August 2023

Modular Storage Key Management for Heterogeneous Hardware and Software

Alex Eisner

Charles Kunzman

Adam Stubblefield

Jon McCune

Smriti Desai

See next page for additional authors

Inventor(s)
Alex Eisner, Charles Kunzman, Adam Stubblefield, Jon McCune, Smriti Desai, Brian Belleville, and HanBin Yoon

Modular Storage Key Management for Heterogeneous Hardware and Software

A server or other computing device can include multiple devices that can store or provide data (called “data sources” in this disclosure). Such data sources include solid state drives (SSDs), hard disk drives (HDDs), network interface controllers (NICs), non-volatile flash storages and chips (e.g., contained on a motherboard), etc. A server can also include multiple processing devices (central processing units (CPUs), graphics processing unit (GPUs), field-programmable gate arrays (FPGAs), or other types of processors) that may be mutually distrustful of each other and may execute independent software stacks (e.g., unique or multiple operating system (OS) instances). A processing device may be capable of driving or managing one or more of the server’s data sources.

It is often important to encrypt the contents of these data sources or the data that is provided by these data sources. For example, an SSD may employ full-disk encryption (FDE) to encrypt the contents of the SSD. The SSD may use a media encryption key (MEK) to encrypt/decrypt its contents. The SSD may be configured to require the SSD to never store the plaintext MEK in nonvolatile memory. The SSD may, however, store an encrypted version of the MEK in nonvolatile memory. In order to encrypt/decrypt the MEK, a key encryption key (KEK) may be used. The KEK may be derived from an access key using a key derivation function (KDF). The processor that manages the SSD may provide the access key on a power cycle of the SSD or another event that requires reauthentication.

There is no one-size-fits-all security implementation for data sources. One processor’s or data source’s configuration may prioritize data durability, which may require all credentials needed to access data stored on, or provided by, the managed data sources to be present on-machine. At the same time, a different processor’s or data source’s configuration may prioritize security against physical theft of the data source and its host machine, and thus, credentials are required to not be on the machine that includes the data source.

Aspects of the present disclosure address the above-noted and other deficiencies by providing a flexible architecture for managing credentials in a server with multiple data sources, even when such data sources have differing security requirements. The present disclosure also provides for access of encrypted data across multiple processing devices with various different security requirements.

FIG. 1 illustrates an example server 100, which includes multiple divisions 110(1) and (2). Each division 110 includes hardware and software. For example, a division 110 may include one or more processing devices 112, one or more memories 114, one or more data sources 120, or other hardware and software. A processing device 112 may drive or manage a data source 120 within its division 110. A memory 114 of the division 110 may store an OS, one or more applications, middleware, services, drivers, etc. A data source 120 may include non-volatile storage 122 and volatile storage 124. In some cases, the non-volatile storage 122 may store an encrypted version of the data source's MEK 130, and the volatile storage 124 may store an unencrypted version of the MEK 132, although not necessarily at the same time. The processing device 112 may use the unencrypted MEK 132 to access the contents of the data source 120.

As can be seen in FIG. 1, a first data source 120(1) may represent a data source 120 configured to prioritize data security and, thus, has a requirement to not store the unencrypted MEK 132 in the non-volatile storage 122. However, a second data source 120(2) may represent a data source 120 configured to prioritize data access and, thus, allows the unencrypted MEK 132 to be stored in non-volatile storage 122.

FIG. 2A illustrates an example process of decrypting an encrypted MEK 130. The processing device 112(2) may generate a node secret seed (NSS) 140. The NSS 140 may include a string of data. The processing device 112(2) may store its NSS 140. In some cases, the processing device 112(2) may generate and store multiple NSSs 140. The processing device 112(2) may provide its NSS 140 to a KDF 210. The KDF 210 may accept, as input, the NSS 140 and other data and may output an access key 212. The processing device 112(2) may provide the access key 212 to the data source 120. The data source 120 may receive the access key 212 as input. The firmware of the data source 120 may use the access key 212 as input to a decryption function 214 to decrypt the encrypted MEK 130 in order to obtain the unencrypted MEK 132. The data source 120 may then store the unencrypted MEK 132 in volatile memory 124 so that the unencrypted MEK 132 is readily available for the processing device 112 or other hardware or software to use to access the contents of the data source 120.

In some cases, a division 110 or a processing device 112 may rotate the access key(s) 212 used in the division 110. Rotating an access key 212 may include generating a new access key 212 and setting that access key 212 as the access key for a data source 120. In some cases, rotation of the access key(s) 212 may be crypto-destructive of the data.

A division 110(1) may interact with another division 110(2), but the divisions 110(1)-(2) may not trust each other to maintain the secrecy of their unencrypted information. One way that the divisions 110(1)-(2) may interact is by encrypting or decrypting an NSS 140 of a processing device 112 in another division 110.

FIG. 2B illustrates an example process for encrypting an NSS 140. In some cases, a first processing device 112(2) may be configured to store its NSS 140 in an encrypted state. The processing device 112(2) may be configured to do this in response to the processing device

112(2) not including a security property (e.g., being metal-detectable). First, the first processing device 112(2) may encrypt its NSS 140 with a key that is stored locally, e.g., on the processing device 112(2). This encrypted NSS 140 may be called the "wrapped NSS" (WNSS) 220. The first processing device 112(1) may then send the WNSS 220 to a second processing device 112(1) of another division 110(1). The second processing device 112(1) may include a security property (e.g., may be metal-detectable). The second processing device 112(1) may use a key stored on a component that includes a security property to encrypt the WNSS 220, creating a doubly wrapped NSS (DWNSS) 222. The second processing device 112(1) may then send the DWNSS 222 back to the first processing device 112(2), and the first processing device 112(2) may store the DWNSS 222 locally. In this manner, the processing device 112(2) can store its own NSS 140, and the NSS 140 has not been encrypted only using keys or other encryption information from the processing device's 112(2) own division 110(2). This process also prevents the other division's 110(1) processing device 112(1) from accessing the unencrypted NSS 140.

To decrypt the DWNSS 222, the first processing device 112(2) sends the DWNSS 222 to the second processing device 112(1). The second processing device 112(1) uses the key to decrypt the DWNSS 222 and obtain the WNSS 220. The second processing device 112(1) sends the WNSS 220 to the first processing device 112(2), which then decrypts the WNSS 220 using its own key. The first processing device 112(2) now has the NSS 140, which it can then use with the process in FIG. 2A to generate the access key 212 to a data source 120 of the first processing device's 112(2) division 110(2).

In one implementation, the first processing device 112(2) may encrypt its NSS 140 with a key stored locally to generate the WNSS 220. The first processing device 112(2) may then store the WNSS 220 remotely. To decrypt, the first processing device 112(2) may retrieve the remote WNSS 220 and decrypt it with the locally stored key.

The architecture of the server 100 can be managed through a combination of hardware and software configurations, which can vary on a machine-by-machine basis. The architecture allows for different processing devices 112 to have different sources from which input is provided to generate access keys 212 for a division's 110 data sources 120.

Another aspect of the present disclosure includes an emergency data recovery procedure. The emergency data recovery procedure may allow access to a data source even if the access key 212, NSS 140, or other data is lost, unretrievable, or otherwise transiently or permanently inaccessible. In some cases, a server 100 or a division 110 may be divided into different security realms. A processing device (which may include a processing device that the server 100 uses to manage the divisions 110(1)-(2) or may include a processing device 112 of a division 110) may generate an asymmetric public/private keypair that pertains to a security realm, and each security realm may have its own keypair. The processing device may store the private key in a secure location, which may be in a secure location of the server 100 or may be on a computing device outside of the server 100. The processing device may distribute the public key to each processing device 112 or data source 120 in the security realm. The processing device 112 or data source 120 may encrypt an access key 212 using the public key. For example, a data source 120 may use the public key to encrypt the access key 212 used to decrypt its encrypted MEK 130, or a processing device 112 may use the public key to encrypt the access keys 212 it uses to decrypt different encrypted MEKs 130. These encrypted access keys 212 ("wrapped encrypted access keys" or "WEAKs") are stored in a configurable location. The location may include the data source 120 to which the access key 212 pertains, a database, partition information, or some other location. In the event of an emergency that calls for data recovery, the private key can be used to recover the access key 212 by using the private key to decrypt the WEAK. Each time an access key 212 is rotated, this process is applied to the new access key 212.

Using security realms mitigates exposure risks if the private key must be accessed, such that even if the private key is compromised, the exposure is limited to the subset of data sources 120 within that private key's security realm. Additionally, the security realm can also employ key rotation such that a new public/private key pair is generated periodically and the access keys 212 are re-encrypted using the new private key. This further limits exposure risk by limiting the time window during which data sources 120 in that security realm might be vulnerable to a compromised private key.

The present disclosure provides a flexible implementation of varying security requirements, allowing for the same system to use data sources that emphasize different security configurations, for example, physical security of managed devices versus isolating different processing devices 112 that share a same common server 100. Physical security mitigates the risk of data compromise if a data source 120 is stolen or physically tampered with, while isolation reduces the risk of breach to one division 110 affecting divisions 110 sharing the same server 100. The present disclosure provides the ability to manage different security requirements for a server 100 composed of multiple hardware and software portions. In many instances, a single division 110 on a server 100 may require multiple OSs to run, and those OSs may or may not be in the same entity's control.

It should be noted that, while FIG. 1 depicts two divisions (i.e., 110(1) and 110(2)), a server 100 may have any number of divisions 110. A division 110 may have any number of data sources 120. A division 110 may have any number of processing devices 112 or other hardware, such as the memory 114. It should also be noted that a division 110 may span across multiple physical servers 100 that may be networked together to form a single logical server 100. Also, different divisions (e.g., the first division 110(1) and the second division 110(2) in FIG. 2B) may be on different servers 100 or machines, which may not be in the same physical location.

Abstract

A server may be divided into one or more divisions, which each may include at least one processing device and one or more data sources. Each data source may include a media encryption key (MEK) that encrypts/decrypts the contents of the data source. The MEK may be encrypted. A processing device of a division may include a node secret seed (NSS) that the processing device may use with a key derivation function to generate an access key. The processing device can use the access key to decrypt the MEK of a data source. To secure the NSS, the processing device may encrypt the NSS, send the encrypted NSS to another processing device outside the division, and the other processing device can encrypt the already-encrypted NSS again and send the double wrapped NSS back to the first processing device. The server may have an access key emergency data recovery procedure.

Keywords: data storage, storage devices, full-disk encryption (FDE), self-encrypting drives (SED), encryption, decryption, server, security, processor, media encryption key (MEK), key derivation, password

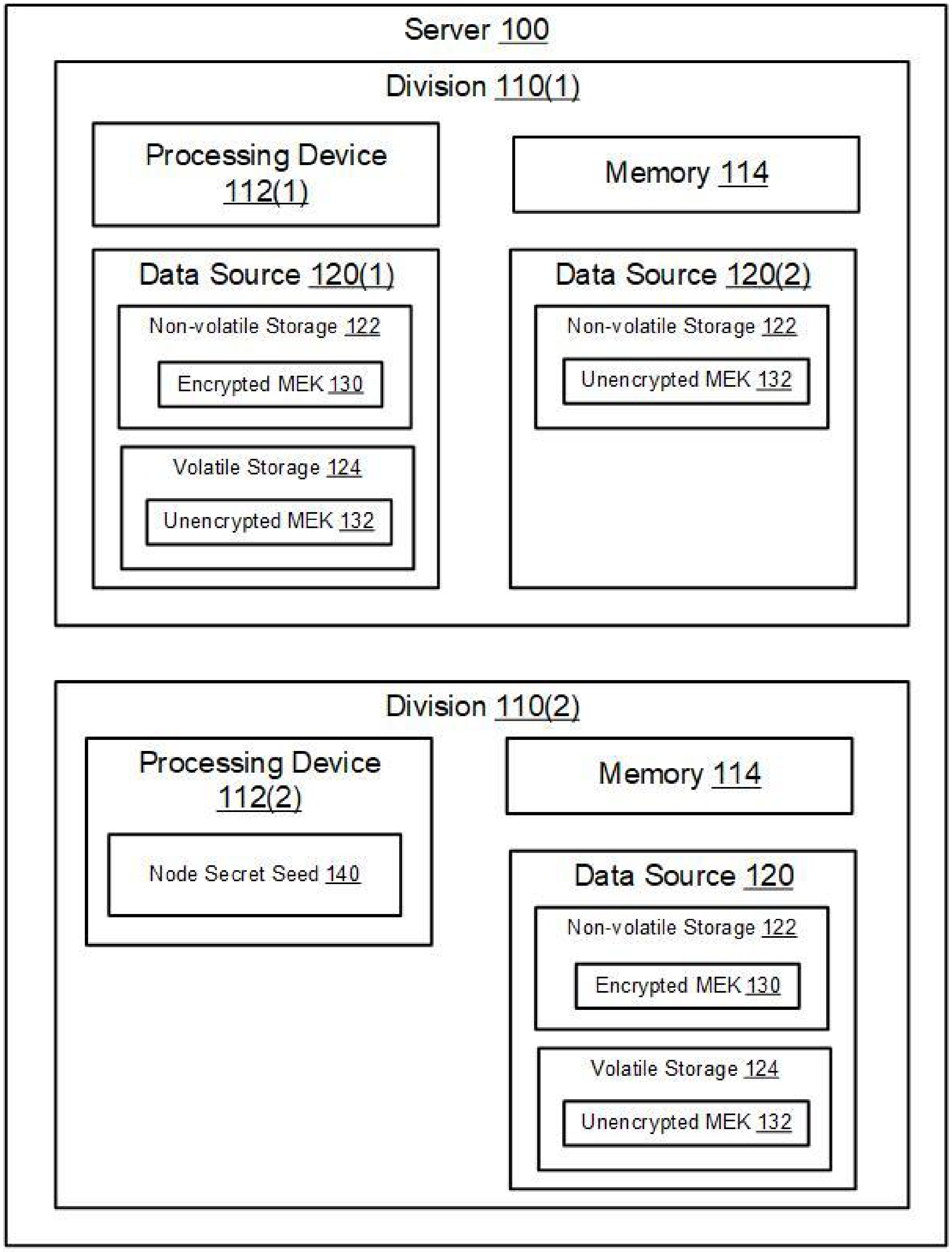
Server 100
Division 110(1)
Processing Device 112(1)
Memory 114
Data Source 120(1)
Non-volatile Storage 122
Encrypted MEK 130
Volatile Storage 124
Unencrypted MEK 132
Data Source 120(2)
Non-volatile Storage 122
Unencrypted MEK 132
Division 110(2)
Processing Device 112(2)
Node Secret Seed 140
Memory 114
Data Source 120
Non-volatile Storage 122
Encrypted MEK 130
Volatile Storage 124
Unencrypted MEK 132

FIG. 1

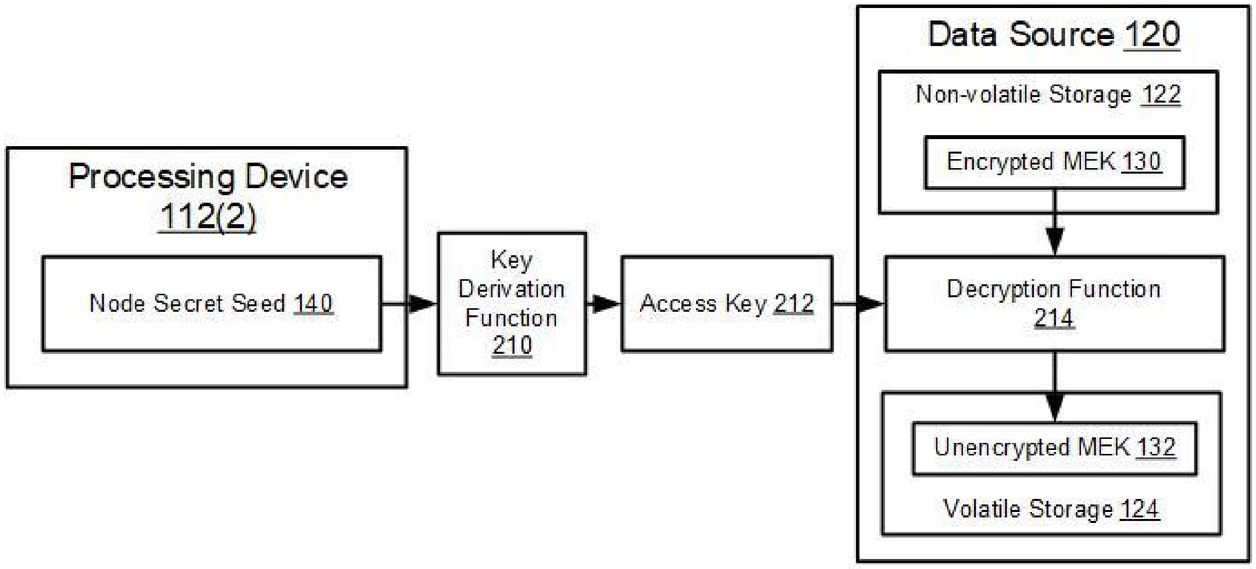
Processing Device 112(2)
Node Secret Seed 140
Key Derivation Function 210
Access Key 212
Data Source 120
Non-volatile Storage 122
Encrypted MEK 130
Decryption Function 214
Unencrypted MEK 132
Volatile Storage 124

FIG. 2A

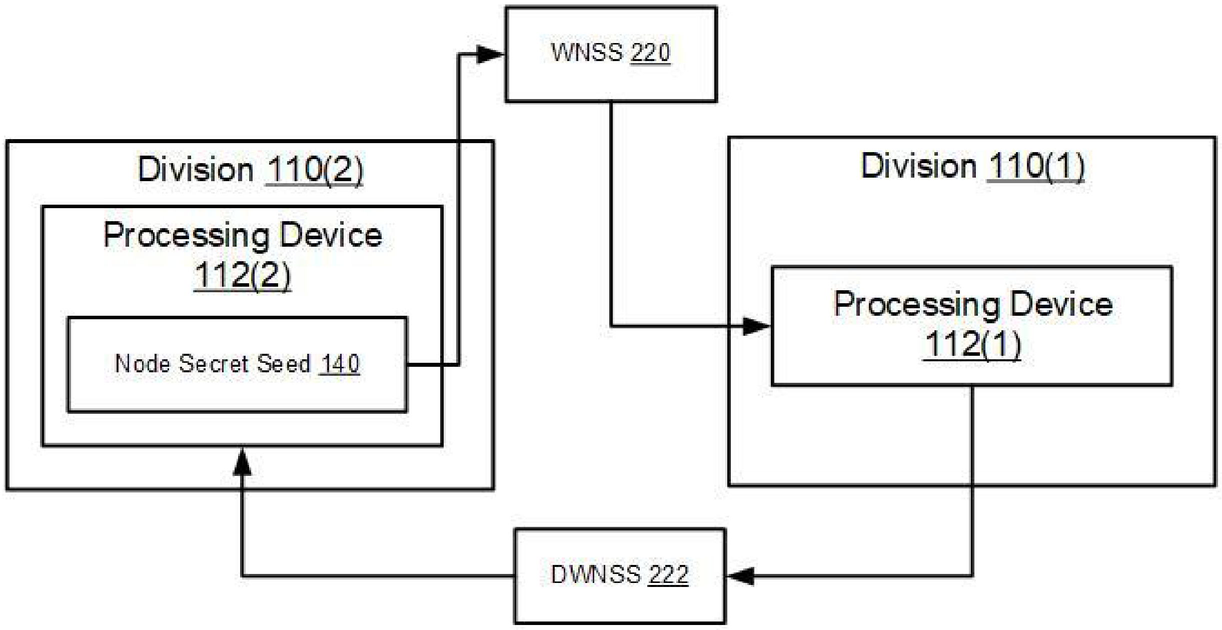
WNSS 220
Division 110(2)
Processing Device 112(2)
Node Secret Seed 140
Division 110(1)
Processing Device 112(1)
DWNSS 222

FIG. 2B