ensuring the integrity of JWT contents, JWT requires a digital signature using cryptographic operations such as RSA signing. However, as the number of microservices and users increases, the burden of authentication services will also increase, which might incur performance degradation.

ScaleTrust provides a cost-effective solution by workload distribution between HSMs and KMS enclaves, considering the JWT characteristics. Figure 2 shows ScaleTrust design for JWT signing. To build JWT system for a microservice, a ScaleTrust HSM generates a key pair for signing a refresh token and sends the generated public key to the target KMS enclaves which are allocated to create access tokens. Then, the KMS enclave checks the validity of the incoming user's refresh token using the provided public key. If the refresh token is valid, the KMS enclave generates an asymmetric key pair or a secret for signing a new access token. Finally, the KMS enclave shares the generated public key or the

secret with the microservice. Note that refresh token signing requests only occur when a new microservice is created, or the previous refresh token is expired, which are much less frequent than access token signing requests made by the microservice users. Therefore, ScaleTrust reduces the burden of an HSM and supports trustworthy signing chain from the refresh token to the access token.

4.2 Authentication between Microservices

Workload distribution of ScaleTrust is applicable for scaling out authentication of benign microservice instances to protect service-to-service communications. To achieve a secure channel between microservices, a recent container orchestration solution [8] utilizes Mutual Transport Layer Security (MTLS). Before establishing MTLS session, microservices rely on a CA service to retrieve certificates. The CA issues a certificate when a microservice successfully verifies the provisioned cryptographic hash of the self-signed root certificate and a shared secret for establishing MTLS session. This means that the CA has to issue a certificate for each microservice. However, compared to monolithic legacy applications, a large number of microservice instances are dynamically generated and terminated with short lifetime, which makes authentication processes become a performance bottleneck. Our system design can be adopted to solve the problem; KMS enclaves deal with issuing short-lived credentials for MTLS channel while HSMs manage the root certificate.

5 PRELIMINARY EVALUATION

Our preliminary evaluation answers below two questions:

- Does ScaleTrust enhance the end-to-end latency by scaling out multiple enclaves?
- Does ScaleTrust provide cost-effective scalability compared to traditional HSM appliances?

We evaluate ScaleTrust with the JWT authentication scenario in §4.1. We use two machines (Quad-core Intel Xeon E3-1280V6 3.90 GHz CPU) for a JWT issue server and a client, respectively. We use SoftHSM [10], a software emulation framework for an HSM, and Intel SGX Linux SDK 2.5 [7] for SGX implementation. For JWT access token signing, each enclave and HSM performs the same SHA-256 with RSA-2048 signing operations. The client running on a different host is directly connected with the JWT issue server through a 10 Gbps LAN cable. We set 1 core pinning for SoftHSM to emulate a real HSM appliance which has restricted resource and for each KMS enclave to isolate CPU resource.

Latency improvement with scaling. We measure the response time of each JWT issue server that utilizes either ScaleTrust KMS or a stand-alone HSM. For each run, a microservice client generates 1 k concurrent and total 10 k access token requests to the server, and we increase the number of ScaleTrust KMS enclaves as 1, 2 and 4. Figure 3 shows a CDF of each result. Scaling out more enclaves provides faster response time by processing RSA signing workloads in parallel. The JWT issue server with 4 SGX enclaves KMS reduces 95 th-percentile response time by 56.3 % compared to the stand-alone HSM KMS. This means that ScaleTrust offers scalability on large-scale cryptographic workloads for improving end-to-end performance.

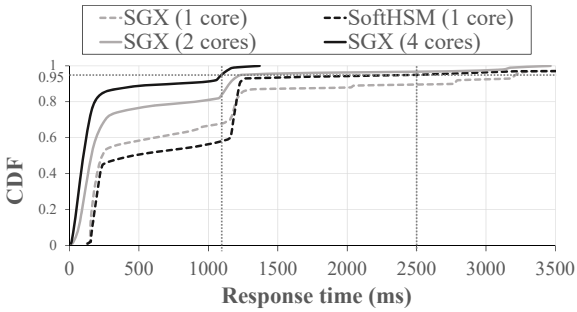

Figure 3: Response time of JWT issue server.

Cost-effective scaling. We evaluate the cost effectiveness of a commercial HSM and our KMS enclave by comparing the signing throughput. A single KMS enclave on Xeon E3-1280V6 CPU ($500 cost) performs 3,600 tps for SHA-256 with RSA-2048 signing operations. On the other hand, a Luna SA A790 HSM ($29,900 cost) is capable to process 10,000 tps for RSA-2048 signing (See Table 2). We compare the tps per dollar, and the result shows that the KMS enclave gives 7.2 tps/$, while the HSM has 0.33 tps/$. Also, cloud-based HSMs such as IBM [5], Amazon [2], and Google [4] charge significant payment to a service provider for each crypto operation or per month basis (e.g., $1,250 for IBM Cloud HSM service). Therefore, ScaleTrust is cost effective compared to existing on-premises HSMs or cloud-based solutions.

6 DISCUSSION

Physical isolation with Intel VCA. Recently, Intel introduces Visual Compute Accelerator (VCA) [19], a PCI-attached co-processor (or accelerator) that allows cloud providers to extend the capability of graphics computation. Although VCA accelerators are originally designed for graphics computation, it is shipped with three Xeon E3 processors that provide the SGX feature, paving a promising way to scale SGX applications on the legacy cloud providers having no modern CPUs [19]. Since VCA is designed as a co-processor, it provides a high-bandwidth, low-latency channel among the cores inside the accelerator via shared memory, and to the host cores via the PICe interface. Such physical separation and the high speed communication channel are desirable primitives to build scalable HSM services securely without

replacing existing host CPUs in the current data center. We believe that Intel VCA introduces new design choices in our software architecture. For example, we can locate the bootstrapping enclave in Intel VCA for reducing latency or utilize Intel VCA as an alternative of HSM thanks to its physical separation. We leave the detailed design and systematic evaluation as future work.

Achieving Split Knowledge. ScaleTrust enables a secure implementation of split knowledge. Split knowledge is a method to distribute a cryptographic key to two or more entities in pieces, ensuring that each of them only has a knowledge of its own separated component and cannot derive the original key. In general, service providers need to ensure split knowledge on their payment services, following the PCI requirement 3.6.6 [30]. Based on our system design, we can accommodate split knowledge by allocating each partial key to KMS enclaves, while an HSM securely stores a complete secret key. Nevertheless, someone still worries that subversion of enclaves (e.g., side-channel attacks) in a single platform might lead to the failure of achieving split knowledge. Because ScaleTrust supports multiple-party cloud platforms, a service provider can securely achieve split knowledge by distributing each partial key into heterogeneous KMS enclaves running on different cloud platforms.

7 CONCLUSION

To address the limited scalability of HSMs, this paper explores a new design space to relieve the burden of HSMs by leveraging a commodity TEE technology. We propose ScaleTrust that utilizes HSMs and SGX enclaves in a hierarchical model to improve performance, deployability, and usability, while enhancing security guarantees compared to legacy HSMs. Our case studies show that ScaleTrust can be applied to security services in microservices that frequently request cryptographic operations. Finally, our preliminary evaluation shows that ScaleTrust improves end-to-end latency and achieves cost-effective scaling.

ACKNOWLEDGEMENT

We thank the anonymous reviewers for their valuable feedback. This research was supported in part by Institute of Civil Military Technology Cooperation Center (ICMTC) funded by the Korea government (MOTIE & DAPA) [18-CM-SW-09]. Taesoo Kim was supported in part by NSF awards CNS-1563848, and gifts from Intel.

REFERENCES

[1] Asylo: An Open and Flexible Framework for Enclave Applications. https://asylo.dev.
[2] AWS CloudHSM. https://aws.amazon.com/cloudhsm.
[3] Egnyte Key Management. https://www.egnyte.com/encryption-key-management.html.
[4] Google Cloud HSM. https://cloud.google.com/hsm.
[5] IBM Cloud HSM. https://www.ibm.com/cloud/hardware-security-module.
[6] IBM Systems cryptographic HSMs. https://www.ibm.com/security/cryptocards/hsms.
[7] Intel® Software Guard Extensions SDK for Linux* OS. https://github.com/intel/linux-sgx.
[8] Official Docker v17.06 documentation. Manage swarm security with public key infrastructure [Accessed Aug. 20, 2019].
[9] Open Enclave SDK. https://openenclave.io/sdk.
[10] OpenDNSSEC SoftHSM. https://www.opendnssec.org/softhsm.
[11] SafeNet Hardware Security Modules. https://safenet.gemalto.com/data-encryption/hardware-security-modules-hsms.
[12] Thales eSecurity Key Management. https://www.thalesesecurity.com/products/key-management.
[13] WolfSSL Intel SGX. https://www.wolfssl.com/wolfssl-intel-sgx-fips-140-2.
[14] FIPS PUB 140-2. *Security requirements for cryptographic modules*, 2001.
[15] J.-B. Bédrune et al. Everybody be Cool, This is a Robbery! https://i.blackhat.com/USA-19/Thursday/us-19-Campana-Everybody-Be-Cool-This-Is-A-Robbery.pdf, 2019.
[16] J. G. Beekman et al. Challenges For Scaling Applications Across Enclaves. In *Proc. SysTEX*. ACM, 2017.
[17] J. Bradley et al. JSON web token (JWT). https://tools.ietf.org/html/rfc7519, 2015.
[18] S. Chakrabarti et al. Intel SGX Enabled Key Manager Service with OpenStack Barbican. *arXiv preprint arXiv:1712.07694*, 2017.
[19] S. Chakrabarti et al. Scaling Intel® Software Guard Extensions Applications with Intel® SGX Card. In *Proc. HASP*. ACM, 2019.
[20] L. Chen. Microservices: architecting for continuous delivery and devops. In *Proc. ICSA*. IEEE, 2018.
[21] CloudFlare. The DNSSEC Root Signing Ceremony. https://www.cloudflare.com/dns/dnssec/root-signing-ceremony.
[22] EFTLAB. HSMs in a Payment Industry. https://www.eftlab.com/hsms-in-a-payment-industry.
[23] J. Haswell. SSD Architectures to Ensure Security and Performance. *Flash Memory Summit*, 2016.
[24] F. R. Konkel. The Pentagon isn't ready yet for classified information to be stored off-premise in the cloud. https://www.nextgov.com/emerging-tech/2015/02/dod-wants-physical-separation-classified-data-cloud-now/105753.
[25] D. Liu et al. Veriui: Attested Login for Mobile Devices. In *Proc. HotMobile*, 2014.
[26] S. Luo et al. TZ-KMS: A Secure Key Management Service for Joint Cloud Computing with ARM TrustZone. In *Proc. SOSE*, 2018.
[27] F. McKeen et al. Innovative Instructions and Software Model for Isolated Execution. In *Proc. HASP*, 2013.
[28] R. Stubbs. Turning Cryptography into a Service. https://www.cryptomathic.com/news-events/blog/turning-cryptography-into-a-service-part-1.
[29] Thales eSecurity. What is FIPS 140-2? https://www.thalesesecurity.com/faq/key-secrets-management/what-fips-140-2.
[30] J. Wilder. PCI Requirement 3.6.6 - Using Split Knowledge & Dual Control. https://kirkpatrickprice.com/video/pci-requirement-3-6-6-using-split-knowledge-dual-control/.
[31] E. Wolff. *Microservices: flexible software architecture*. Addison-Wesley Professional, 2016.
[32] T. Yarygina et al. Overcoming Security Challenges in Microservice Architectures. In *Proc. SOSE*. IEEE, 2018.
[33] J. Zhang et al. Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues. *IEEE Access*, 2018.