Technology Guide

Zero Trust - Harden Data Security with Intel Xeon Processors

Authors

Xiang Wang

Heqing Zhu

Liyan Wang

Yun Lan

Ju Huang

Eric Jones

Hongjun Ni

1 Introduction

Data security is a pillar of zero trust maturity model and a top priority of the IT digital transformation. There are many well-established standards for data security. The National Institute of Standards and Technology (NIST) publishes a specification on baseline security controls for adequate protection of data at rest and enterprises are mandating compliance with all data security recommendations therein.

Encryption is the most common way of protecting data at rest. Disk encryption, as a dominant solution to encrypt data at rest, delivers comprehensive protection and thanks to the in-processor acceleration technologies it is fully transparent to user applications. However, existing encryption for data at rest is not absolutely secure from vulnerabilities, so can be exploited by sophisticated attacks, which can lead to secret keys being obtained if a system is hacked. These vulnerabilities can lead to the loss or manipulation of sensitive data which can cause both reputational, operational, and financial damage to the enterprise. To address the shortcomings of existing disk encryption solutions, organizations are turning to incorporate zero trust architecture. Intel has a broad set of technologies that can be used to optimize zero trust architecture, including capabilities that enhance security with key lifecycle management via confidential computing technology, as well as the crypto performance acceleration.

This paper introduces a full disk encryption solution enhanced by Intel confidential computing technology as well as a new reference design that realizes zero trust security standards. Our solution delivers core features including key management, key protection, and full disk encryption (FDE). This solution achieves security enforcement with confidential computing technology such as Intel® Software Guard Extensions (Intel® SGX) and Intel® Trust Domain Extensions (Intel® TDX) on Intel® Xeon® Scalable processors. This paper provides a reference on how to build a more secure full disk encryption system with the 3rd, 4th and 5th Gen Intel® Xeon® Scalable processors that fulfill the requirements of encryption at rest required by enterprises. The solution is designed to be used by edge server, SASE/XDR server systems, and also network security appliance.

This document is part of the Network Transformation Experience Kits.

Table of Contents

Figures

Tables

Document Revision History

Revision	Date	Description
001	May 2023	Initial release.

1.1 Terminology

Table 1. Terminology

Abbreviation	Description
ACL	Access Control List
AES	Advanced Encryption Standard
AES-NI	Intel® Advanced Encryption Standard New Instructions
CISA	Cybersecurity & Infrastructure Security Agency
CSP	Cloud Service Providers
Enclave	Ring 3 application software running inside the Intel SGX protections
FDE	Full Disk Encryption
HSM	Hardware Security Module
ISA	Instruction Set Architecture
KMS	Key Management System
LUKS	Linux Unified Key Setup
NIST	National Institute of Standards and Technology
PoP	Point of Presence
SaaS	Software as a Service
SASE	Secure Access Service Edge
SIEM	Security Information and Event Management
SED	Self-encrypting Drive
TPM	Trusted Platform Module
XDR	Extended Detection and Response
ZTNA	Zero Trust Network Access

1.2 Reference Documentation

Table 2. Reference Documents

Reference	Source
Intel® Xeon® Scalable Platform Built for Most Sensitive Workloads	https://www.intc.com/news-events/press-releases/detail/1423/intel-xeon-scalable-platform-built-for-most-sensitive
Crypto Acceleration: Enabling a Path to the Future of Computing	https://newsroom.intel.com/articles/crypto-acceleration-enabling-path-future-computing
Golang	https://go.dev/
HashiCorp Vault	https://www.Vaultproject.io/ https://medium.com/hashicorp-engineering/hashicorp-Vault-performance-benchmark-13d0ea7b703f
Occlum	https://github.com/occlum/occlum
LUKS2 On-Disk Format Specification	https://gitlab.com/cryptsetup/LUKS2-docs/blob/main/luks2_doc_wip.pdf
Intel SGX Programming Reference and SDK for Linux	https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html#combined https://download.01.org/intel-sgx/latest/linux-latest/docs/ https://github.com/intel/linux-sgx
National Institute of Standards and Technology FIPS Publication 197, Advanced Encryption Standard (AES)	https://csrc.nist.gov/publications/detail/fips/197/final
3rd Generation Intel® Xeon® Scalable Processor - Achieving 1 Tbps IPsec with Intel® Advanced Vector Extensions 512 (Intel® AVX-512) Technology Guide	https://networkbuilders.intel.com/solutionslibrary/3rd-generation-intel-xeon-scalable-processor-achieving-1-tbps-ipsec-with-intel-advanced-vector-extensions-512-technology-guide
Create Intel SGX VM in the Azure portal	https://docs.microsoft.com/en-us/azure/confidential-computing/quick-create-portal
Intel® Software Guard Extensions (Intel® SGX) – Key Management Reference Application (KMRA) on Intel® Xeon® Processors Technology Guide	https://networkbuilders.intel.com/solutionslibrary/intel-sgx-kmra-on-intel-xeon-processors-technology-guide

Reference	Source
Intel® Software Guard Extensions (Intel® SGX) - Key Management Reference Application (KMRA) on Intel® Xeon® Scalable Processors User Guide	https://networkbuilders.intel.com/solutionslibrary/intel-sgx-kmra-on-intel-xeon-processors-user-guide
Intel® Software Guard Extensions (Intel® SGX) – Securing Private Keys in an Encrypted Enclave for Your Service Mesh Demo	https://networkbuilders.intel.com/intel-software-guard-extensions-intel-sgx-securing-private-keys-in-an-encrypted-enclave-for-your-service-mesh-demo
Intel® Trust Domain Extensions (Intel® TDX)	https://www.intel.com/content/www/us/en/developer/articles/technical/intel-trust-domain-extensions.html
Intel® AVX-512 and Intel® QAT - Accelerate WireGuard Processing with Intel® Xeon® D-2700 Processor Technology Guide	https://networkbuilders.intel.com/solutionslibrary/intel-avx-512-and-intel-qat-accelerate-wireguard-processing-with-intel-xeon-d-2700-processor-technology-guide
Zero Trust - Zero Trust Reference Architecture Technology Guide	https://networkbuilders.intel.com/solutionslibrary/zero-trust-zero-trust-reference-architecture-technology-guide

2 Overview

Protecting data at rest is mandatory in most systems for security considerations. Cybersecurity & Infrastructure Security Agency (CISA) defines data security as a pillar in zero trust maturity model to secure data at rest. There are increasing trend of attacks that exploit security loopholes in solutions for data at rest protection. To address these issues, well-defined regulation and compliance requirements (NIST 800-53, FIPS 200, FIPS 199, GDPR, HIPAA, etc.) provide guidance on protecting data at rest. Encryption is the most critical approach to avoid the exposure of sensitive data at rest. There are several different software-based methods for achieving encrypted data at rest including application, file system, and block layer encryption methods. Encryption at the block layer, also called as full disk encryption, provides the most comprehensive protection and is transparent to applications. It has thus received the wide adoption in production systems. Full disk encryption can be achieved either with software-based block subsystems or through specialized storage hardware called self-encrypting disks (SEDs).

2.1 Common Full Disk Encryption Solutions

2.1.1 TPM-based Full Disk Encryption

A typical production system with disk encryption is shown in Figure 1. Trusted Platform Module (TPM) is widely used as a secure key storage for full disk encryption. The Linux Unified Key Setup (LUKS) is a standard disk encryption specification for full disk encryption on Linux machines.

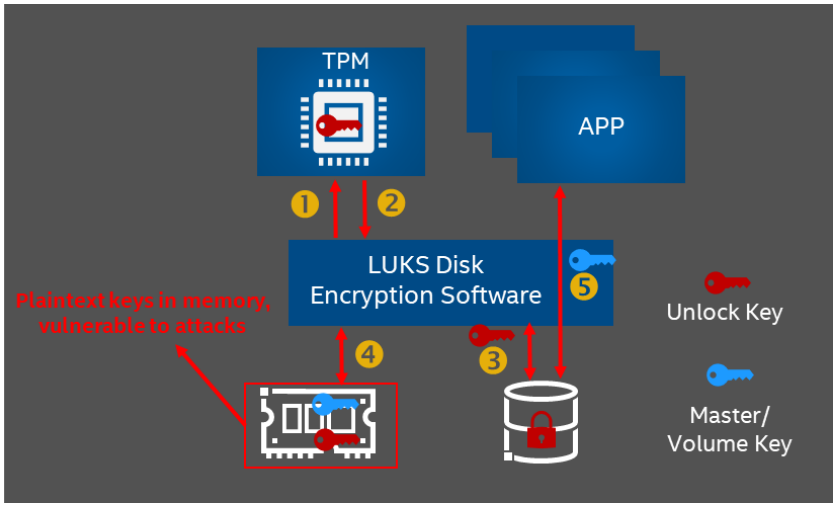

Figure 1. TPM-based Full Disk Encryption System

Below depicts the overall workflow:

1) The disk encryption software requests the LUKS unlock key from the TPM.
2) If the system integrity is verified with TPM, the TPM releases the decrypted LUKS unlock key to the disk encryption software.
3) The disk encryption software then uses the decrypted LUKS unlock key (in memory) to unlock the encrypted LUKS partition.
4) The LUKS master key is derived from the unlock key and stored in memory for the disk encryption software processing.
5) Applications operate with the LUKS partition as if it were unencrypted with the LUKS master key.

The main vulnerability exists since full disk encryption keys always stay plaintext in memory.

2.1.2 HSM-based Full Disk Encryption

Hardware security module (HSM) based full disk encryption is another deployment model in production. Such systems typically consist of production servers, a key management server (KMS), and a hardware security module (HSM).

- **HSM**: An appliance with customized hardware designed for secure and performant cryptographic key generation and storage. HSMs are expensive and have user interfaces that are less flexible than alterative software-based solutions. HSM has served enterprise data centers for years but it is more difficult to integrate HSMs at the network edge and in the cloud.
- **KMS**: Data security solution vendors such as Venafi, Vault, and Fortanix which provide certificate and key management solution. These extensive software platforms typically offer KMS functionality with a full key lifecycle management suite of features. KMSs are feature-rich and offer flexible user and application interfaces making them easily consumable by end-users and client applications. However, KMSs are disadvantaged by weaker security guarantees such as the inability to secure data in use, that is, when using KMSs, secrets may be stored in plaintexts in the application's address space. Cloud KMSs such as AWS KMS and Azure Key Vault are hosted in public clouds and are typically hardware protected and accelerated by HSM.
- **Production Server**: To meet various regulatory and compliance requirements, production servers may run a disk encryption process to protect data at rest. The disk encryption process usually retrieves the encryption key from a KMS to encrypt and decrypt application data before it is read from or written to disk. Here, the production sever can be a server running in the enterprise data center, or edge server, or a network/security system.

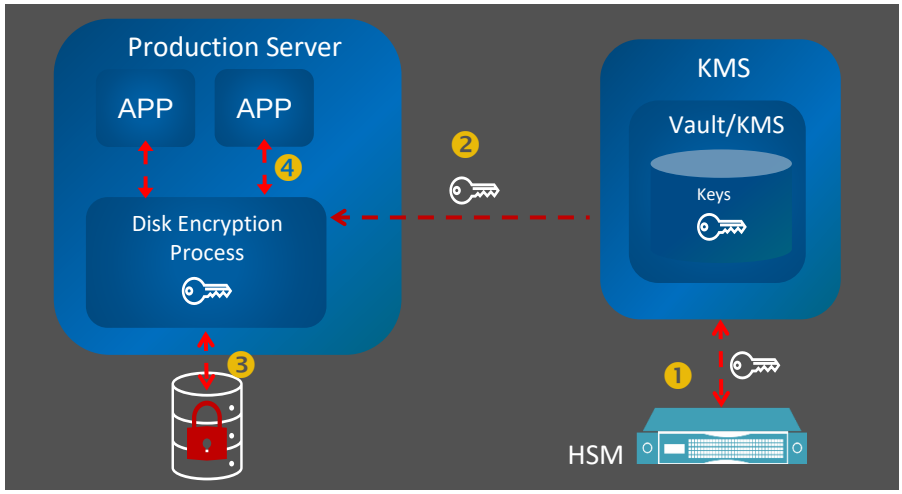

Figure 2. HSM-based Full Disk Encryption System

The basic workflow of the system shown in Figure 2 is as follows:

1) The KMS provisions cryptographic keys and conducts full lifecycle key management.
2) Upon an authenticated and authorized request from the production server, the KMS brokers the encryption key between the HSM and the production server's disk encryption process via a TLS protected connection.
3) The disk encryption process running on the production server uses the received key to enable the full disk encryption setup.
4) The read/write traffic between the encrypted disk partition and client applications will be automatically encrypted and decrypted respectively.

2.2 Issues with Existing Encryption at Rest Solutions

Securing data at rest and in transit alone may not always be enough to protect enterprise resources. Protecting data in-use is increasingly important due to more sophisticated attacks nowadays. As an example, a recent supply chain attack which targeted a Software as a Service (SaaS) build and distribution system allowed a malicious party to successfully exfiltrate filesystem data that was encrypted at rest. The theft of encrypted data would normally pose little risk; however, the attackers were able to extract the decryption key from a running program potentially exposing software supply chain secrets that were thought to be secured at rest. Existing encryption at rest solutions that do not leverage technologies which protect data in use may not always be secure. Existing solutions with such vulnerabilities are victims of an increasing number of critical incidents.

If data is encrypted at rest, then the encryption key is better to be secured even while in use. Figure 3 illustrates a scenario in which an encryption key is delivered to an unprotected running process. The address space of the process does contain the key in clear text given no method was applied to encrypt its memory despite the program having access to sensitive data. If the attacker can retrieve the key from the exfiltrated memory dump then any data, although encrypted at rest, is at risk of decipherment.

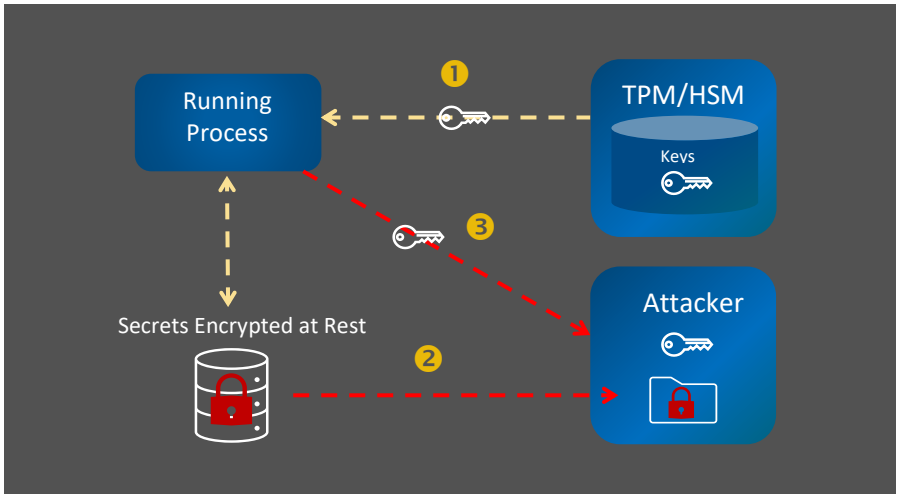

Figure 3. Attack on Unprotected Process containing Encryption Key

3 Intel® Processor Technology

3.1 Intel® Xeon® Processors

Network & Security Accelerators	5th Gen Intel Xeon Scalable Processor	4th Gen Intel Xeon Scalable Processor	3rd Gen Intel Xeon Scalable Processor	Intel® Xeon® D-2700/1700 Processors
Intel AVX-512	✓	✓	✓	✓
Intel Crypto Acceleration	✓	✓	✓	✓
Intel QAT(Integrated)	✓	✓		✓
Intel SGX	✓	✓	✓	✓
Intel TDX	✓	Limited		
Intel AMX	✓	✓		
Intel Deep learning Boost(VNNI, BF16)	✓	✓	✓	✓

Figure 4. Intel® Xeon® Processors

Figure 4 shows the comparison of security related features in the latest Intel® Xeon® processors.

3rd Gen Intel® Xeon® Scalable Processor delivers advanced features in networking and security. In crypto acceleration, new vector Intel® Advanced Encryption Standard New Instructions (AES-NI) and Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instructions deliver crypto performance improvements. This benefits the processing of both data in transit (network protocols like IPsec and TLS) and data at rest (file encryption, block encryption, etc).

Intel® SGX, in 3rd Gen Intel Xeon Scalable Processor, secures data in use. It addresses vulnerabilities of exposed plaintext secrets (private keys, tokens, etc) in memory that could be stolen by sophisticated attacks. Intel SGX is a core technology to design a zero trust system with hardened secrets protections.

Intel® Deep Learning Boost technology with Vector Neural Network Instructions (VNNI) improves inference performance for deep learning workloads. As zero trust solutions start to embrace AI for intelligent user to application segmentation, these built-in AI acceleration features could benefit zero trust solutions. Intel AVX-512 speeds up raw unstructured data processing for AI model training. Overall, deploying zero trust solution with 3rd Gen Intel Xeon Scalable Processor allows security and performance in place.

4th Gen Intel® Xeon® Scalable Processor has acceleration and security features (such as Intel SGX) from prior generation while at the same time upgrades with accelerators for key security workloads. Intel® QuickAssist Technology (Intel® QAT) is now built in the processor as an accelerator. Intel QAT offloads and accelerates asymmetric encryption tasks such as TLS. Intel® Trust Domain Extensions (Intel® TDX) is introducing new deployment of hardware-isolated virtual machines (VMs) called trust domains (TDs). Intel TDX is designed to isolate VMs from the virtual-machine manager (VMM)/hypervisor and any other non-TD software on the platform to protect TDs from a broad range of software attacks. Intel TDX is expected to be available through select public cloud service providers (CSP). Intel® Advanced Matrix Extensions (Intel® AMX) accelerates both inference and model training, which are now increasingly critical given pervasive AI usage and added intelligence in security systems for use cases such as zero-day threat detection and prevention.

5th Gen Intel® Xeon® Scalable processor has all features supported in 4th Gen Intel Xeon Scalable processor. In addition, Intel TDX will be widely available outside the public CSP services starting from this generation.

Intel® Xeon® D-2700/1700 Processors are designed with advanced security features, including Intel AVX-512, Intel crypto acceleration, Intel QAT, Intel SGX and Intel Deep Learning Boost technology. Their target use cases include network and security appliances, edge networking and SASE. A design focused on deployments at the edge for its better power and space utilization efficiency features.

3.2 Encryption at Rest with Intel Confidential Computing Technology

3rd Gen Intel Xeon Scalable processor is equipped with Intel SGX that allows standard off the shelf servers enabled for confidential computing. The follow-on 4th Gen Intel Xeon Scalable processor and Intel Xeon D-2700/1700 processers also carry over support of Intel SGX. Intel SGX offers hardware-based memory encryption or a secure enclave in memory that isolates specific application code and data in trusted execution environment (TEE). 4th and 5th Gen Intel Xeon Scalable processor also deliver Intel TDX support to secure VM in trust domain (Intel TDX availability on 4th Gen limited to select public CSPs).

Intel SGX/TDX allows zero trust principle to protect cryptographic keys in use. So disk encryption process in production servers can securely conduct crypto operations within Intel SGX/TDX (encrypted memory) instead of processing keys in un-encrypted memory. In addition, key management software can also be protected by Intel SGX/TDX. Intel SGX/TDX can significantly reduce the attack surface.

3.3 Encryption at Rest with Intel Crypto Technology

Crypto technology is at the core of encryption data at rest. This usually involves data encryption key and key encryption key. AES algorithms are widely used for encryption at rest and included as part of compliances by NIST who recommends both AES-256 and AES-128 for long-term storage use. Full disk encryption is mostly based on symmetric encryption algorithms as defined in Linux Unified Key Setup (LUKS) on-disk format specification. Since these encryption algorithms are compute intensive, Intel technologies such as Intel QAT and Vector AES (VAES) instruction-set found in Intel processors can significantly accelerate the performance of encryption at rest.

4 Open-Source Software Technology for Encryption at Rest

There is a strong open-source software ecosystem for encryption at rest, ranging from key and secret management, Linux user space utility tools, Linux kernel device-mapper crypto modules, etc. Intel contributes to such open-source communities in large extent.

4.1 HashiCorp Vault

HashiCorp Vault is a widely used open-source software, It is popular to DevOps scenario, it has the automated access control and life cycle management for secrets, such as encryption keys, passwords, API tokens, certificates and etc. It is used to build an encryption at rest reference solution with following responsibilities:

- Cryptographic key management: Vault generates, encrypts and store keys for disk encryption in backend secret engine, such as generic key-value store . There also can be an advanced life cycle management for keys.
- Authentication: Vault grants access to keys, and encryption capabilities by issuing a token based on policies associated with client's role and configured secret id. Users can define access capabilities for security policy control for associated secret engine. Only for client with valid Vault token, role and secret id can retrieve disk encryption key from Vault via restful API calls.

Vault can deliver a secure and automated encryption key management for disk encryption operations. However it still exposes encryption keys in plaintext in memory when conducting encryption operations. This creates a potential loophole that attackers could exploit to steal the secret keys. Running Vault in an Intel SGX/TDX can resolves this issue.

4.2 Cryptsetup and dm-crypt

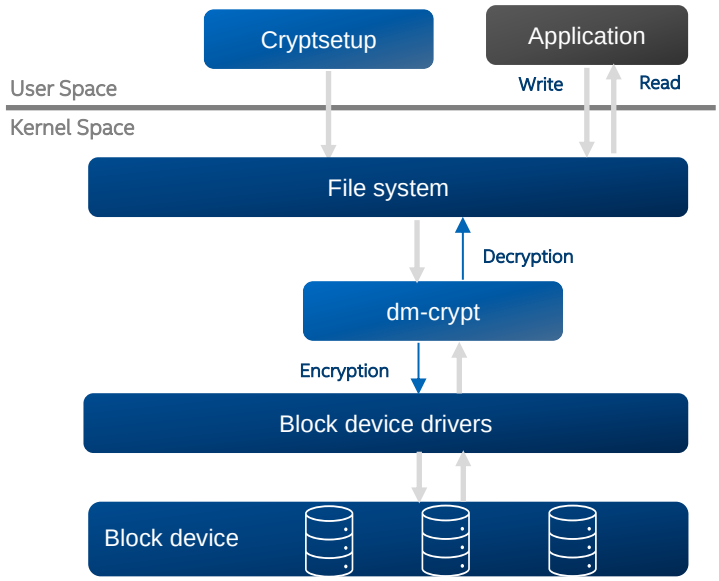

Figure 5. Cryptsetup and dm-crypt Workflow

Linux Unified Key Setup (LUKS) is a full disk encryption specification on Linux. Most full disk encryption solutions on Linux are based on LUKS. As show in Figure 5, Open-source software Cryptsetup is a user space utility tool that provides a convenient way to setup full disk encryption. It sets up full disk encryption based on a Linux kernel module called dm-crypt. When application need to write and read data to encrypted disk partition, dm-crypt conducts the heavy lifting of encrypting and decrypting for all disk IO traffic. The combination of Cryptsetup and dm-crypt delivers a set of baseline features for full disk encryption. At the core of the whole workflow is the use of keys for disk encryption. A mature solution requires key lifecycle management and protection of key from exposure.

4.3 Vaultlocker

Vaultlocker is an open-source software framework to automate full disk encryption. As a broker between Cryptsetup and Vault, Vaultlocker maintains encryption keys in Vault and provide wrapper interfaces to supply the key to Cryptsetup for full disk encryption. It sets role and permission for key storage in the key-value backend engine of Vault so only access to key is with permission. Furthermore, Vaultlocker uses Cryptsetup to encrypt specified disk partition. Once the partition setup is completed, all data writes to disk partition are encrypted automatically and all data reads from disk partition are decrypted automatically.

5 Zero Trust Full Disk Encryption Solution

Given the increasing attack surfaces, it is always challenging to build an encryption at rest system immune to exploits. We design a secure full disk encryption solution with confidential computing technology, a new way to realize the zero trust principles.

5.1 System Architecture

The system consists of Vault, Vaultlocker and LUKS software (Cryptsetup and dm-crypt) as shown in Figure 6.

- The Vault acts as a KMS and maintains disk encryption key lifecycle management that is required as best security practice. It stores encryption keys for disk encryption and only access with specified role and permission could operate on the key.
- Vaultlocker is a broker between Vault and disk encryption software Cryptsetup. Vault client within vaultlocker establishes https connections and perform key operations with Vault. Vault stores encryption key generated by vaultlocker. When required, vaultlocker retrieves key from Vault key value backend store with permission. Vaultlocker can trigger cryptsetup with retrieved key to setup encrypted partition on disk.
- All the operations including Vault, Vaultlocker, Cryptsetup are protected within the Intel SGX/TDX.

This solution hardens security by combining powerful open-source software with advanced hardware features on 3rd, 4th and 5th Gen Intel Xeon Scalable processors and Intel Xeon D-2700/1700 processors.

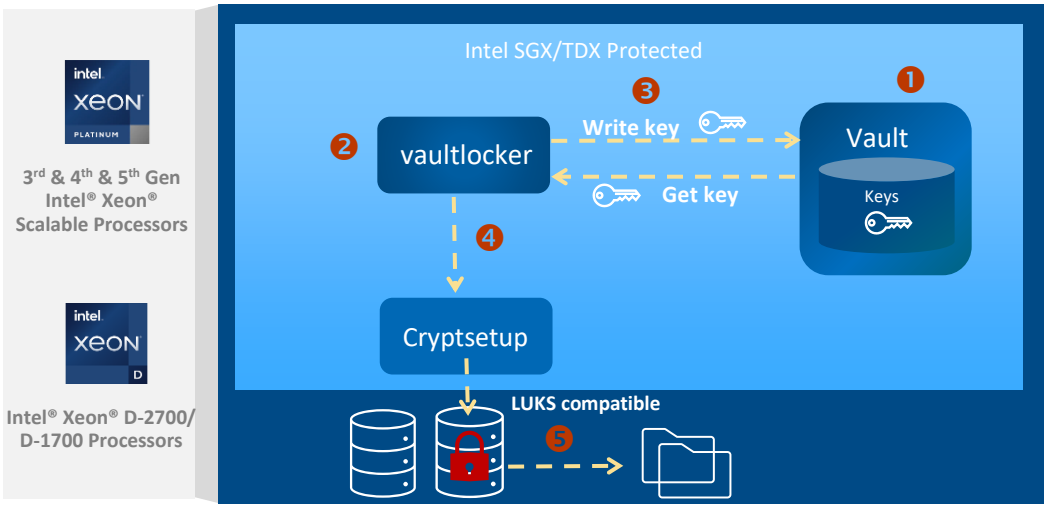

Figure 6. Full Disk Encryption Architecture Overview

5.2 System Workflow

Figure 6 shows system workflow:

1. Vault configuration by creating a backend secret engine for key-value store. Add authentication approach with client application role, secret id and access policy (create, read, update, delete and list).
2. Vaultlocker configuration with the host address of Vault and also with client role, secret id and backend secret engine name configured in step 1.
3. Vaultlocker generates a disk encryption key and stores it in Vault.
4. Vaultlocker retrieves key from Vault with preconfigured role and permissions and triggers Cryptsetup to encrypt specified disk partition.
5. Create file system for encrypted partition and mount the encrypted disk.

Once the setup is done, all full disk encryption operations are fully transparent to applications. Every disk write and read from applications to the encrypted partition will trigger dm-crypt kernel module which in turn call kernel crypto module to conduct encryption for each write request and decryption for read requests.

5.3 Zero Trust Network Access Example Usage Scenario

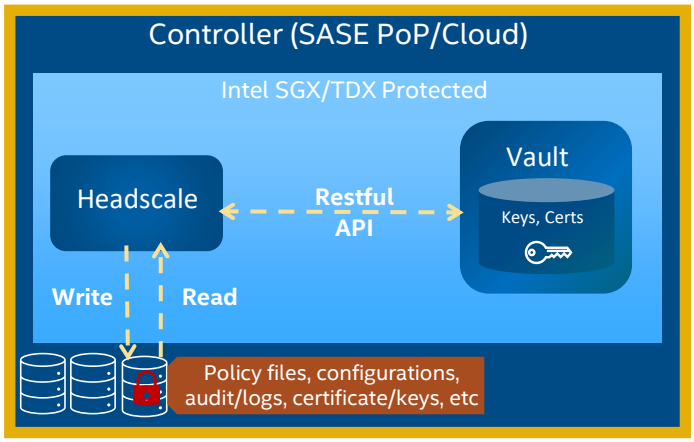

Figure 7. Full Disk Encryption Usage in Intel Zero Trust Network Access Reference Architecture

Figure 7 shows full disk encryption usage in Intel Zero Trust Network Access Reference Architecture (Intel ZTNA Reference Architecture) which is designed for SASE ZTNA use case. Securing data at rest is a key requirement for SASE infrastructure. Intel ZTNA Reference Architecture complies with NIST's definition of zero trust architecture with decoupled control path and data path. The ZTNA controller for control path can be deployed at a SASE point of presence (PoP) infrastructure that require regulations and compliances for data security. By applying our full disk encryption solution, the ZTNA controller can maintain sensitive files (policy files, configurations, audit/logs, certificate/keys, etc) in encrypted disk partition. Each write and read traffic

to this encrypted disk is fully protected by Intel SGX/TDX. This significantly reduces the risk of attackers getting plaintext encryption keys for full disk encryption from system memory and thus avoids the case of decrypting credentials in storage.

This example shows a good reference about how to apply our solution to SASE systems. Our full disk encryption solution with Intel SGX/TDX hardened security on Intel Xeon processors can scale out to infrastructures designed with protecting data at rest as a major target.

6 Full Disk Encryption Deployment

This section shows the basic deployment of full disk encryption solution. It covers detailed install instructions for core components and setups.

6.1 Deployment Setup

6.1.1 Vault Setup

1. Install the Vault.

```
# wget https://github.com/hashicorp/vault/archive/refs/tags/v1.10.3.tar.gz
# mkdir -p vault_v1.10.3
# tar -xvzf v1.10.3.tar.gz -C vault_v1.10.3 --strip-components=1
# cd vault_v1.10.3
# rm -f vault_v1.10.3/bin/vault
# go mod tidy
# go build -o bin/vault
```

2. Edit the configuration file of Vault.

```
# echo "storage \"raft\"{
#            path=\"/etc/vault\"
#            node_id=\"raft_node_1\"
# }
# listener \"tcp\" {
#   address     = \"127.0.0.1:8200\"
#   tls_disable = 1
# }
# cluster_addr=\"http://127.0.0.1:8201\"
# api_addr=\"http://127.0.0.1:8200\"" >vault_config.hcl
# cp vault_config.hcl /etc/vault/
```

3. Start Vault.

```
# ./vault server -config /etc/vault/vault_config.hcl -log-level=error &
# export VAULT_ADDR="http://127.0.0.1:8200"
# ./vault operator init -key-shares=1 -key-threshold=1
```

The initialize command will output the root token and unseal key of vault, we need to save them to a safe place.

The status of vault is sealed by default. Before saving secrets to vault, we need to unseal it first.

```
# ./vault operator unseal ${unseal_key}
```

4. Configure Vault.

```
# vault secrets enable -path=luks kv

// enable auth method approle
# vault auth enable approle
#
// create a role vaultluks
# vault write -f auth/approle/role/vaultluks
#
// read role-id to config vaultlocker later
# vault read auth/approle/role/vaultluks/role-id
#
// generate a secret-id to configure vaultlocker later
# vault write -f auth/approle/role/vaultluks/secret-id
#
// config policy for the role vaultluks
```

```
# echo "path \"luks/*\" {
#     capabilities = [\"create\", \"read\", \"update\", \"delete\", \"list\"]
# }
# " > vaultluks.hcl
# vault policy write vaultluks vaultluks.hcl
# vault write auth/approle/role/vaultluks policies="vaultluks"
```

6.1.2 Vaultlocker Setup

1. Install Vaultlocker.

```
// install vaultlocker
# apt update
# apt-get install vaultlocker
```

2. Configure Vaultlocker to use approle *vaultluks* to login Vault. The *role_id* and *secret_id* are generated at previous step of "Configure Vault".

```
# echo "[vault]
# url = http://127.0.0.1:8200
# approle = ${role_id}
# secret_id = ${secret_id}
# backend = luks
# " > /etc/vaultlocker/vaultlocker.conf
```

3. Encrypt the specified disk partition. Take "/dev/vdb" as an example.

```
# vaultlocker encrypt /dev/vdb
```

Running this command, vaultlocker will create an encrypted disk partition above disk /dev/vdb named crypt-{UUID}. UUID is randomly generated. The key-value pair <UUID, encryption key> will be stored into the kv backend *luks* of vault.

6.1.3 File System Setup

Create file system for encrypted partition and mount the encrypted disk.

```
# mkfs.ext4 /dev/mapper/crypt-${UUID}
# mkdir -p /etc/headscale
# mount /dev/mapper/crypt-${UUID} /etc/headscale
```

7 Zero Trust Reference Architecture Software Availability

Intel Zero Trust Network Access Reference Architecture builds a good reference implementation of zero trust system. It differentiates with other solutions by applying the latest Intel security technologies including confidential computing and crypto acceleration. The first release (22.06) covers all features including user authentication with Azure AD, service authorization with RBAC, secure WireGuard tunnel and secrets protection with Intel SGX. This full disk encryption solution is a new feature introduced in 23.03 release. The 23.03 release also brings new capabilities including TDX support, IPSec support and Let's Encrypt integration for certificate management. To access this software, please contact your Intel sales partner, or the document authors.

8 Summary

Protecting data at rest is a fundamental requirement embraced by enterprise IT systems. But current solutions are still vulnerable to sophisticated attacks given the increasing trend of security incidents. Complying with zero trust principles for securing data at rest is critical to reduce attack surface and avoid data and credential theft.

This document demonstrates best practices to strengthen data security with software system design and Intel Xeon Scalable and Xeon D processors. The combination of Intel confidential computing feature (Intel SGX/TDX), Vault and LUKS software provides comprehensive protection for full disk encryption without exposing plaintext keys in memory. This solution gives a good reference that can scale out to IT infrastructure designs to fully secure data at rest with Intel technologies.

Appendix A Platform Configuration

Name	Inspur NF5280M6
Vendor	Inspur
Product Name	Server Machine
BIOS Version	06.00.01
SGX	Yes
SGX EPC size	2GB (max 64GB)
OS	Ubuntu 20.04.3 LTS
Kernel	5.16.0-rc4
IRQ Balance	enabled
CPU Model	Intel® Xeon® Platinum 8358 @ 2.60GHz
Base Frequency	2.6GHz
CPU Family	6
CPU Model	106
CPU(s)	128
Thread(s) per Core	2
Core(s) per Socket	32
Socket(s)	2
NUMA Node(s)	2
Turbo	enable
Memory Installed	512GB

Software Configuration	Software version	Location
Host OS	Ubuntu 20.04.3 LTS	https://ubuntu.com/
Kernel	5.16.0-RC4	https://www.kernel.org/
Vault	Vault 1.10.3	https://www.vaultproject.io/
Occlum	Occlum 0.26.3	https://occlum.io/
VaultLocker	1.0.6	openstack-charmers/vaultlocker: Automated storage and retrieval of dm-crypt keys using Vault (github.com)

1122/DN/WIPRO/PDF 751287-001US