SECURING FLASHARRAY™ "DATA AT REST"

This brief describes how FlashArray protects data stored in arrays against unauthorized access

Potential threats to the security of data stored in FlashArray systems include:

- Unauthorized access
- Recovery of stored data from misappropriated storage devices and arrays
- Recovery of stored data from decommissioned arrays.

This brief describes how arrays protect against the latter two threats. A companion brief, TB-160202, describes how FlashArray prevents unauthorized access to arrays.

FLASHARRAY DATA ENCRYPTION

The Purity//FA software that runs in FlashArray systems uses the well-known AES-256 algorithm in counter (CTR) mode to encrypt all data and most metadata stored on flash and staged in NVRAMs. The software uses write sequence numbers that increase monotonically for the life of an array to construct a unique AES-256 counter for every encryption.

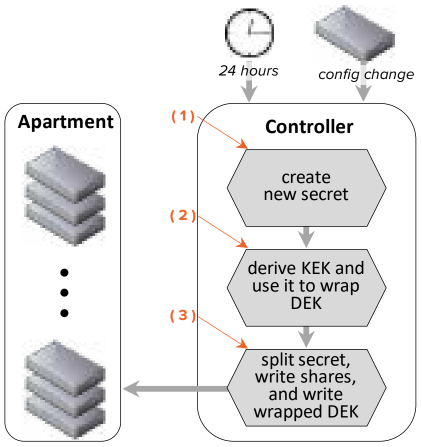
Software encryption makes quality and speed consistent regardless of flash device type. Moreover, SSDs need not be AES-256-capable to be supportable by FlashArray. Controllers use processor encryption instructions to minimize computational load.

Each array has a *data encryption key* (DEK), which it generates when it initializes its *apartment* (complement of flash storage devices). An array initializes its apartment when it is first installed, and at other times, for example, when it is "wiped" for return to Pure Storage after a trial. Arrays store their DEKs in *wrapped* (encrypted) form using the standard AESKW algorithm.

PROTECTING THE DEK

FlashArray encryption key management is transparent to administrators. An array wraps its DEK using AESKW with a *key encryption key* (KEK) which it derives from a random secret generated every 24 hours or whenever its apartment membership changes. It derives a new KEK from each secret[1] and re-wraps and stores its DEK. An array splits its secret into shares, and stores each share persistently in the header area of a different flash device. The splitting algorithm is such that shares from a *quorum* (more than half) of the devices in an apartment is required to recreate the full secret.

Every 24 hours, and whenever its apartment membership changes, an array's primary controller generates a new secret (1) from which it derives a new KEK (2). It uses the KEK to wrap its DEK and then splits the secret into shares which it writes to different flash device header areas along with the wrapped DEK (3).


24 hours
config change
Apartment
Controller
(1)
create new secret
(2)
derive KEK and use it to wrap DEK
(3)
split secret, write shares, and write wrapped DEK

Refreshing the KEK

1 At the time pf publication, secret and KEK are identical unless *enhanced protection for data at rest* (page 2) is in use.

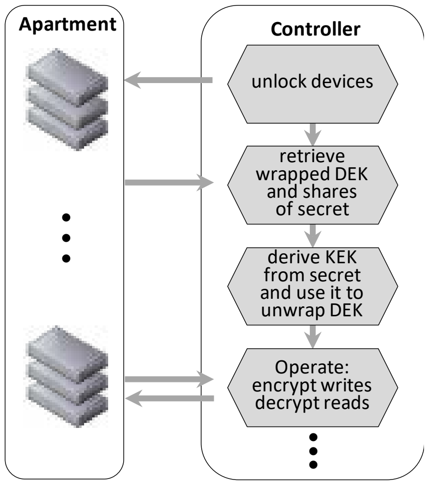
RETRIEVING THE DEK AT STARTUP

FlashArray SSDs power on in a locked state, in which no data can be read or written. Arrays use a “well-known” (universal) key to unlock their devices; data on devices, including header area contents cannot be retrieved or overwritten without the universal key, which is never exposed on any interface. Direct Flash Modules (DFMs) do not power on locked.

After unlocking its devices at startup, an array’s primary controller:

- retrieves secret shares from flash device header areas and recombines them to recreate the array’s current secret
- derives the KEK from the current secret and uses it to unwrap the array’s DEK which it uses to encrypt data and metadata and decrypt it upon retrieval from flash.

Neither secrets, KEKs, nor the DEK are are ever exposed on any external interface.


Apartment
Controller
unlock devices
retrieve wrapped DEK and shares of secret
derive KEK from secret and use it to unwrap DEK
Operate: encrypt writes decrypt reads

Recovering the DEK at Startup

SUMMARIZING FLASHARRAY DATA AT REST PROTECTION

In summary, decrypting data stored on a FlashArray device requires:

- Unrestricted access to two more than half the devices in the array’s apartment
- For SSD-equipppped arrays, unlocking the devices at power-on
- Retrieval of the array’s wrapped DEK and sufficient shares of its secret from device header areas
- Knowledge of FlashArray structures—segments, storage blocks, logical pages, and data block descriptors, and data reduction algorithms.

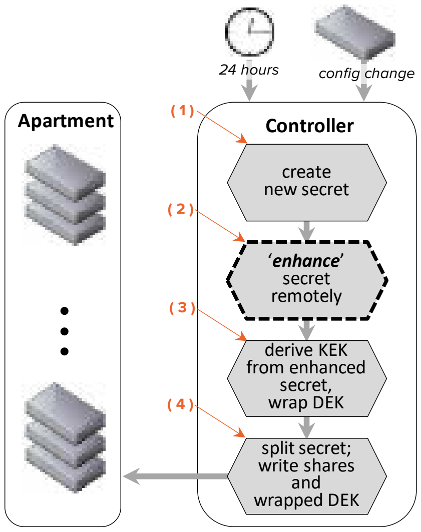
Together, these form a high barrier to scavenging data from individual devices. But if it is possible for an entire array to fall into the hands of an attacker, additional data security may be necessary.

ENHANCED PROTECTION FOR DATA AT REST

To protect against data theft in situations in which an attacker gains unrestricted physical access to an array, Pure Storage offers two optional mechanisms for enhancing data at rest security. The principle behind both mechanisms is to separate part of the information required to recreate an array’s secrets from the array. Each time an array using one of these security enhancers creates a new secret (1), it sends it to the enhancer, which performs a cryptographically strong transformation and returns the result (2). The array derives a new KEK from the enhanced secret and uses it to wrap its DEK (3), but it splits the *unenhanced* secret into shares which it writes along with the wrapped DEK in its device headers (4).

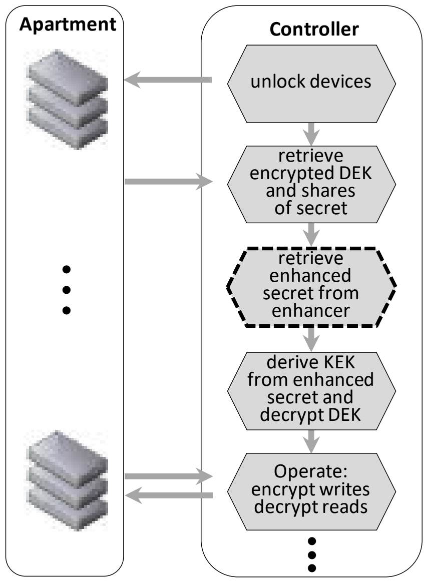
On power-up, an array unlocks its devices, retrieves shares, and recreates the current (unenhanced) secret. It sends the secret to the enhancer, which transforms it and returns it to the array. The array derives the KEK from the enhanced secret and uses it to unwrap its DEK for use.

If the array cannot communicate with the enhancer, it cannot recover the enhanced secret that was used to wrap the DEK, and therefore cannot decrypt stored data.


24 hours
config change
Apartment
Controller
(1)
create new secret
(2)
‘enhance’ secret remotely
(3)
derive KEK from enhanced secret, wrap DEK
(4)
split secret; write shares and wrapped DEK

Refreshing the KEK (Enhanced)

Pure Storage offers two forms of enhanced data at rest protection:


Apartment
Controller
unlock devices
retrieve encrypted DEK and shares of secret
retrieve enhanced secret from enhancer
derive KEK from enhanced secret and decrypt DEK
Operate: encrypt writes decrypt reads

Recreating the DEK at Startup (Enhanced)

SmartCards

The FlashArray *Rapid Data Locking* (RDL) option consists of removable *Spyrus Rosetta Smart Cards* in readers connected to each controller. Users *initialize* Smart Cards with a *passphrase* which passphrase cannot be read.

An array's primary controller sends each secret it generates to its Smart Card reader. The card transforms the secret using its passphrase and returns the result to the controller. The controller splits the original (not enhanced) secret into shares and stores them in flash device header areas.

To unwrap its DEK at startup, a controller reads the wrapped DEK and the shares of its secret, recreates the secret, and sends it to the Smart Card for enhancement. The Smart Card returns the enhanced secret to the controller, which derives the current KEK and uses it to unwrap its DEK.

If an array's Smart Cards are removed and it is restarted, its enhanced secret cannot be recreated, so its KEK cannot be derived, its DEK cannot be unwrapped, and data on its devices cannot be deciphered.

KMIP Server Integration

Some organizations use *Key Management Internet Protocol* (KMIP) servers to manage their data encryption keys centrally. When an array integrated with such a KMIP server generates a new secret, it splits it, stores the shares in device header areas, and sends the full secret (secured by Mutual TLS) to the KMIP server. The KMIP server transforms the secret cryptographically using HMAC-SHA256 and returns the result to the array. The array uses the transformed secret to derive the KEK and uses it to unwrap its DEK.

Each time the array starts up, its primary controller retrieves shares from device header areas, recreates its current secret, and sends it to the KMIP server for enhancement. The server enhances the secret and returns it. The array derives the current KEK from it and uses the KEK to unwrap its DEK. If the KMIP server is inaccessible, the array cannot recover its enhanced secret, so its KEK cannot be derived, its DEK cannot be unwrapped, and data on its devices cannot be decrypted.

Arrays and KMIP servers use TLS Mutual Authentication and encryption to secure communication with each other. At the time of publication, Purity//FA supports the Vormetric Data Security Manager (DSM) KMIP server from Thales e-Security, Inc.

ARRAY DECOMMISSIONING AND PERMANENT REMOVAL OF FLASH DEVICES

When Pure Storage *decommissions* a FlashArray chassis or storage shelf for removal from customer premises, as for example when a proof of concept evaluation finishes or when a hardware update or capacity consolidation is performed, a Pure Storage or partner system engineer issues CLI commands to a controller to "wipe" (erase) all affected flash devices. Prior to erasure, the customer (not the engineer) must delete all host objects, eradicate all volumes, and interact with the array's CLI to explicitly authorize the Pure Storage representative to perform the wipe. The controller performs the wipe by issuing commands to devices that make all stored data and metadata, including the header areas that contain KEK shares and the wrapped DEK inaccessible. After a flash device has been erased, all read commands return zeros until LBAs or flash pages are overwritten.

On request, Pure Storage will provide users with Certificates of Erasure and visual records of successful execution of the procedure. The company maintains a registry of Certificates of Erasure.

Even if by some physical means data remnants could be recovered from a securely erased device, they would have been encrypted by the array's (destroyed) DEK. There would be no way to decrypt either the data remnants or the metadata that associates them with the LBAs to which hosts had written them. Thus, while the details of the FlashArray flash device erasure process differ from the multi-pass overwriting typically used with magnetic disks, the result is the same.

In addition to the decommissioning of all flash devices returned to Pure Storage, the company meets the needs of organizations whose policies require retention of retired storage devices by offering a fee-based program that allows users to take permanent ownership of decommissioned devices.

FLASHARRAY DATA SECURITY CERTIFICATION

At the time of publication, FlashArray FA-400 series systems with Purity//FA version 4, and FlashArray//M70R2 and FlashArray//X90R2, both with Purity//FA version 5 are US Federal Government FIPS 197 certified and FIPS 140-2 Level 1 validated.[2] The Spyrus Rosetta SmartCards included with the RDL option are FIPS-140-2 Level 3 validated.[3] When an encryption device such as the SmartCard or a KMIP-compliant key management server is used with a FlashArray, the encryption device's FIPS 140-2 validation is the applicable FIPS certification for the combined system.

The National Information Assurance Partnership (NIAP, https://www.niap-ccevs.org/) has certified FA-400 and FlashArray//M series arrays that run Purity//FA version 4.7 as being compliant with the Common Criteria Network Device Protection Profile, Version 1.1.[4] Certification of newer array models, Purity//FA versions, and CC Profile versions is in progress.

While no formal certification of compliance with the NIST SP 800-88 R1 standard exists at the time of publication, FlashArray complies with the standard. Compliance is demonstrated in Kroll OnTrack Erasure Verification, which reports results of using the NIST test methodology to erase data from FlashArray storage devices and to verify that after erasure data can no longer be retrieved. The Kroll OnTrack report is available on request from Pure Storage account executives.

[2] https://csrc.nist.gov/projects/cryptographic-module-validation-program/Certificate/2467
[3] https://csrc.nist.gov/projects/cryptographic-module-validation-program/Certificate/2772
[4] https://www.niap-ccevs.org/Product/Archived.cfm?par303=Pure%20Storage%2C%20Inc%2E