How To Use An Encryption Key Per I/O

Sponsored by NVM Express™ organization, the owner of NVMe® Family of Specifications
Speakers

Festus Hategekimana

Frederick Knight

SOLIDIGM

NetApp
Agenda

• Key Per I/O Technology Overview
• How to Use Key Per I/O
  • One-time Setup
    • Capabilities Discovery
    • Enabling and Configuring Key Per I/O
  • Host Management of a Storage Device’s Key Cache
    • Loading of Encryption Keys into a Storage Device
    • Specifying Encryption Keys During I/O
    • Locking the Key Cache
  • Disabling Key Per I/O
• Industry Specifications Status
• Q & A
Key Per I/O Technology Overview

- Co-developed by NVM Express & TCG to enable Storage Devices (SDs)' support of Host-Managed (i.e., Customer-managed) Storage Encryption Use Cases.
- Hosts no longer need to encrypt-at-compute with customer supplied encryption keys. They can now parallelize encryption across SDs with host-supplied Media Encryption Keys (MEKs) to increase storage systems’ performance & bandwidth.
- Encrypted MEKs are injected into Self Encrypting Drive (SED)'s key cache and assigned a “Key Tag” by host SW.
- Subsequent I/O can use the “Key Tag” to identify the MEK to encrypt/decrypt data to/from the SD in a non-contiguous fashion.
- MEKs are encrypted (wrapped) by a Key Encryption Key (KEK).
- KEKs may be supplied encrypted via RSA-based Key Wrapping.
- MEKs are not stored in the NVM of the drive and are lost on power loss.
- Crypto erase is done by deleting the MEK from the Key Manager and the SSD’s key cache or by sanitizing entire SD.
Setting up Key Per I/O (one time setup): Capabilities Discovery

NVMe® Device Identify Discovery

- Identify Controller
  - Key Per I/O Capabilities field
    - Key Per I/O Supported (KPIOS) bit
    - Key Per I/O Scope (KPIOSC) bit
- Identify Namespace
  - Key Per I/O Status field
    - Key Per I/O Supported in Namespace (KPIONS) bit
    - Key Per I/O Enabled in Namespace (KPIOENS) bit
  - Maximum Key Tag (MAXKT) field
  - Key Per I/O Data Access Alignment and Granularity (KPIOADAAG) field

TCG Discovery (via NVMe Security Receive)

- Feature Level0 Discovery
  - Key Per I/O Security Protocols & ComIDs
  - Security properties for secure encryption key transport (RSA-OAEP wrapping, AES-GCM wrapping, etc.)
  - Number of Key Tags Supported (Globally vs Per-Namespace)
  - Maximum Supported Key Unique Identifier for Encryption Keys
  - Etc…
- Namespace Level0 Discovery
  - Managed By Key Per I/O bit
  - Number of Allocated Key Tags
Setting up Key Per I/O (One Time Setup): Enabling Key Per I/O

NOTE: Exact command tokenization details can be found in the TCG Key Per I/O Application Note.
Setting up Key Per I/O (One Time Setup): Enabling Key Per I/O

NOTE: Exact command tokenization details can be found in the TCG Key Per I/O Application Note

NOTE: Transitioning MEKs ownership to the host implies loss of SD-generated MEKs for namespaces managed by Key Per I/O (SD-generated MEKs are retained for namespaces not managed by Key Per I/O).
Setting up Key Per I/O (One Time Setup):
Configuring Key Per I/O

- Update Admin Credentials from defaults
- Configure Key Per I/O Policies Table (e.g., enable Replay Protection, enable RSA Wrapped KEKs, Disable Plaintext KEKs, etc.)
- Configure Key Tag Allocation Table (e.g., allocate Number of Key tags for each KPIO namespace, enable additional namespaces for KPIO, etc.)

NOTE: Exact command tokenization details can be found in the TCG Key Per I/O Application Note

HOST

SD

NVMe Security Send / Receive
[TCG SET Configure Key Per I/O Security Provider ]

Status
Host Management of the SD’s Key Cache:
Initial Loading of KEKs & MEKs

- Batch all KEKs & MEKs in a single KMIP message to the drive

NOTE: Exact command tokenization details can be found in the TCG Key Per I/O Application Note
Host Management of the SD’s Key Cache:
Initial Loading of KEKs & MEKs
Host Management of the SD’s Key Cache: Initial Loading of KEKs & MEKs
Host Management of the SD’s Key Cache: Selecting MEKs to Use During I/O

- NVM Express TP4055 defines new KPIO-related Command Extension Type (CETYPE) in DWORD12 and Command Extension Value (CEV) in DWORD13 fields for all read and write I/O commands to indicate to the Storage Device:
  - Key Tag Presence (CETYPE != 0).
  - Key Tag Value (CEV == KEYTAG) associated with MEK to be used for encryption or decryption of data in that I/O command.
Host Management of the SD’s Key Cache: Selecting MEKs to Use During I/O

Read/Write IO Example:
Host Management of the SD’s Key Cache: Updating the Key Cache

NOTE: Exact command tokenization details can be found in the TCG Key Per I/O Application Note

GET KEKs and MEKs’ UIDs from KeyUIDs Key Store

GET [MEK Key UID, Key Wrapping Specification (Encrypt with previously injected KEK(Key UID))]

NVMe Security Send / Receive [TCG KMIP IMPORT Keys]

- Batch all new MEKs in a single KMIP message to the drive

Status

NOTE: Updating Key Cache does NOT clear data written by previous keys!
New / Additional MEKs are loaded using previously established KEKs.

SD’s Volatile Key Cache/Table State On Init Key Cache Load

<table>
<thead>
<tr>
<th>NSID0</th>
<th>KeyTag0</th>
<th>Key 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSID1</td>
<td>KeyTag0</td>
<td>Key 2</td>
</tr>
<tr>
<td>NSID1</td>
<td>KeyTag1</td>
<td>Key 3</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSIDNN</td>
<td>KeyTagM</td>
<td>Key P</td>
</tr>
</tbody>
</table>

SD’s Volatile Key Cache/Table State After NSID0 Keys Update

<table>
<thead>
<tr>
<th>NSID0</th>
<th>KeyTag0</th>
<th>Key 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSID1</td>
<td>KeyTag0</td>
<td>Key 2</td>
</tr>
<tr>
<td>NSID1</td>
<td>KeyTag1</td>
<td>Key 3</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSIDNN</td>
<td>KeyTagM</td>
<td>Key P</td>
</tr>
</tbody>
</table>
Host Management of the SD’s Key Cache: Selecting new MEKs to Use During I/O

![Diagram showing NVMe I/O Queue and Key Lookup]

- **NSID1**: Write (Data, DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag1)
- **NSID0**: Write (Data, DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag0)
- **NSID1**: Write (Data, DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag2)
- **NSID0**: Read (Data, DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag0)

Key Lookup:
- **NSID0**: KeyTag0, Key 1
- **NSID1**: KeyTag2, Key 2
- **NSID1**: KeyTag1, Key 3
- **NSIDNN**: KeyTagM, Key P

Status: 3rd command fails with Invalid Key Tag Error Code

Updated SD’s Volatile Key Cache/Table At Runtime:
- **NSID0**: KeyTag0, Key 1
- **NSID1**: KeyTag2, Key 2
- **NSID1**: KeyTag1, Key 3
- **NSIDNN**: KeyTagM, Key P

NVM: Old Data Protected By Key 0

Data Protected By Key 1

AES-XTS

User Data

Plaintext User Data

Ciphertext User Data to/from NVM
Host Management of the SD’s Key Cache: Locking the Key Cache (All NSes vs. Per NS Locking)

- **NOTE:** Clear All MEKs command on all KPIO-Managed NSes clears their Key Cache and does not affect data.

- **NOTE:** Clear Single MEK Command targets a single slot of the key cache for a particular NS. No impact to the data.

---

**SD’s Volatile Key Cache/Table State After Clearing Single Key (Example)**

<table>
<thead>
<tr>
<th>NSID</th>
<th>Key Tag</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>KeyTag0</td>
<td>Key 1</td>
</tr>
<tr>
<td>1</td>
<td>KeyTag2</td>
<td>EMPTY</td>
</tr>
<tr>
<td>1</td>
<td>KeyTag3</td>
<td>EMPTY</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>NN</td>
<td>KeyTagM</td>
<td>Key P</td>
</tr>
</tbody>
</table>

**SD’s Volatile Key Cache/Table State After Clearing all KPIO Keys (Example)**

<table>
<thead>
<tr>
<th>NSID</th>
<th>Key Tag</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>KeyTag0</td>
<td>EMPTY</td>
</tr>
<tr>
<td>1</td>
<td>KeyTag2</td>
<td>EMPTY</td>
</tr>
<tr>
<td>1</td>
<td>KeyTag3</td>
<td>EMPTY</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>NN</td>
<td>KeyTagM</td>
<td>EMPTY</td>
</tr>
</tbody>
</table>
Disabling Key Per I/O

**NOTE:** Disable Key Per I/O Usage
- NVMe Security Send / Receive
- [TCG REVERT to purge all keys from the drive & Deactivate Key Per I/O Usage]

**NOTE:** Disable on all KPIO NSes.
- A successful REVERT execution makes host user data irretrievable even if the same keys are re-injected into the SD after re-enabling Key Per I/O

**NOTE:** Disable Per KPIO NS (NSID0 for ex).
- A successful SET execution makes host user data irretrievable even if the same keys are re-injected into the SD after re-enabling Key Per I/O

SD’s Volatile Key Cache/Table State On Cmd Completion (Example)
- NSID0: N/A  SD’s Generated Key0
- NSID1: N/A  SD’s Generated Key1
- NSIDNN: N/A  SD’s Generated KeyNN

OR

SD’s Volatile Key Cache/Table State On Cmd Completion (Example)
- NSID0: N/A  SD’s Generated Key0
- NSID1: KeyTag0  Key 2
- NSIDNN: KeyTagM  Key P

NOTE: Exact command tokenization details can be found in the TCG Key Per I/O Application Note
## Industry Specifications Status

<table>
<thead>
<tr>
<th>Specification</th>
<th>Industry Standard Body</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVMe® TP4055</td>
<td>NVM Express</td>
<td>Ratified</td>
</tr>
<tr>
<td>TCG Key Per I/O SSC v1.00</td>
<td>TCG</td>
<td>In Public Review</td>
</tr>
<tr>
<td>TCG Key Per I/O Application Note v1.00</td>
<td>TCG</td>
<td>In Public Review</td>
</tr>
<tr>
<td>TCG SIIS v1.11</td>
<td>TCG</td>
<td>Published</td>
</tr>
</tbody>
</table>
Questions?