An Experimental Comparison of Enclave TokenVaults and HSMs for Real-Time Card Tokenization

Vimal Teja Manne *

Posted Date: 29 January 2026

doi: 10.20944/preprints202601.2238.v1

Keywords: tokenization; hardware security module; trusted execution environment; Intel SGX; confidential computing; payment gateway; real time systems

 doi:10.20944/preprints202601.2238.v1

Article

An Experimental Comparison of Enclave Token Vaults and HSMs for Real-Time Card Tokenization

Vimal Teja Manne

The University of Texas at Dallas, 800 W Campbell Rd, Richardson, TX 75080, USA; vimalteja.m@gmail.com

Abstract

Card tokenization in payment gateways is usually anchored on hardware security modules (HSMs) that store keys and execute sensitive cryptographic operations under PCI controls. This model is mature but difficult to scale elastically and not well aligned with cloud native architectures that rely on horizontal growth on commodity servers. Trusted execution environments (TEEs), such as Intel SGX or confidential virtual machines, keep tokenization code and key material inside isolated enclaves while the surrounding gateway stack remains untrusted. This paper designs an enclave based token vault that sits in the real time card authorization path, constructs a matching HSM backed baseline that shares the same key hierarchy, token format, and storage layout, and compares them under trace driven workloads from a public anonymised credit card dataset. The core novelty is a head to head experimental comparison of HSM backed and enclave backed token vaults under a shared workflow and replayed transaction trace, which isolates the effect of the isolation boundary itself. We record end to end authorization latency, sustained throughput, and CPU utilisation for both designs, and study sensitivity to HSM round trip delay and operation mix. In our testbed configuration, the enclave backed vault delivers roughly 45% higher maximum stable throughput than the HSM backed baseline while keeping 99th percentile latency within online authorization budgets, at the cost of higher CPU utilisation on gateway hosts. The comparison highlights where enclave token vaults can approach or surpass HSM based deployments and where certified HSMs still offer stronger assurances or operational advantages, providing guidance for hybrid designs in PCI sensitive payment environments.

Keywords: tokenization; hardware security module; trusted execution environment; Intel SGX; confidential computing; payment gateway; real time systems

1. Introduction

Card tokenization replaces primary account numbers (PANs) with surrogate tokens so that most application databases fall outside strict PCI DSS scope [1–3]. The token vault that maintains the mapping between PANs and tokens is therefore one of the most sensitive components in the gateway and must deliver low latency for online authorizations, sustain peak transaction rates, and satisfy compliance requirements.

Modern payment gateways are increasingly cloud native, multi tenant, and regionally distributed. Operators want to scale capacity elastically while keeping cardholder data protected and audit requirements manageable. Placing traditional HSM clusters close to every region and gateway tier can be operationally and economically difficult, which motivates looking at enclave technology as an isolation boundary that aligns better with cloud scheduling and regional replication.

In many deployments, token vaults are anchored on HSMs that keep cryptographic keys in tamper resistant hardware, expose a constrained cryptographic interface, and are certified under standards such as FIPS 140 [4,5]. This model fits long standing PCI DSS audit practice [6,7] but ties capacity and deployment flexibility to specialised appliances. Throughput is bounded by the HSM interface,

 doi:10.20944/preprints202601.2238.v1

capacity is added in coarse hardware units, and traffic from cloud regions to a central HSM pool incurs extra latency and failure modes [8].

Trusted execution environments such as Intel SGX, ARM TrustZone, and confidential virtual machines provide an alternative isolation boundary using general purpose servers. They encrypt and integrity protect selected execution contexts even when the operating system or hypervisor is untrusted [9–11]. Prior work uses TEEs for key storage and service side cryptography, including in embedded payment settings [12,13], while surveys of TEE vulnerabilities catalog side channel and microarchitectural risks that differ from those examined in classical HSM evaluations [14]. What remains less clear is how an enclave backed token vault compares to an HSM backed vault when both sit in a latency critical authorization path under realistic workloads.

This paper treats the token vault as a component that can be realised either with a classical HSM backed design or with an enclave backed design and compares these options under the same system assumptions and trace driven workload. By fixing a shared key hierarchy, token format, storage layout, and replay trace, the comparison isolates the effect of the isolation boundary and gives operators concrete performance and cost trade offs. This comparative treatment, rather than a single point design, is the main contribution.

The main contributions are:

- A common system and threat model for real time card tokenization that considers both HSM backed and enclave backed token vaults under PCI DSS and FIPS 140 aligned assumptions.
- Matching vault designs for payment gateways, including key hierarchy, tokenization and detokenization workflows, storage layout, and implementation notes that make the prototypes reproducible without depending on a specific vendor stack.
- A comparative experimental study on a realistic replayed credit card trace that quantifies end to end authorization latency, throughput, CPU utilisation, and sensitivity to HSM round trip time and operation mix, together with an illustrative cost model and a discussion of when enclave based vaults can complement rather than replace certified HSMs and confidential VM based deployments.

2. Background and Related Work

Card tokenization reduces PCI scope by replacing PANs with surrogate tokens so that most application databases no longer hold raw card data [1,2]. The token vault becomes a high value target whose design must balance latency, throughput, backup, and audit [6,15].

FIPS 140 specifies requirements and evaluation levels for cryptographic modules, and many payment HSMs are certified under this standard [4,5]. In practice, HSMs terminate secure channels, generate and wrap keys, and perform MAC or PIN related operations for payment applications, aligning with architecture centric PCI DSS approaches [6]. Fixed capacity and network distance to cloud hosted services can make HSMs the throughput or latency bottleneck in multi tenant gateways [8].

TEEs such as Intel SGX and confidential virtual machines provide hardware assisted isolation for selected code and data even if the operating system or hypervisor is compromised [9,11]. Designs usually keep key material and sensitive logic inside enclaves, expose a narrow interface to untrusted components, and rely on attestation to bind enclaves to binaries and configurations [13]. Surveys of TEE attacks show that enclave implementations face side channels and microarchitectural issues that fall outside classical HSM evaluations [14].

Several systems combine HSMs and TEEs. SGX enclaves can front a smaller pool of HSMs and offload cryptographic operations [16,17], SGX accelerator cards host high density enclave workloads in data centres [18], and TEEnder addresses enclave migration with HSM backed protection of enclave state [19]. Tokenization work often assumes a software vault [15] but does not quantify the trade off between HSM anchored and enclave anchored vaults in a payment gateway. To our knowledge, this is the first systematic experimental comparison of HSM and TEE backed vaults under a shared key hierarchy and trace replayed workload.

 doi:10.20944/preprints202601.2238.v1

3. System and Threat Model

3.1. System Overview and Tokenization Workflow

We consider a card payment gateway that processes real time authorization requests from merchant environments to issuer side hosts. A merchant terminal or point of sale device initiates an authorization by sending a message that carries either a PAN or an existing token together with transaction attributes and terminal data. This message reaches the gateway front end over a secure channel, where it is parsed and routed toward the tokenization service and to issuer or acquirer connectors, as shown in Figure 1.

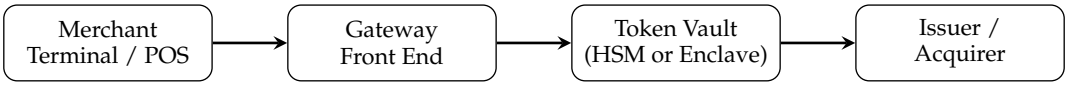

Figure 1. Online authorization path from merchant terminal to issuer. The token vault sits between gateway and issuer and is realised either as an HSM backed or an enclave backed service.

The vault is a logical service that maintains mappings between PANs and tokens for a set of merchants and card ranges [1,2]. When an incoming request carries a PAN and the merchant is configured for tokenization, the gateway calls the vault to obtain or create a token. When a request carries a token and the downstream issuer expects a PAN, the gateway calls the vault to perform detokenization. The vault interacts with a backing store that holds encrypted PAN records, indexes, and metadata such as token domains or expiry flags.

The key hierarchy separates storage encryption from key wrapping. A master key protects key encryption keys (KEKs). Each KEK protects a set of data encryption keys (DEKs) that encrypt PAN fields at rest. The backing store contains records of the form

$$\text{record} = \big(\text{token}, E_{\text{DEK}}(\text{PAN}), \text{metadata}\big),$$

with indexes that map from token to record and from PAN derived values to record.

3.2. Threat Model

We assume a remote adversary interested in recovering PANs from the token vault, tampering with token to PAN mappings, or observing cardholder data in transit. The adversary may compromise software components in the cloud hosted gateway environment, including operating systems, container runtimes, and application processes outside the HSM or enclave, and may be a malicious or curious cloud operator who can inspect disk snapshots, database contents, and internal network traffic.

For the HSM backed design, we assume that the HSM hardware, firmware, and tamper resistance behave according to their FIPS 140 certified security level [4]. The adversary cannot extract keys directly from the HSM without physical compromise. Denial of service attacks that exhaust HSM capacity or disrupt connectivity are possible but are not the focus.

For the enclave backed design, we assume that the CPU and TEE implementation enforce enclave isolation, that attested enclave code corresponds to the intended vault binary, and that the hardware vendor root of trust functions correctly [9,13]. The adversary may fully control the operating system, hypervisor, and cloud management plane and can read and modify data outside enclave memory. Side channel attacks on the TEE, microarchitectural leakage, and physical attacks are not directly mitigated by the prototype; we treat these as residual risks and discuss them in Section 6 following existing surveys [14].

Across both designs we assume that standard symmetric cryptography is not broken, that transport protections such as TLS are correctly configured, and that end user terminals and issuer hosts follow relevant payment network rules.

 doi:10.20944/preprints202601.2238.v1

4. Design and Implementation of Token Vaults

4.1. HSM Backed Token Vault

The HSM backed vault runs as a stateless service in the gateway cluster and delegates key dependent operations to a remote HSM over a mutually authenticated channel [4,5]. Other gateway components reach the vault through an internal RPC interface with methods for tokenization, detokenization, and metadata management.

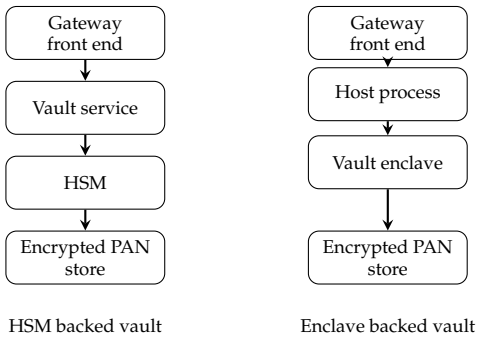

Figure 2. Architecture of HSM backed and enclave backed vaults. Both designs share the same database layout and gateway interface but terminate cryptographic operations either in an HSM or inside an enclave.

On a tokenization request, the service derives a lookup key, probes the backing store, and either returns an existing token or generates a new one. On a miss it invokes the HSM to encrypt the PAN under a DEK, then writes a new record to the store. Detokenization uses the token as lookup key and calls back into the HSM to decrypt the stored PAN.

A master key is created inside the HSM and never leaves in clear form. Under this master key, one or more KEKs are generated and identified by labels. DEKs are either generated in the HSM and exported only as wrapped values or derived from a KEK and record specific diversifier. The HSM enforces key usage flags and domain separation.

4.2. Enclave Backed Token Vault

In the enclave backed design, token vault logic and key material execute inside a TEE instance on a commodity server, while surrounding gateway components remain untrusted [9,13]. The vault is split into enclave code and untrusted host code. The host runs as a normal process, implements the same RPC interface as the HSM backed vault, and handles network operations and access to the backing store. The enclave implements tokenization and detokenization logic, the key hierarchy, and record level processing and runs in protected memory.

When the host receives a tokenization request, it validates basic structure and invokes an enclave call to pass the PAN, token domain, and merchant attributes into enclave memory. Inside the enclave, the code derives the lookup key, constructs database keys, and decides whether to reuse an existing token or create a new one. Encryption and decryption of PAN fields occur entirely inside the enclave; database operations run in the host using ciphertexts and metadata.

The key hierarchy mirrors the HSM structure. A master key is generated when the enclave is provisioned and stored only in enclave memory. To survive restarts, the master key is sealed to disk using the TEE sealing mechanism and unsealed when the enclave starts again on the same platform [9]. KEKs and DEKs are derived from the master key and contextual information using a key derivation function and are reconstructed after unsealing rather than stored outside the enclave.

4.3. Common Storage, Caching, and Implementation Notes

Both designs share a common storage layout. The backing store is a replicated key value database whose records follow the format above, with encrypted PAN fields and cleartext metadata such as token domain, merchant identifier, creation timestamp, and status flags. The primary key is the token. A secondary index maps from a truncated hash of the PAN and token domain to record identifiers to

 doi:10.20944/preprints202601.2238.v1

detect existing entries. For tokenization, the vault computes the index key, probes the index, and either returns an existing token or generates and stores a new one. For detokenization, the vault uses the token as primary key, retrieves the record, and decrypts the PAN.

Both implementations maintain a small in memory cache of recent mappings keyed by derived lookup keys. The cache resides outside the HSM and outside the enclave and holds only tokens and opaque identifiers. PAN fields are protected using AES 256 in an authenticated encryption mode, and DEKs are wrapped under KEKs using a standard AES key wrap construction. The prototypes use a gRPC style RPC layer between the gateway and vault services and a replicated key value store for the backing database. Exact vendor products are deployment specific, but the description above fixes the properties that materially affect performance and lets another team reproduce the setup by matching CPU class, TEE capability, HSM class, and database role rather than particular model numbers.

5. Experimental Setup and Evaluation

5.1. Dataset and Workload Construction

The experiments use a widely adopted anonymised credit card transaction dataset [20] with 284 807 transactions over roughly 48 hours. Each record carries a timestamp, a transaction amount in EUR, and a binary fraud label. Time is measured in seconds since the beginning of data collection, which lets us reconstruct inter arrival gaps for replay. Table 1 summarises key properties.

Table 1. Summary of credit card transaction dataset.

Metric	Value
Total transactions	284 807
Fraudulent transactions	492
Fraud rate	0.173%
Mean amount	88.35 EUR
Median amount	22.00 EUR
99th percentile amount	1 017.97 EUR

Transaction amounts are highly skewed, with many low value purchases and a long tail of larger payments. To characterise workload intensity over time, we aggregate the trace into ten minute bins and compute the number of transactions in each bin. In this series, the minimum bin load is 111 transactions, the maximum is 2 219, and the median is 1 249, with a mean of about 989. For the experiments, we replay the trace in timestamp order through the gateway front end, preserving original inter arrival times so that both vault designs see the same sequence of requests. Where we scale throughput beyond the original trace, we compress inter arrival times uniformly while keeping ordering unchanged, which keeps the transaction mix and diurnal shape constant and makes the replay procedure itself easy to reproduce.

5.2. Platform Configuration and Methodology

The vault services run on identical server class machines in the same cluster. Each server is a modern x86 host with hardware support for the chosen TEE extension, tens of gigabytes of RAM, and a high speed data centre network interface. The HSM backed configuration connects to a network attached payment HSM appliance over a low latency internal network; the enclave backed configuration runs the vault enclave on the local CPU with the same number of cores and memory and uses the same backing store. We omit vendor names and exact model numbers since these are deployment specific and sometimes confidential, but both configurations share the same host hardware profile and database backend so that the only systematic difference is the isolation mechanism used for the vault.

We measure end to end authorization latency from front end receipt to issuer side response, sustained throughput in transactions per second, and CPU utilisation of the vault service and enclave runtime. Latency distributions are reported as medians and high percentiles, throughput is computed

 doi:10.20944/preprints202601.2238.v1

over one minute windows, and CPU utilisation is sampled from operating system statistics. Each reported point corresponds to a single replay run for a given configuration and workload. Running independent repetitions and reporting variance would strengthen the statistical picture, but was not feasible for all combinations under the available hardware budget, so we treat the absence of variance estimates as a limitation and discuss this in Section 6.

We focus on two operating points that occur in the trace: a moderate load phase around 1 000 transactions per second and a high load phase around 2 000. For both phases we extract latency percentiles and resource usage and determine maximum stable throughput by increasing concurrency until 99th percentile latency grows sharply and queues no longer drain between bursts. We also consider a tokenization heavy versus detokenization heavy mix and controlled HSM latency injections during busy windows, which gives a comparative view of how each design behaves under different operation mixes and under degraded HSM connectivity.

5.3. Latency and Throughput

Table 2 summarises end to end authorization latency for both vault designs. At around 1 000 transactions per second the enclave backed vault reduces median latency by about 14 percent compared to the HSM backed vault and improves the 99th percentile by roughly 22 percent. At around 2 000 transactions per second the gap widens in the tail, with the HSM backed vault showing more pronounced queuing delays.

Table 2. End to end authorization latency percentiles (milliseconds).

Configuration	Load	Median	95th	99th
HSM backed	1 000 tps	22.1	35.4	49.7
Enclave backed	1 000 tps	19.0	28.3	38.9
HSM backed	2 000 tps	27.5	45.2	65.1
Enclave backed	2 000 tps	23.4	35.6	52.8

Table 3 reports maximum stable throughput and average CPU utilisation. The HSM backed vault sustains around 2 200 transactions per second before latency increases sharply, with moderate CPU load on the application hosts. The enclave backed vault reaches roughly 3 200 before saturation, at the cost of higher CPU utilisation.

Table 3. Throughput and CPU utilisation at peak stable load.

Configuration	Max stable throughput (tps)	CPU utilisation
HSM backed	2 200	48%
Enclave backed	3 200	78%

For reference, we also implemented a software only vault in which keys are held in process memory and all cryptographic operations run in the gateway service without HSM or enclave isolation. This configuration is not suitable for a compliant deployment, but it provides an upper bound on how the same workflow behaves without hardware protection. In our testbed it sustained about 3 500 transactions per second before saturating CPU, roughly 9 percent higher throughput than the enclave backed vault and about 37 percent higher than the HSM backed vault. The gap between this baseline and the enclave design therefore quantifies the cost of TEE isolation for this workload.

5.4. Tokenization and Detokenization Mix

To see whether the designs behave differently for tokenization and detokenization, we construct a workload variant where 80 percent of requests are detokenization calls and 20 percent are first time tokenizations. Table 4 reports 99th percentile latency per operation type at a combined load of 1 500 transactions per second.

 doi:10.20944/preprints202601.2238.v1

Table 4. 99th percentile latency by operation type at 1 500 transactions per second.

Configuration	Tokenization (ms)	Detokenization (ms)
HSM backed	56.3	44.8
Enclave backed	43.1	36.5

For both designs, detokenization is cheaper than first time tokenization because it avoids database writes. The gap is larger in the HSM backed case, where HSM calls dominate the path for new records. The enclave backed vault narrows this gap because database operations and enclaved cryptography share the same CPU budget.

5.5. Sensitivity to HSM Round Trip Latency

To test sensitivity, we emulate higher round trip latency to the HSM by inserting an artificial delay in the HSM client and replay the trace at a fixed load of 1 800 transactions per second. Table 5 shows the effect on 99th percentile latency. The enclave backed configuration is insensitive to this parameter and is therefore omitted from the table.

Table 5. Effect of HSM round trip latency on 99th percentile latency.

HSM RTT	HSM backed (ms)
0.5 ms	52.6
1.0 ms	59.8
2.0 ms	71.4

The HSM backed vault shows roughly linear sensitivity to added round trip delay at high load. Beyond about one millisecond, the HSM configuration moves close to typical online latency targets, while the enclave design remains stable.

6. Discussion, Limitations, and Conclusion

The measurements indicate that both vault designs can maintain end to end authorization latency within typical online budgets under workloads derived from the public credit card trace and similar variants. The HSM backed vault exhibits higher per operation latency and saturates earlier as load increases, while the enclave backed vault sustains higher throughput at the cost of higher CPU utilisation on gateway hosts. Compared to a software only vault that keeps keys in process memory, the enclave design incurred about a 9 percent throughput penalty, whereas the HSM backed design lagged by about 37 percent, so most of the performance gap is attributable to crossing the HSM boundary rather than to enclaving itself. In our testbed, the enclave backed design delivers about 45 percent higher maximum stable throughput than the HSM backed design while keeping 99th percentile latency within common authorization targets.

Security guarantees differ mainly in the location and strength of the isolation boundary. In the HSM backed design, long term keys and DEKs terminate in a dedicated cryptographic module evaluated under FIPS 140 [4]. Application processes and databases never see keys in the clear, and compromise of the surrounding gateway stack does not directly expose PANs unless the attacker can also subvert the HSM interface. Residual risks lie in misuse of HSM APIs, key management procedures, and denial of service when HSM capacity is exhausted. In practice, many operators already have mature processes and audit evidence for HSM fleets, so migrating entirely away from HSMs may require revisiting compliance playbooks, incident response procedures, and regulator expectations.

In the enclave backed design, the isolation boundary is the TEE instance that hosts the vault logic and key hierarchy. The threat model assumes that the CPU and TEE implementation protect enclave memory from a compromised operating system, hypervisor, or cloud operator [9,13]. Under that assumption, leakage of disk snapshots, database contents, or process memory outside the enclave does not reveal PANs or keys. At the same time, the enclave design inherits residual TEE risks, including

 doi:10.20944/preprints202601.2238.v1

side channel leakage and dependence on firmware and microcode quality [14]. In the prototype we do not enable aggressive side channel mitigations such as dedicated core pinning or cache partitioning across the cluster; quantifying the security benefit and performance impact of such measures, and validating them against concrete attack classes, is a natural next step. A realistic deployment would also need operational monitoring for attestation failures, enclave crashes, and firmware rollouts, which adds a different kind of complexity compared to HSM lifecycle management.

Operators also care about cost per transaction and about how easily capacity can be moved between regions. Table 6 sketches a simple normalised model based on hypothetical but plausible ratios for a deployment that sustains 2.5 million authorizations per hour. The values are illustrative rather than exact but show how the same performance measurements can feed into capacity and cost planning.

Table 6. Illustrative normalised cost per 10^6 authorizations.

Cost component	HSM backed	Enclave backed
HSM hardware and support	1.00	0.20
Gateway compute	0.40	0.85
Database and storage	0.30	0.30
Total (normalised)	1.70	1.35

The HSM backed design pays a higher fixed cost for specialised hardware but uses gateway CPU more sparingly. The enclave backed design removes most HSM cost but consumes more general purpose compute. Depending on cloud pricing, HSM licence models, and utilisation, either design can be more economical, and the model can be instantiated with provider specific numbers in future work. Beyond the two options in this paper, confidential VMs provide a third isolation mechanism where the entire vault process and its dependencies run inside a protected virtual machine boundary. The same replay harness and key hierarchy could support a three way comparison between HSM backed vaults, SGX style enclaves, and confidential VM based vaults, which would give operators a clearer view of the trade offs between these isolation choices.

Several limitations remain. First, the evaluation is confined to a single public dataset and one class of workload derived from it; testing additional traces, longer time horizons, and multi tenant mixes would help assess generality. Second, experiments use a single hardware configuration and single replay run per setting; reporting variance across multiple runs and across HSM vendors would strengthen reproducibility. Third, security discussion focuses on isolation boundaries and residual TEE risks at a qualitative level; quantitative analysis of side channel mitigations, attestation workflows, and enclave hardening would improve confidence. Finally, we do not evaluate multi region replication or failover patterns. Extending the study to a multi region scenario, where token domains, regional data residency rules, and cross region failover all interact with vault placement, would show how quickly each design recovers from regional outages and how well latency targets are preserved for tenants spread across several jurisdictions.

From a practitioner perspective, the measurements also give a concrete recipe for incrementally evolving an existing HSM only gateway. A practical path is to introduce enclave backed vault instances for lower risk portfolios or non production regions first, re use the trace driven methodology from this paper to calibrate capacity and latency under local workloads, and then extend the enclave deployment to additional tenants and regions as operational teams and auditors become comfortable with TEE based controls.

Overall, the study shows that token vaults in real time payment gateways can be realised either with a classical HSM backed design or with an enclave backed design, and that the choice has measurable performance and cost consequences even when the tokenization workflow and key hierarchy remain fixed. The core novelty is the systematic experimental comparison of these two options under a shared replayed trace, which directly quantifies the trade off between physical HSM

 doi:10.20944/preprints202601.2238.v1

assurance and enclave based elasticity and provides a concrete comparative analysis rather than isolated results for a single mechanism. In our configuration, the enclave backed vault can approach or surpass the HSM backed vault in throughput while keeping latency within online authorization targets, at the cost of higher CPU utilisation and reliance on TEE security assumptions. The HSM backed vault offers a stronger physical boundary and alignment with established certification practices but incurs higher per operation cost and lower elasticity. Hybrid designs where an HSM stores or generates the vault master key and enclaves or confidential VMs act as scalable front ends could combine advantages from both sides; implementing and evaluating such hybrid vaults, together with multi region and multi tenant deployments and a fuller treatment of side channel mitigations and variance across runs, is an important direction for future work.

References

1. Williams, B.R. How Tokenization and Encryption Can Enable PCI DSS Compliance. *Information Security Technical Report* **2010**, *15*, 160–165. https://doi.org/10.1016/j.istr.2011.02.005.
2. Stapleton, J.; Poore, R.S. Tokenization and Other Methods of Security for Cardholder Data. *Information Security Journal: A Global Perspective* **2011**, *20*, 91–99. https://doi.org/10.1080/19393555.2011.560923.
3. Liu, J.; Xiao, Y.; Chen, H.; Ozdemir, S.; Dodle, S.; Singh, V. A Survey of Payment Card Industry Data Security Standard. *IEEE Communications Surveys and Tutorials* **2010**, *12*, 287–303. https://doi.org/10.1109/SURV.2010.031810.00083.
4. NIST. FIPS PUB 140-2: Security Requirements for Cryptographic Modules. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, 2002. https://doi.org/10.6028/NIST.FIPS.140-2.
5. Sommerhalder, M. Hardware Security Module. In *Trends in Data Protection and Encryption Technologies*; Mulder, V.; et al., Eds.; Springer: Cham, 2023; pp. 83–87. https://doi.org/10.1007/978-3-031-33386-6_16.
6. Ataya, G. PCI DSS Audit and Compliance. *Information Security Technical Report* **2010**, *15*, 138–144. https://doi.org/10.1016/j.istr.2011.02.004.
7. Peterson, G. From Auditor-Centric to Architecture-Centric: SDLC for PCI DSS. *Information Security Technical Report* **2010**, *15*, 150–153. https://doi.org/10.1016/j.istr.2011.02.003.
8. Bahtiyar, Ş.; Gür, G.; Altay, L. Security Assessment of Payment Systems under PCI DSS Incompatibilities. In Proceedings of the IFIP International Information Security and Privacy Conference (IFIP SEC). Springer, 2014, Vol. 428, *IFIP Advances in Information and Communication Technology*, pp. 395–402. https://doi.org/10.1007/978-3-642-55415-5_33.
9. McKeen, F.; Alexandrovich, I.; Berenzon, A.; Rozas, C.V.; Shafi, H.; Shanbhogue, V.; Savagaonkar, U.R. Innovative Instructions and Software Model for Isolated Execution. In Proceedings of the Proceedings of the 2nd Workshop on Hardware and Architectural Support for Security and Privacy (HASP). ACM, 2013. https://doi.org/10.1145/2487726.2488368.
10. Luo, S.; Hua, Z.; Xia, Y. TZ-KMS: A Secure Key Management Service for Joint Cloud Computing with ARM TrustZone. In Proceedings of the 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE). IEEE, 2018, pp. 180–185. https://doi.org/10.1109/SOSE.2018.00030.
11. Wang, X.; Madaan, D.; et al. Confidential VMs Explained: An Empirical Analysis of AMD SEV-SNP and Intel TDX. In Proceedings of the Proceedings of the ACM SIGMETRICS 2025, 2025. to appear, https://doi.org/10.1145/3700418.
12. Pirker, M.; Slamanig, D. A Framework for Privacy-Preserving Mobile Payment on Security Enhanced ARM TrustZone Platforms. In Proceedings of the Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2012, pp. 1155–1160. https://doi.org/10.1109/TrustCom.2012.28.
13. Brorsson, J.; Nikbakht Bideh, P.; Nilsson, A.; Hell, M. On the Suitability of Using SGX for Secure Key Storage in the Cloud. In *Trust, Privacy and Security in Digital Business*; Springer, 2020; Vol. 12395, *Lecture Notes in Computer Science*, pp. 32–47. https://doi.org/10.1007/978-3-030-58986-8_3.
14. Muñoz, A.; Ríos, R.; Román, R.; López, J. A Survey on the (In)Security of Trusted Execution Environments. *Computers and Security* **2023**, *129*, 103180. https://doi.org/10.1016/j.cose.2023.103180.
15. Thakur, A.; Saxena, A. Improved Vault-Based Tokenization to Boost Vault Lookup Performance. *International Journal of Computer Applications* **2019**, *177*, 24–32. https://doi.org/10.5120/ijca2019919666.

 doi:10.20944/preprints202601.2238.v1

16. Han, J.; Kim, S.; Kim, T.; Han, D. Toward Scaling Hardware Security Module for Emerging Cloud Services. In Proceedings of the Proceedings of the 4th Workshop on System Software for Trusted Execution (SysTEX). ACM, 2019. https://doi.org/10.1145/3342559.3365335.
17. Beekman, J.G.; Porter, D.E. Challenges for Scaling Applications Across Enclaves. In Proceedings of the Proceedings of the 2nd Workshop on System Software for Trusted Execution (SysTEX). ACM, 2017. https://doi.org/10.1145/3152701.3152712.
18. Chakrabarti, S.; Hoekstra, M.; Kuvaiskii, D.; Vij, M. Scaling Intel Software Guard Extensions Applications with Intel SGX Card. In Proceedings of the Proceedings of the 2019 ACM Workshop on Hardware and Architectural Support for Security and Privacy (HASP) co-located with ISCA. ACM, 2019, pp. 6:1–6:9. https://doi.org/10.1145/3337167.3337174.
19. Guerreiro, J.; Moura, R.; Silva, J.N. TEEnder: SGX Enclave Migration Using HSMs. *Computers and Security* **2020**, *96*, 101874. https://doi.org/10.1016/j.cose.2020.101874.
20. Pozzolo, A.D.; Caelen, O.; Johnson, R.A.; Bontempi, G. Calibrating Probability with Undersampling for Unbalanced Classification. https://www.kaggle.com/mlg-ulb/creditcardfraud, 2015. Credit Card Fraud Detection dataset.