Trusted Computing Group Storage Work Group
Key Per I/O Security Subsystem Class (SSC) Specification FAQ

September 25, 2024

Q. What is Key Per I/O (KPIO)?

A. NVMe Key Per I/O (KPIO) is a fine-grained data at-rest encryption mechanism that builds on the NVMe 2.0 specification family. Per the NVMe TP4055, “The Key Per I/O capability provides a mechanism to use encryption keys that have been injected into an NVM subsystem by a host. The mechanism to perform activation of the Key Per I/O capability, encryption key injection, management, and association to encryption key tag is outside the scope of the NVMe specification. One mechanism is defined in the TCG Storage Security Subsystem Class: Key Per I/O Specification using the NVMe Admin commands Security Send and Security Receive.”

Q. How does TCG contribute to KPIO?

A. The Trusted Computing Group (TCG) has specified a KPIO architecture that is interoperable with NVMe KPIO (TP4055). The specific TCG details are included in the TCG Storage Security Subsystem Class: Key Per I/O v1.00, TCG Key Per I/O App Note v1.0, and the TCG Storage Interface Interactions Specification (SIIS) v1.11 or later, which can be found at https://trustedcomputinggroup.org/work-groups/storage/ .

Q. What kind of scenarios are a good fit for KPIO?

A. KPIO is a particularly useful encryption technology for scenarios where a Storage Device is being shared by multiple hosts and/or multiple tenants on a host (e.g., VM and containers). Since a KPIO-enabled Storage Device plays no significant role in the key management lifecycle, the keys can come from a variety of sources external to the device. This can result in a Storage Device having encrypted data (ciphertext) protected by numerous media encryption keys (MEKs), which may only be known to a particular host and/or tenant. Among the many benefits this provides:

- Rendering drive exfiltration useless since the MEKs used to encrypt user data are never stored by the device and are immediately lost to the device on power loss (i.e., data-at-rest protection & platform binding guarantees).
- Enabling granular data encryption and eradication use cases where the data to be eradicated may be distributed across multiple devices and may be mixed with other data that needs to be retained.
- Storage system throughput/bandwidth increases stemming from parallelization of encryption of distributed objects that belong to the same encryption domain (i.e., share the same encryption key) across multiple devices’ HW encryption engines.
- Enabling the scaling of number of tenants per device without corresponding increases in on-device key management implementations/complexities and maintenance costs.
- Enabling enhanced TCG device’s namespace management capabilities where admins no longer require tenants’ intervention to lock access to a namespace.

Q. Conceptually, how does the KPIO encryption work?

A. The basic concept is that the encryption engine (XTS-AES-256) of a Storage Device is used to encrypt some or all data in the Storage Device, but the key management lifecycle is handled external to the Storage Device. To do this, a small volatile key cache (typically a few hundred to a few thousand keys) on the Storage Device is loaded with MEKs that the Storage Device will use. All read and write commands for the KPIO managed portion of the

Storage Device include a KPIO key tag that is used as a pointer into the key cache. The Storage Device uses the data encryption key stored in the specified key cache location to either encrypt (for writes) or decrypt (for reads) host data.

Q. What is a KPIO key tag?

A. A KPIO key tag consists of a Namespace ID and a Key Tag pair. The Key Tag value identifies a specific Key Tag Slot that is associated with a specific namespace.

Q. How are Key Tags Unique?

A. The scope of a Key Tag value is the associated Namespace. The Key Tag value is a zero-based index; if N key tags are supported for a namespace, then the Key Tags range from 0 to N-1.

Each namespaces will have Key Tag values of 0, 1...N-1 where each Key Tag may be associated with a different media encryption key.

Q. How is the key cache configured to support namespaces?

A. The alignment of host/tenants is typically for the whole Storage Device (sub-system) or some number of namespaces. In NVMe technology, a namespace is a collection of logical block addresses (LBA) accessible to host software. For a KPIO-enabled Storage Device, the key cache is carved up in such a way that each namespace is permitted to use a maximum number of keys (the total keys for all namespaces must not exceed the Storage Device's key cache size); key tags specified on reads/writes are relative to the namespace (e.g., key tag 1 can exist for each namespace).

Q. Are there constraints on the number of MEKs that can be used?

A. The host/tenant is not restricted to a maximum number of MEKs and may use a very large number of MEKs. This is accomplished by deleting and injecting MEKs as they are needed (typically before I/O requires the MEK) while remaining within the limits of the number of MEKs for the associated namespace.

Q. Does KPIO provide transport interfaces' communications security?

A. The Key Per I/O SSC does not have provisions to secure general communications with a storage device. However, the KPIO SSC defines a mechanism to wrap (encrypt) keys as they are transported across interfaces into the Storage Device. Transport encryption technologies such as PCIe IDE and SPDM may be used to protect host data and keys across the interface. KPIO is a data-at-rest protocol and therefore, its point of encryption is within the Storage Device.

Q. What are the different types of keys that are used in KPIO?

A. The following keys are used in KPIO:

- Encrypted Key Encryption Keys (eKEKs)
  - Symmetric keys that are generated, managed, and supplied to the Storage Device by the host (or tenants). They are sent to the Storage Device during initial setup, one-time configuration of KPIO on a device. They represent access control to be able to update other KEKs or inject MEKs to be used for a specific namespace (i.e., they are used to provide confidentiality and authenticate Media Encryption Keys (MEKs) or other KEKs as they are provisioned into the Storage Device). These KEKs are typically retained in non-volatile storage so that they do not have to be reloaded after Storage Device power cycles. They are erased on KEK update operation or on Revert.

- Encrypted Media Encryption Keys (eMEKs)
  - Symmetric keys that are generated, managed, and supplied to the Storage Device by the host. They are sent to Storage Device on every power-on reset to be able to read/write user data with a Storage

Device. They are only kept in volatile key cache of the Storage Device and are lost when a device experiences power cycles.

- Authority PINs
  - These are credentials supplied to the Storage Device during the configuration of KPIO. Admin Authority PIN/Credential is used to represent access control for configuring which KEKs are permitted to access a particular namespace.

- Asymmetric Key Pair
  - RSA-based Key Pair pre-provisioned into the Storage Device by the device's manufacturer. The device's public key is used to protect Key Encryption Keys (KEKs) as they are transported into the Storage Device.

Q. How are symmetric keys (KEKs and MEKs) injected using KPIO?

A. To use the KPIO functionality, KPIO must be activated. Once activated, the initial key encryption keys (KEKs) can be injected, using either a plaintext KEK option (over a secure communication channel such as PCIe IDE/SPDM or in a secure environment) or a wrapped KEK option using a PKI public key. Once the initial KEK has been injected, it can then be used to inject other wrapped KEKs and/or wrapped MEKs for each namespace.

The injection occurs over a newly defined stateless TCG security protocol (protocol ID 0x03), with keys' payload and attributes formatted into a KMIP Request Message. The KMIP Request Message is transported as a ComPacket payload in a Security IF-SEND command. The response is returned to the host as a KMIP Response Message, also encoded as a ComPacket payload in a Security IF-RECV command.

Using a stateless security protocol (i.e., security protocol with no TCG sessions and associated TCG tokenization) for injecting keys allows for a dynamic and efficient injection of a large number of keys across multiple namespaces using a single command, which is useful for minimizing configuration latencies in multi-VM environments (e.g., latencies during VMs creation, VMs migration, Servers Boot-up, etc.)

Q. How is OASIS Key Management Interoperability Protocol (KMIP) leveraged?

A. Due in part to the popularity of the OASIS Key Management Interoperability Protocol (KMIP) in handling keys for storage-based encryption, KPIO uses a small subset of KMIP based commands to inject and manage keys in the Storage Device. These KMIP commands conform with the KMIP 2.1 specification, but they are not associated with any of the OASIS KMP Profiles. Under the right conditions it may be possible for a host to leverage a key management server, using KMIP, in such a way that the host may be able to serve as a KMIP proxy between the KMS and the Storage Device.

Q. Can KPIO-enabled Storage Devices be managed separately from use?

A. Yes, configuring and managing KPIO may be separate from the hosts/tenants using KPIO. This may be necessary in particular, when one or more hosts/tenants have limited visibility of the Storage Device's namespaces. This separation of duties is possible within KPIO because Key Per I/O decouples device ownership from tenant access control, tenant access control from media encryption keys, and media encryption keys from LBA ranges, allowing significant flexibility with better ability to scale number of tenants and number of "encryption domains".

Q. Does KPIO require significant host side management capabilities to use?

A. KPIO can be as hands off for a host as Opal, but with the added benefit of being in direct control of the actual MEKs used by the device. For example, if a host only needs a single key per namespace, the host can set up all the keys it needs at boot and then perform I/O as it normally does with the only exception that the I/O commands are modified to specify that a key tag is present and that it has a value of 0 (default key tag value).

More host side management capabilities are only needed for managing more complex use cases. For example, use cases needing multiple keys and key tags per namespace (e.g., multi-tenants keys mapped to containers, virtual machines, etc.), require additional management, especially on Storage Devices with smaller key cache. Similarly, drive migration and hot insertion scenarios where the keys need to be swapped, re-provisioned, or managed on-demand.

Q. Are there performance implications associated with the use of KPIO?

A. Generally, no. There may be a small performance hit with deep queued I/Os on Storage Devices with a small key cache if that I/O requires new MEKs to be swapped in and out of the Storage Device's key cache. In those use cases, the performance penalty would be comparable to existing SED architectures such as Opal/eSSC experience when a host needs to perform I/O on a locked range.

Q. Where can you obtain copies of the TCG Key Per I/O SSC specification and related TCG specifications?

A. The TCG Key Per I/O SSC specification v1.00 and related TCG specifications (e.g., TCG Key Per I/O App Note v1.0 and the associated errata, TCG SIIS v1.11) are available at https://trustedcomputinggroup.org/work-groups/storage/