



Enterprise Self-Encrypting Drives

User Guide - Part 1

100515636, Rev. B
September 2015

© 2015, Seagate Technology LLC All rights reserved.

Publication number: 100515636, Rev. B September 2015

Seagate, Seagate Technology and the Spiral logo are registered trademarks of Seagate Technology LLC in the United States and/or other countries. Seagate, and SeaTools are either trademarks or registered trademarks of Seagate Technology LLC or one of its affiliated companies in the United States and/or other countries. All other trademarks or registered trademarks are the property of their respective owners.

No part of this publication may be reproduced in any form without written permission of Seagate Technology LLC.

Call 877-PUB-TEK1 (877-782-8351) to request permission.

One gigabyte, or GB, equals one billion bytes and one terabyte, or TB, equals one trillion bytes when referring to hard drive capacity. Accessible capacity may vary depending on operating environment and formatting. Quantitative usage examples for various applications are for illustrative purposes. Actual quantities will vary based on various factors, including file size, file format, features and application software. Seagate reserves the right to change, without notice, product offerings or specifications.

Contents

1.0	Introduction	3
1.1	The fundamentals of data encryption	4
1.1.1	Encryption basics	4
1.1.2	The Advanced Encryption Standard (AES)	5
1.1.3	Block ciphers.....	6
1.1.4	Cipher Block Chaining (CBC)	6
1.2	Hash functions	7
1.3	Drive locking	8
1.4	The Self-Encrypting Drive (SED)	8
1.4.1	Types of keys.....	9
1.5	Summary.....	9
2.0	SED disk drive overview	10
2.1	Authentication keys (passwords).....	12
2.2	Logical ports	13
2.3	SID (Secure ID).....	13
2.4	MSID (Manufacturer's Secure ID)	13
2.5	PSID.....	14
2.6	BandMaster	15
2.7	EraseMaster	15
2.8	Taking ownership of an SED	16
2.9	Summary.....	17
3.0	Storing the passwords and encryption keys.....	18
3.1	Changing a BandMasterX password.	20
3.2	Summary.....	20
4.0	User data bands.....	21
4.1	Data band rules and guidelines	23
5.0	Firmware download.....	24
6.0	AES encryption algorithm	25
7.0	Asymmetric keys and digital signatures	26

1.0 Introduction

This user guide provides a comprehensive introduction to security and full disk encryption as it is implemented in Seagate Secure™ enterprise Self-Encrypting Drive (SED) models. SED models communicate with a host system using the standard protocol defined by the Trusted Computing Group (TCG), an organization sponsored and operated by companies in the computer, storage and digital communications industry.

Most of the published material on this subject is in the form of standards. Standards are documents which provide the definitive text on the subject and are the ultimate reference for the industry's design and development teams. These documents however are hardly fodder for the inquisitive amateur and are not recommended as an alternative to your favorite night time reading. That's where this manual comes in. If you have to know about encryption and data security as it applies to disc storage, you've come to the right place.

This manual forms Part 1 of the Users' Guide and will introduce and explain the subject matter using a stepped approach to ease you into the terminology used by the data security intellectuals with as little pain and mathematical wizardry as possible. In Part 2 of the Users' Guide, you will find the information necessary to communicate with the drive using the TCG protocol. In short, Part 1 tells you what you can do with the drive and Part 2 tells you how you can do it. If you are interested in the SED User Guide Part 2, Trusted Storage Architecture-Training Manual, please request it directly from your Seagate engineering contact. A Non-Disclosure Agreement is required for Part 2.

If you stay with us all the way to the back cover of Part 1, we can promise you a good working knowledge and understanding of

- Data encryption and decryption
- Hashing
- Symmetric and asymmetric keys
- Digital signatures and secure messaging
- Drive locking
- Cryptographic data erase
- Encryption keys and authentication keys
- Security partitions
- Password and data access management
- Taking ownership of the drive and activating the security features
- User data bands
- SCSI security commands
- Authenticated firmware downloads

If this is what you were looking for, welcome aboard.

1.1 The fundamentals of data encryption

Before we get into the specifics of Self-Encrypting Drives (SEDs) we need to have a working understanding of the encryption process. What is it and how does it work?

1.1.1 Encryption basics

Encryption is a process whereby a **plain text** or **clear text** message is disguised in such a way as to hide its meaning. It stands to reason that this would not be a particularly clever thing to do unless the process could be reversed and the **encrypted text** (also known as **cipher text**) could be **decrypted** (or **deciphered**) back to the original message.

In Figure 1, we see that a clear text message is encrypted by a piece of hardware we'll call an encryption engine and subsequently decrypted back to the original clear text by passing it through a decryption engine. We call these engines because they work on the data as they pass through and perform the required conversion without introducing any noticeable delay in the data flow.

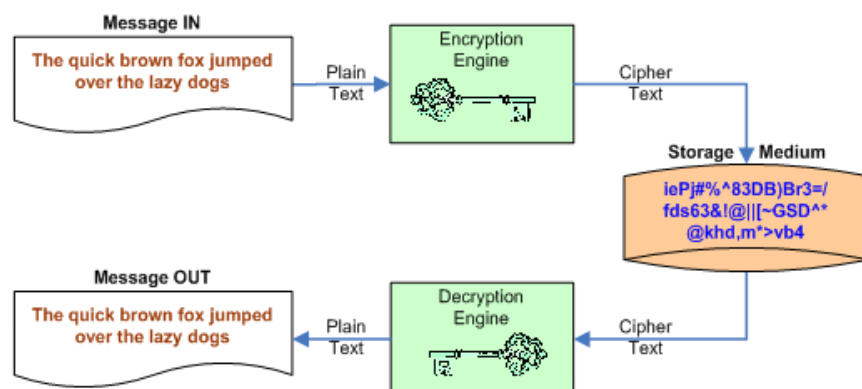


Figure 1. Encryption and decryption

Both the encryption and decryption engines use a key which is the secret ingredient in the text transformation processes. To keep things as simple as possible, this is a **symmetric key**, which means the same key is used for both the encryption and the decryption process. We're all familiar with at least one symmetric key, the one which both locks and unlocks the door to our house.

Here's a very simple example of symmetric encryption. Let's suppose our encryption engine uses a very simple process which will add incoming plain text to a secret key value, letter by letter, and output the result as cipher text. In this case we'll give all letters a value corresponding to their position in the alphabet such that:

"A" = 1, "B" = 2, "C" = 3, "D" = 4, "Z" = 26, " " (space) = 27.

Now if we add "B" (2) to "G" (7) we get "I" (9). If the addition produces numbers which are greater than 27, for example $(N + R) = (14 + 18) = 32$, we simply loop back around the alphabet, $(32 - 27) = 5 = "E"$. Suppose our clear text message is "HELLO WORLD" and we have selected the secret encryption key "ENIGMA" (repeated as required to match the length of the clear text message), then for our very simple encryption engine, the encryption process would provide the following result:

HELLO WORLD + ENIGMAENIGM => MSUSAAAB SQ

Message	Encryption Key	Cipher Text
---------	----------------	-------------

It isn't rocket science to deduce that our decryption engine would need to perform a similar but opposite function (subtraction in this case) in order to reconstitute the original message:

MSUSAAAB SQ – ENIGMAENIGM => HELLO WORLD

Cipher text Decryption Key Message

Any prying eyes that got access to our cipher text would not be able to deduce the original plain text message unless two things were known:

1. The secret key
2. How the encryption engine works—the **algorithm** used to compute the cipher text.

Any cryptographer worth his salt would be able to break our code in a heartbeat, so we could beef it up by making the algorithm more complex. Instead of assigning sequential numbers to the letters of the alphabet, we could use a lookup table, assign any unique number we want to each of the alphabetic characters, process the message in reverse order, add redundant characters, and add any number of other algorithm complexities to make the key more secure. The other thing we could do is to use a more complex key which is as large as manageably possible (to cut down the number of repetitions) and one that does not form a readable word or phrase, making it more difficult for an attacker to break the code. A key made up of random characters would fit the bill since its structure is entirely unpredictable.

To make the encrypting process easier to handle by the electronic hardware, we could perform the encryption on a block by block basis. Additionally, if we made the key size equal to the block size, we could avoid having to repeat (concatenate) the key within the block. This is the technique that is used and though block sizes vary depending on the algorithm and the current state of the technology, the more common block sizes in Self-Encrypting Drives are 128 and 256 bits.

1.1.2 **The Advanced Encryption Standard (AES)**

Believe it or not, in the real world of disk drive cryptography, the only secret is the encryption key itself. The algorithm (encrypting process) is not only well known but is a standard called the Advanced Encryption Standard (AES) which is recommended by the US government.

Two versions of this standard are used in Seagate disk drives, **AES128** and **AES256**. The numbers refer to the bit-size of the encryption key (and the block size) used by the algorithm, which must be a 128-bit (16 byte) or 256-bit (32 byte) **random number**. Without knowing the encryption key, this algorithm makes it virtually impossible to decipher the code and since the algorithm is in general use, the more exposure it gets to being unsuccessfully attacked and broken, the higher our confidence in it.

Another advantage of being a standard is that it provides a common denominator for the manufacture of encrypting devices. So, all vendors are dancing to the same tune—this makes it easier to check that all vendors of encrypting hardware are compliant with government requirements.

For those interested in reading a short description of the AES128 algorithm and seeing a simplified block diagram, refer to Section 6.0.

1.1.3 Block ciphers

As mentioned in our discussion on AES, we encrypt the clear text message in blocks of 128 or 256 bits at a time. In other words, we are using a block cipher in our encryption engines.

Figure 2 shows how a simple block cipher called **Electronic Code Book (ECB)** constructs the encrypted data. Each block of plain text (P) is encrypted with the key (E) and outputs cipher text (C) ready for storage. Since we are using the same encryption key for every block of data, we would expect identical blocks of clear text to produce identical blocks of cipher text. This is clearly undesirable since it provides an attacker with a clue that could be used to determine the key.

If attackers are able to manipulate the clear text and if they can view the resultant cipher text, they could use small and precise changes in the clear text, see what affect that had on the cipher text, and use this information to help identify the key. But don't be alarmed, the word "if" appears twice in the last sentence. The first would normally not be satisfied and the second would never be satisfied.

As we will see later, the encrypted data never leaves the drive and is not available over the I/O (interface). Nevertheless, this characteristic of ECB is seen as a weakness which has been addressed and rectified by **Cipher Block Chaining**.

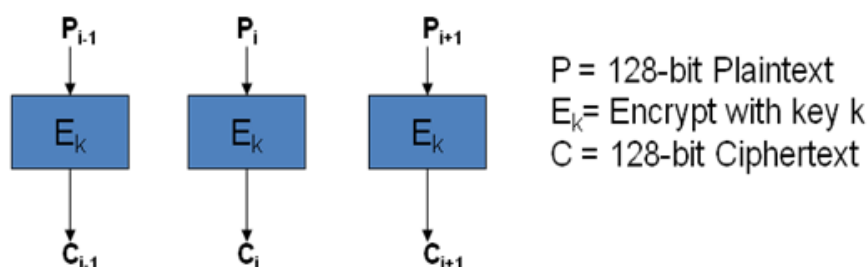


Figure 2. Electronic Code Book

1.1.4 Cipher Block Chaining (CBC)

As you can see from Figure 3, CBC is similar to ECB except that the cipher text from the previous encryption block is XOR'd with the plain text in the current block. This effectively randomizes the clear text in every stage and prevents cipher text duplications. Ah yes, but what about the first stage? Clearly there is no cipher text available to the first stage, so we've compensated for this by adding a component called the **Initialization Vector (IV)**. This isn't magic, it's simply a secret 128-bit number known only to the disk drive.

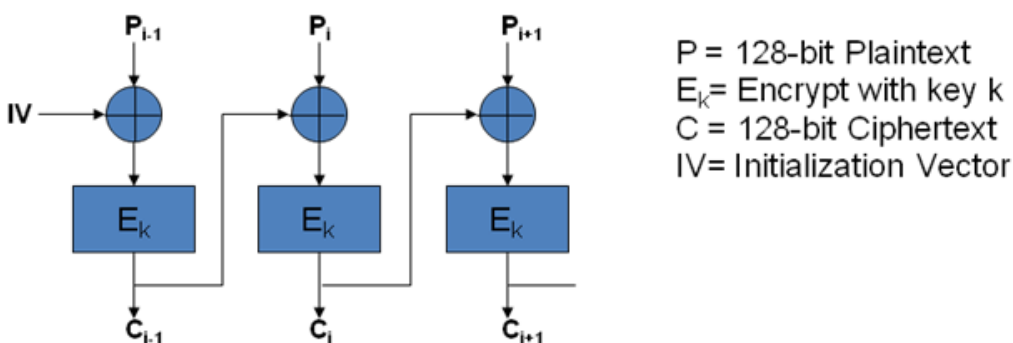


Figure 3. Cipher Block Chaining

Now that we know something about AES and CBC, there's one more subject that we need to visit before we take a look at how the disk drive applies and manages encryption in data storage. That subject is **hash functions**.

1.2 Hash functions

Hash functions take an arbitrarily long string of bytes and produce a fixed size result, sometimes called the **digest** or fingerprint. In Figure 4, we see a string of bytes **m** being input to a hash function which produces a fixed size hash output **h(m)**.



Figure 4. The hash function

The properties of a hash function can be simply stated as follows:

- It must be a one-way function. Given **m** it's easy to compute **h(m)** but given **h(m)** it's not possible to find **m**. That is, you cannot create the original message from the digest.
- It must have good collision resistance. A collision means computing the same hash $h(m)$ for two different input streams. In other words it should be practically impossible to find two messages **m1** and **m2** such that **h(m1) = h(m2)**.

Hash functions have been used in disk drives since the early days of magnetic storage. They have been used as an integrity check on data fields to signal an error condition when data corruption has taken place.

There are various hash functions in use but one of the more common provides a digest called Cyclic Redundancy Check (CRC). Today, we append a 4-byte CRC check to Fibre Channel and Serial Attached SCSI (SAS) data frames before they are sent by a controller across the interface to the drive. As the data frame is received, the drive computes the CRC (hash) of the data and compares it to the appended CRC. Any discrepancy indicates the data has been corrupted and the frame should be resent.

So, how do we select a suitable hashing algorithm that adequately meets the properties given above? Once again, the federal government made our job of selecting a good hashing function a lot easier because the NSA (National Security Agency) designed the **Secure Hash Algorithm (SHA)** which fits the bill very nicely. The SHA algorithm used in Seagate drives produces a digest of 256 bits and is consequently called SHA256.

1.3 Drive locking

In the previous section we showed that all data from the host is encrypted prior to being recorded on the media with a write command, and decrypted to plain text before being returned to the host in a read operation. So where's the security in that? Well the answer is that there is no security implicit in the encryption process when anyone can have access to the drive, or more specifically the data encryption key (DEK).

What do we know about the DEK? Well, we know that it's a random number, and we'll find out shortly that it never leaves the drive. In other words, no one knows the value of the DEK. So now what? How do we stop anyone from accessing the data? The answer is that even though we don't know the value of the DEK, we can tell the drive to prevent anyone from using it unless they have appropriate authorization. In other words the drive owner can put access to the DEK, and hence access to the data, under password (authentication key) control. This is called drive locking.

The drive keeps track of the locked status of the data in a portion of the security electronics called the "Locking Security Partition¹" or "Locking SP." To lock/unlock the user data we would need to talk to the Locking SP, prove that we have the authority to access the data by providing the correct authentication key, and set the locked/unlocked parameter to the appropriate value. When the drive is powered off and on, it will automatically go into the locked state and will require the user to present the correct password to unlock the drive for data access. The Locking SP allows us to control access to other things in the drive, but more about that in due course.

There is a second security partition in the drive called the "Admin SP." Access to this security partition is controlled by another authentication key which allows the owner to set up access permissions to other functions within the drive which might otherwise compromise drive security. For example, the Admin SP allows the owner to enable or disable firmware download operations. Since a firmware download could be thought of as a brain transplant for the drive, strict precautions need to be taken, not just in allowing the download in the first place, but ensuring that the firmware is genuine, it came from the right source, and it is compatible in form, fit, and function.

1.4 The Self-Encrypting Drive (SED)

Self encryption simply means that all of the data written to the storage medium is encrypted by the disk drive before being written and decrypted by the disk drive when it is read. All interface data passing between the host controller and the disk drive is in clear text and therefore, as far as read/write operations from the host controller are concerned, it's business as usual. As shown in Figure 5, the encryption engine is located between the drive's interface electronics and the data buffer (cache) so that all information which is temporarily at rest in the data buffer is encrypted, as is all data on the media side of the encryption engine.

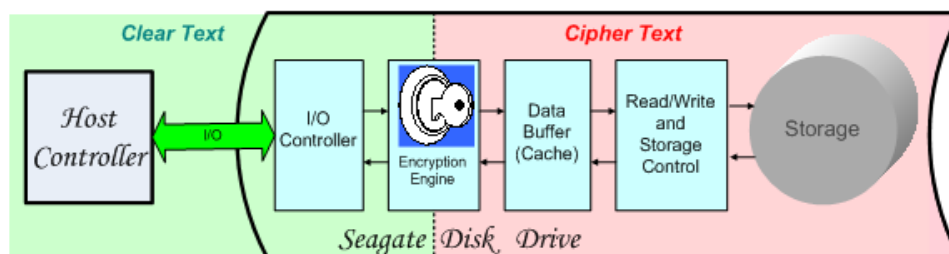


Figure 5. Division of clear text and cipher text in the disk drive

1 Also called "Locking Security Provider" in the standards documents.

The principal reason for employing encryption with locking is to provide security for the data that is recorded on the disk surfaces. This is usually referred to as **security of data at rest**. Simply put, if the disk drive loses power, as would be the case if it were removed from its owner's system, its recorded data will be locked against unauthorized access as soon as power is reapplied. With power reestablished, the host system will need to prove to the drive that it is the owner by providing the drive with the appropriate ownership **credentials**. Credentials? It's time we talked a little bit more about keys and passwords.

1.4.1 Types of keys

There are two types of keys used in SEDs, the data encryption key and the authentication (locking) key.

1.4.1.1 The encryption key

We've already become quite familiar with the encryption key. This is a symmetric key that is used by the encryption engine to generate the unique cipher text from the plain text and to subsequently recover the plain text from the cipher text. It is generated by the drive as a 32-byte random number and is a secret known only to the drive.

The encryption key cannot be accessed by an external source but it can be changed by the drive's owner. Changing the encryption key is not something that should be done lightly because, when changed, all of the data previously written to the drive will now be read and decrypted with a different key and therefore will become unrecoverable. On the other hand, this is an effective technique for destroying data on the disk and is commonly referred to as a **cryptographic erase**—a useful tool if a drive has to be repurposed (used in a different application where the data is neither required nor wanted) or scrapped.

1.4.1.2 The authentication key

This key is used to lock and unlock a function or feature in the drive that requires the owner's permission. It is an owner's credential though we would probably be much happier if we simply called it a **password**. Although there is only one encryption key, there can be several passwords assigned to each drive. As well as needing a password to be able to read and write data on the drive, we've just seen another good example of a drive feature that calls out for password protection—the cryptographic erase. We'll find and explain some other needs for passwords as we dig deeper into encryption and security.

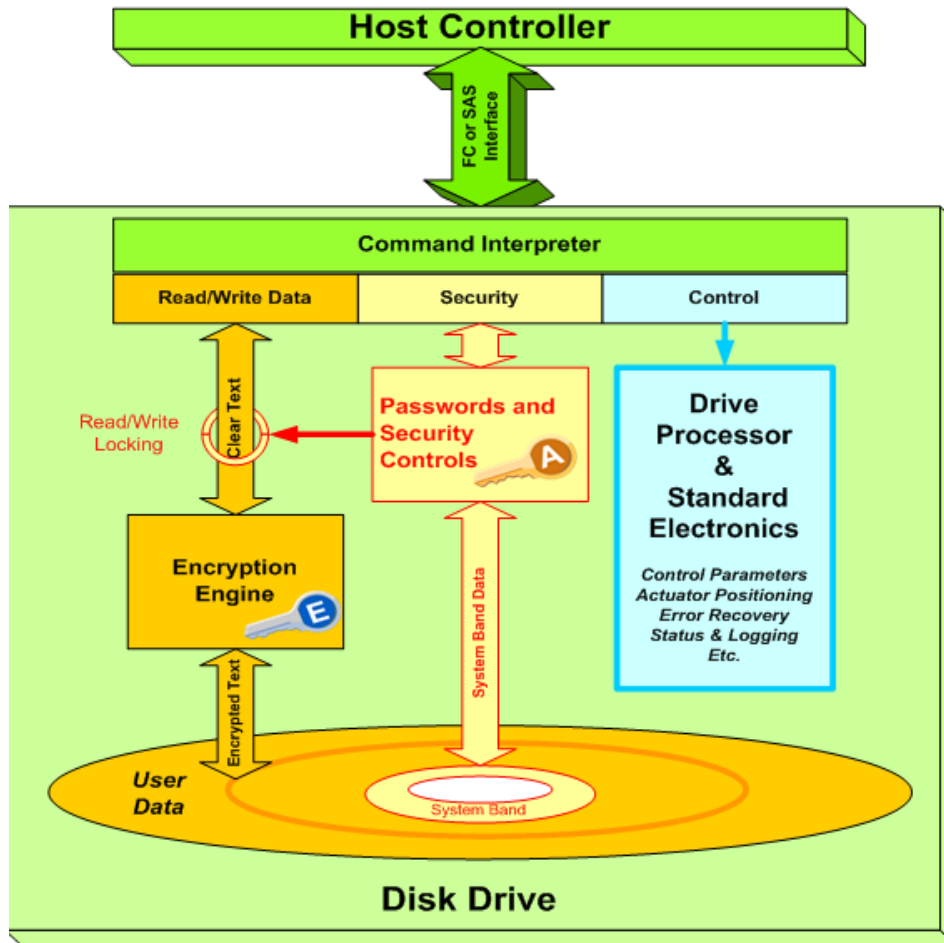
1.5 Summary

By way of introduction to Self-Encrypting Drives, we've taken a quick look at encryption and hashing, in particular the government approved Advanced Encryption Standard (AES), SHA-1 and SHA256. We know that AES is used in Seagate disk drives and that we employ a block encryption mode called Cipher Block Chaining. Finally, we've seen that there are two types of keys used in Self-Encrypting Drives, encryption keys and authentication (or locking) keys, each of which plays a part in providing security for the stored data. Encryption on its own does not provide protection against unauthorized access to the data, for that we need a couple of gatekeepers in the drive. We call these gatekeepers the "Locking SP" and the "Admin SP."

In terms of storing data, I/O communication, drive format, and operational performance, an encrypting drive performs in exactly the same fashion as a non-encrypting drive. In fact, if an SED is installed in an older (legacy) system, the system will function normally and will be unaware of the hidden talents the newcomer has to offer.

Note. We'll see later that this last paragraph is not totally accurate—there are a couple of commands that are not allowed in SED drives but these commands are normally used only in a diagnostic environment.

Now let's take a closer look at how the security features are integrated into an Enterprise SED.



The system band is shown in Figure 6 and comprises an area at the inner radius of the media which is used by the drive to store the information needed to manage its security.

Also in Figure 6 we see that the SCSI (Small Computer Systems Interface) commands from the host controller can be divided into three main categories:

- a. *Control commands.* These cause the drive to perform particular functions (e.g. Drive Reset, Start Spindle), to query or change drive parameters (e.g. Mode Sense, Mode Select), to read status or logging information (e.g. Inquiry, SMART), to provide diagnostic functions, and so on.
- b. *Read and write commands.* They allow access to the media and cause user data to be read or written.
- c. *Security commands.* These are used to define and authenticate the users who may access or erase the stored data, modify security parameters to suit a particular application or installation, or perform special operations such as firmware downloads.

As far as (a) and (b) are concerned it's pretty much business as usual whether we're talking to an SED or not, but (c) is a different matter all together.

How does the host controller communicate with the security block of the drive? Like the rest of the activity on the I/O, the dialogue is handled by SCSI commands and responses.

Two new SCSI commands have been defined by the T10 INCITS (International Committee for Information Technology Standards) Technical Committee in conjunction with the Trusted Computing Group (TCG). These commands are called "**Protocol Out**" and "**Protocol In**," and between them they carry all the security information between the drive and the Host. All security instructions and related information going to the drive are contained in the Protocol Out command and all security information required from the drive must be invoked by the host in a Protocol In command. Please note that these commands deal only with security related information. The normal command dialogue (including Read and Write) is handled by the standard SCSI command set. We'll take a closer look at the construction of these Protocol commands in Part 2 of the Users' Manual.

The security block handles the drive's passwords and makes sure that only the owner has access to the stored data. When the drive is powered up, it is in a locked state and all media access is disallowed until a password has been received by the drive and has been validated against the owner's password which is stored on the system band. Once the validation is complete, the drive retrieves the encryption key from the system band and loads it into the encryption engine. The drive is now unlocked and normal read/write activity can take place.

As we will see shortly, there are additional passwords managed by the security block. These control things like firmware downloads, access to diagnostic resources, and cryptographic erasure of the data. In general, the security block only affects operations which are related to user data.

In the locked state, the drive will still accept commands which do not involve access to user data such as Inquiry, Mode Page, Spindle Start/Stop and so on. These commands are grouped under "Control" in Figure 6 and are managed by the block entitled "Drive Processor & Standard Electronics."

Table 1 provides a list of all the commands which are monitored by the security block and shows how the drive reacts to these commands when it is locked and when it is unlocked. Note that with the exception of Read Buffer, all of these commands attempt to access the media and therefore (with no access controls in place) could constitute a direct threat to the security of user data.

Table 1: Data access command restrictions

Command	User Data Accessed? (Y/N)	Drive is unlocked	Drive is locked
PRE-FETCH	Y	Executable	Check condition
READ BUFFER (Mode 1Ch) ¹	Y	Executable	Check condition
READ BUFFER	N	Check condition	Check condition
READ LONG	Y	Check condition	Check condition
READ	Y	Executable	Check condition
REASSIGN BLOCKS	Y	Executable	Check condition
VERIFY (BYTCHK=1)	Y	Executable	Check condition
WRITE AND VERIFY	Y	Executable	Check condition
WRITE BUFFER ² (except modes 4h - 7h, 0Eh, 0Fh)	Y	Check condition	Check condition
WRITE LONG (WR_UNCOR = 0)	Y	Check condition	Check condition
WRITE LONG (WR_UNCOR = 1) ³	Y	Executable	Check condition
WRITE SAME	Y	Executable	Check condition
WRITE	Y	Executable	Check condition

¹Read Buffer (1C): Return Error History (No user data involved)

²Write Buffer (4h - 7h, 0Eh, and F0): Variations on Download Microcode (no user data involved)

³Mark the LBA as containing a pseudo unrecovered error (no user data involved)

When the drive is unlocked, the commands shown in table 1 are once again in play, but what's the deal with the four commands that continue to react with a check condition even when the drive is unlocked? These commands are Read Buffer, Read Long, Write Buffer and Write Long.

Read/Write Buffer

As we saw in Figure 5, all of the data in the buffer (cache) is encrypted and since access to the cipher text is not allowed in encrypting devices, these commands have been removed from the SED's repertoire.

Read/Write Long

These commands allow the cipher text on the media to be manipulated and viewed and therefore, like the Read/Write Buffer commands, they have been removed from the SED's executable command set.

Note. The Federal Government does not permit storage devices to be operated as coding machines where a clear text input provides a viewable cipher text output. Such devices would fall under the category of "munitions."

2.1 Authentication keys (passwords)

We mentioned above that the security block in the drive manages more than one password. Well, now it's time to take a closer look at these passwords and what they do.

There are two security partitions (SPs) defined in the drive:

- Admin SP
- Locking SP

We visited these briefly in Section 1.3 when we talked about drive locking. The Admin SP controls access to the **logical ports** on the drive and the Locking SP controls access to the user data on the media.

2.2 Logical ports

Some specialized functions within the drive have been assigned access via a logical (or virtual) port. These functions have no physical port on the drive that may be accessed through a connector. They provide access to maintenance or diagnostic resources within the drive and are not used during normal operation. These ports do not provide access to user data and may be locked or unlocked by the owner (the password holder) at any time. In addition, they may be conditioned to lock automatically following a drive power cycle. The only logical port which will concern us is the Firmware Download Port, which must be unlocked before firmware updates will be accepted by the drive. We'll talk about firmware downloads in detail in Section 5.0 of this Users' Guide.

There are four authentication keys needed in order to gain access to all of the drive's operational resources. These are 32-byte passwords which are identified by the credential names **SID**, **MSID**, **PSID**, **BandMaster** and **EraseMaster** as shown in Figure 7. We'll take a look at each of these in turn:

2.3 SID (Secure ID)

This password is also known as the owner's password or owner's credential. It is used to lock and unlock the firmware download port. It can also be used to prevent a drive from being switched into the Diagnostic State (see the MSID discussion below) for advanced failure analysis.

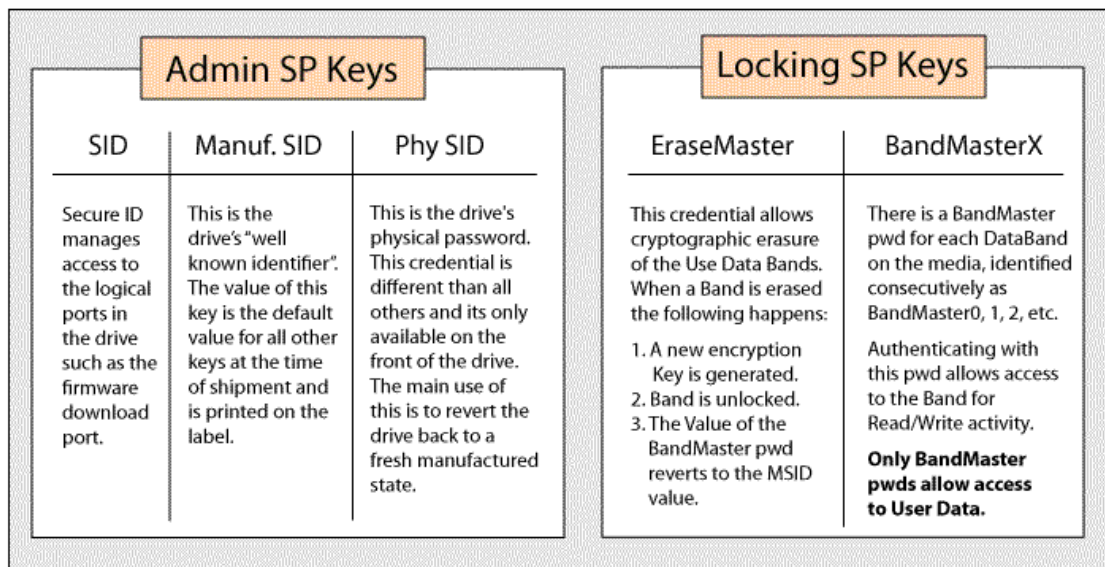


Figure 7. Security partitions and associated authentication keys

2.4 MSID (Manufacturer's Secure ID)

This password is assigned by Seagate during the manufacturing process and is a password that cannot be changed by the host system. When the drive is shipped from Manufacturing, all the other passwords namely SID, BandMaster and EraseMaster are set to the value of MSID which can be obtained electronically from the drive across the interface.

When the drive is delivered, the new owner should personalize the drive by defining new passwords for SID, BandMaster and EraseMaster. Failure to do so means that anyone can use the MSID to preempt the owner and take control of the drive. Such an attack on the drive is known as Denial of Service (DoS) since the rightful owner has been locked out.

2.5 PSID

PSID is the Physical owner of the drive (Physical SID). This password is assigned by Seagate during the manufacturing process and is a password that cannot be changed by the host system. The PSID credential:

- a. Is printed in readable format on the base of the drive label as shown in Figure 8.
- b. May also be provided on the label in bar code form.



Figure 8. Sample drive label

PSID can be used to invoke a RevertSP command which will result in the drive being reverted back to a default state and will complete a crypto erase on the entire drive. This will result in the drive acting like it just spun up new for the first time.

PSID together with a Makers Credential, whose value is known only to Seagate, can cause the drive to be switched into the diagnostic state. This state would be used by Seagate to perform advanced diagnostics on drives which have been returned under warranty for failure analysis. When a drive is switched to the diagnostic state by Seagate, the switch causes an automatic cryptographic erase to be performed and all user data is lost.

It is important to note here that drives returned under warranty will be locked against data access as soon as they are removed from the host system and power is lost. Access to user data is therefore only possible under password control. The automatic cryptographic erase (change of encryption key) performed during the switch to diagnostic mode is an added precaution to assure customers that access to their data is impossible during advanced diagnostic analysis.

If customers are still unconvinced that their data is secure, they can use the owner's password (SID) to disable the Makers Credential. This is not an action that should be undertaken lightly however since it will prevent the drive from being switched into diagnostic mode and hence disable any attempt at failure analysis and subsequent re-disposition.

2.6 BandMaster

This password is part of the Locking SP and controls access to the user data on the media. On a non-encrypting drive, the media contains one contiguous LBA space which starts at LBA 0 and ends at LBA max. On SEDs, the LBA space is divided into two or more data bands, each of which has its own password. That's why, in Figure 7, we call this password BandMasterX where X is the band number and takes the value 0, 1, 2, 3, and so on depending on how many bands are available on the particular disk. A disk with 16 data bands would have 16 BandMaster passwords numbered BandMaster0 through BandMaster15. All BandMaster passwords default to the value of MSID on new drives. We'll dig deeper into the subject of data bands in due course.

2.7 EraseMaster

The host needs to use EraseMaster in order to perform a cryptographic erase on a data band. Although there may be multiple data bands, the same EraseMaster password is used for each of them, but they must be erased one at a time by specifying the band number with the erasure request. Cryptographic erasure of a band causes the following actions to take place:

1. A new data encryption key is created for the band in question (cryptographic erase)
2. The band is unlocked for reading and writing. After a cryptographic erase, any data read from the media would be decrypted with the new key. This means that a Read to an LBA which has not been written with the new key will result in gibberish being returned to the host.
3. The BandMaster password reverts to the value of MSID

In order to discourage a brute force attack on a BandMaster password, a **Try Limit** and **Persistence** setting may be specified by using the EraseMaster credential.

The drive has a counter which keeps track of the number of unsuccessful authentication attempts. The Try Limit sets an upper bound to this counter. This becomes the number of times the drive will accept a false password before shutting the drive down against further attacks. A Try Limit of 0 means there is no limit. The default value is 1024.

The Persistence setting determines whether the count persists through a power cycle. What does "shutting the drive down" mean? Well, that depends on the Persistence setting which has been selected. When the drive shuts down it essentially becomes unresponsive to any further authentication attempts. If the count does not persist through a power cycle, it will be reset to zero and another round of authentication attempts can be made until the Try Limit is once again reached. If the count persists through a power cycle it is not reset to zero and once the Try Limit is reached, the drive will lock out all further authentication attempts whether the drive is power cycled or not.

Regardless of the Persistence setting, if a successful authentication is made at any time before the Try Limit is reached, the counter is reset to zero.

2.8 Taking ownership of an SED

When the drive is shipped from the factory, all of the user configurable credentials are set to the value of MSID, and the drive is unlocked ready for installation in the new host system. The drive will perform as a standard non-encrypting drive at this time with unrestricted access¹, so the first thing the new owner should do is to personalize the drive to preclude the possibility of malicious access and possible **Denial of Service (DoS)**.

To personalize the SID, follow the procedure² shown in Figure 9.

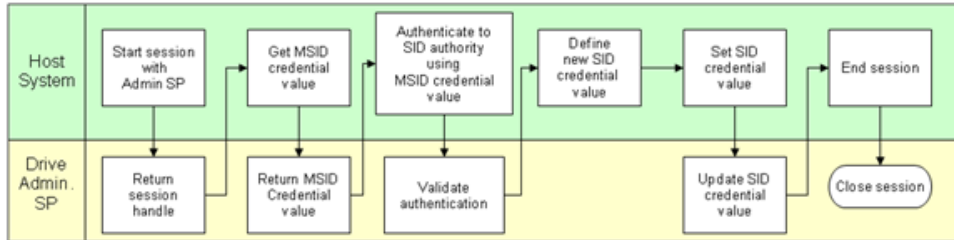


Figure 9. Changing the SID credential value

The host starts a session with the drive's Admin SP and asks the drive to provide the value of MSID, which is the current default value of SID. Now that SID is known, the host can access the Admin SP's credential table and change the value of SID as required. The value of SID is now a secret known only to the drive owner.

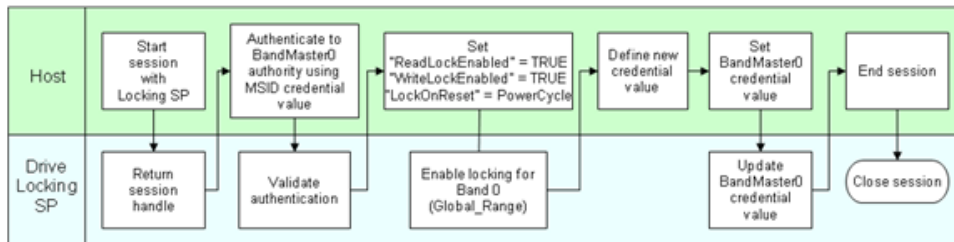


Figure 10. Setting up a data band

¹ The download port (logical port) defaults to the locked state and the R/W Long and R/W Buffer commands are not available.

² The protocol for the procedures shown in Figures 9 and 10 is covered in detail in the Self-Encrypting Drive Users Guide, Part 2.

A similar process is used to change the credential values (passwords) for the Locking SP rather than the Admin SP, however a look at Figure 10 will show that there are some additional parameters that need attention with BandMasterX.

The *ReadLockEnabled* and *WriteLockEnabled* parameters do not lock the data band in question, they merely enable the locking mechanism and allow the parameter *LockOnReset* to be effective. When this parameter is set to *PowerCycle*, the data band will lock automatically when power is removed and reapplied.

Though not shown here, the drive can also be locked and unlocked by setting the parameters *ReadLocked* and *WriteLocked* to *TRUE* or *FALSE*.

One further point, the *ReadLockEnabled* and *WriteLockEnabled* parameters must be set as a pair such that they must both be *TRUE* or they must both be *FALSE*. The drive will ignore any attempt to set a *TRUE/FALSE* or *FALSE/TRUE* combination. The same applies to the *ReadLocked/WriteLocked* pair.

In an Enterprise application, the assumption is made that the operating environment is secure and that band locking is only really required when the drive is removed from its cabinet (causing loss of power). Failure to ensure that the data band will lock on a power cycle allows access to user data following removal of the drive from its system enclosure.

If there is any suspicion of malicious activity during normal operation of the drive, the *ReadLocked/WriteLocked* pair can be used to lock and unlock the data band as required.

2.9 Summary

An SED and a standard drive have the same hardware and with a few exceptions respond to the same SCSI command set, however the SED goes through additional manufacturing processes which install security partitions on the drive and enable the encryption engine. No one is allowed access to the cipher text on the media or in the data buffer, and therefore the SCSI commands that would allow this access are not available on SEDs. When shipped from the factory, all drive passwords are set to the value of the MSID credential which is printed on the label or may be obtained electronically from the drive. On receipt of a new drive, the owner should change all passwords to personalize the drive in order to prevent a DoS attack.

3.0 Storing the passwords and encryption keys

Since we encrypt the user data before storing it on the media, we obviously don't want to drop the ball when it comes to storing the drive's keys by committing them to storage in plain text format. So what do we do to protect the integrity of the owner's passwords and the data encryption keys? Well, let's find out by taking a close look at Figure 11.

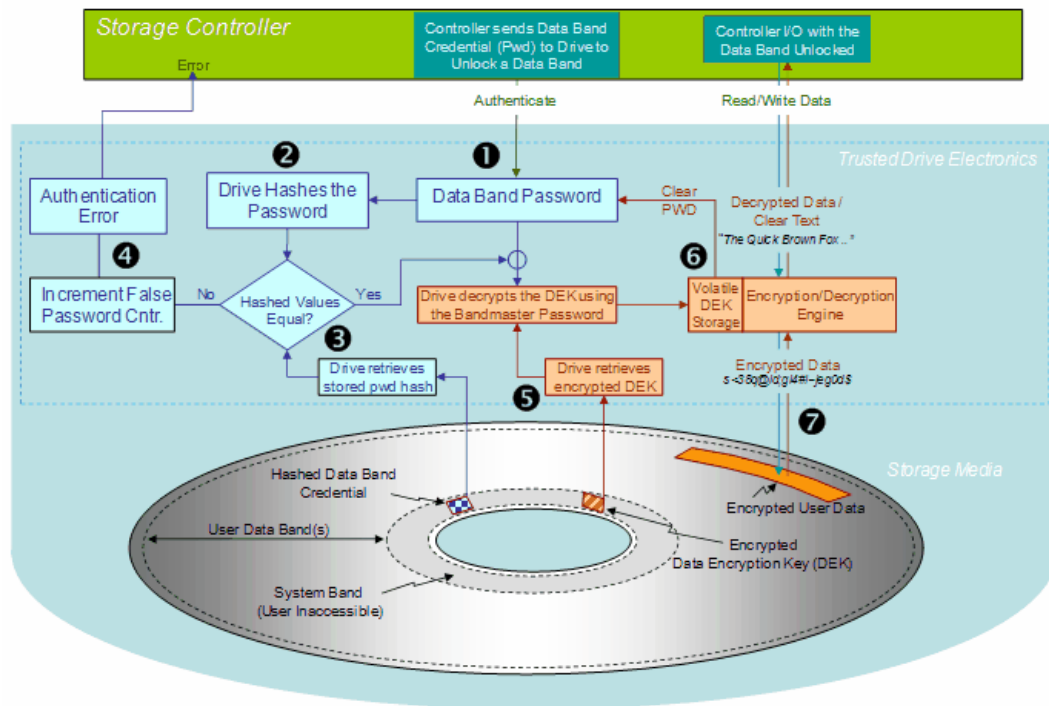


Figure 11. Password and encryption key storage

1. The controller sends a BandMaster password to the drive as part of the authentication process. The password is in clear text and is held in temporary volatile storage for a few milliseconds until the authentication process is complete.
2. The drive hashes the password and forms a SHA256 digest.
3. The drive retrieves the previously-stored password digest from the system band and compares it to the value obtained in (2).
4. If the hashed values are not equal, the drive will inform the controller of an authentication error and will increment the false password counter.
5. If the hashed values are equal, the drive will retrieve the encrypted data encryption key from the system band and decrypt it to clear text using the BandMaster password from the controller as the decryption key. Since authentication has been successful, the drive will clear the false password counter.
6. The drive loads the clear text value of the encryption key into the encryption engine and clears the BandMaster password from temporary storage.
7. With the password verified and the encryption key loaded, the drive can now respond to Read/Write commands from the host.

There are two important points that should be emphasized from the foregoing discussion. The first is that the drive protects the integrity of the user's passwords by storing only their hashed values. This is true for SID and EraseMaster as well as the set of BandMaster passwords. Since a hash is a one-way function, there is no way to recover the clear text password from the stored hash.

The second point is that the data encryption key (DEK) is itself encrypted with the BandMasterX password before being stored on the media. This is significant because losing the password for a data band means that the stored DEK cannot be decrypted and that is tantamount to a cryptographic erase. In other words, there are two ways to perform a cryptographic erase on a data band

1. Tell the drive to change the encryption key
2. Destroy the BandMasterX password¹

Before we move on, there is a particular scenario involving the DEK that is worthy of our attention. Suppose the drive owner decides that the drive should not lock the data bands after a power cycle. This could be achieved by setting the parameter *LockOnReset* = []² rather than the customary *LockOnReset* = [*Powercycle*]. Now, if the drive is moved from one system to another its data would be accessible as soon as power is reapplied. Since this leaves the drive wide open to unauthorized access, it's hardly a recommended procedure, but more than that, it causes a problem with the encryption key. Why? Well let's take a closer look.

Normally, after power up, the host will send the band password to the drive to unlock the band and allow the drive to recover the DEK, as we saw in Figure 13. But if the data band is already unlocked when the drive is powered up, the host doesn't need to send the password and since the drive can't derive the password from the stored hash, how do we decrypt the DEK? Fortunately the drive knows that it will be faced with this dilemma as soon as *LockOnReset* is set to *null*. The drive could simply get the DEK from the encryption engine and store it on the media as plain text. This would solve the problem but it breaks the rule that says we are not allowed to store unencrypted keys on the media. "Why not?", you say, "the band is unlocked and we can read the data, so why bother about an unencrypted key?" Good point. Well, because an unscrupulous but resourceful hacker might find a way to read the DEK and use it to compromise your data should you decide to lock the band at a future date. Since storing an unencrypted DEK is not the solution, the drive will use an alternate technique to disguise the key prior to storing it on the media.

To relock the data band, the host must first authenticate³ to BandMasterX and the DEK encryption process reverts to normal.

1 In this case, since the password is lost, the owner would recover the storage space by using the EraseMaster credential to change the encryption key - as in a normal crypto erase. This has the side effect of unlocking the band and defaulting the BandMasterX credential to the MSID value which in turn allows the owner to reinstate a new value for BandMasterX.

2 This represents a "null" value.

3 The host needs to prove ownership of the band by providing the correct value for the BandMasterX password.

3.1 Changing a BandMasterX password.

If the host suspects that a data band password may have been compromised, it can be changed at any time as shown in Figure 12.

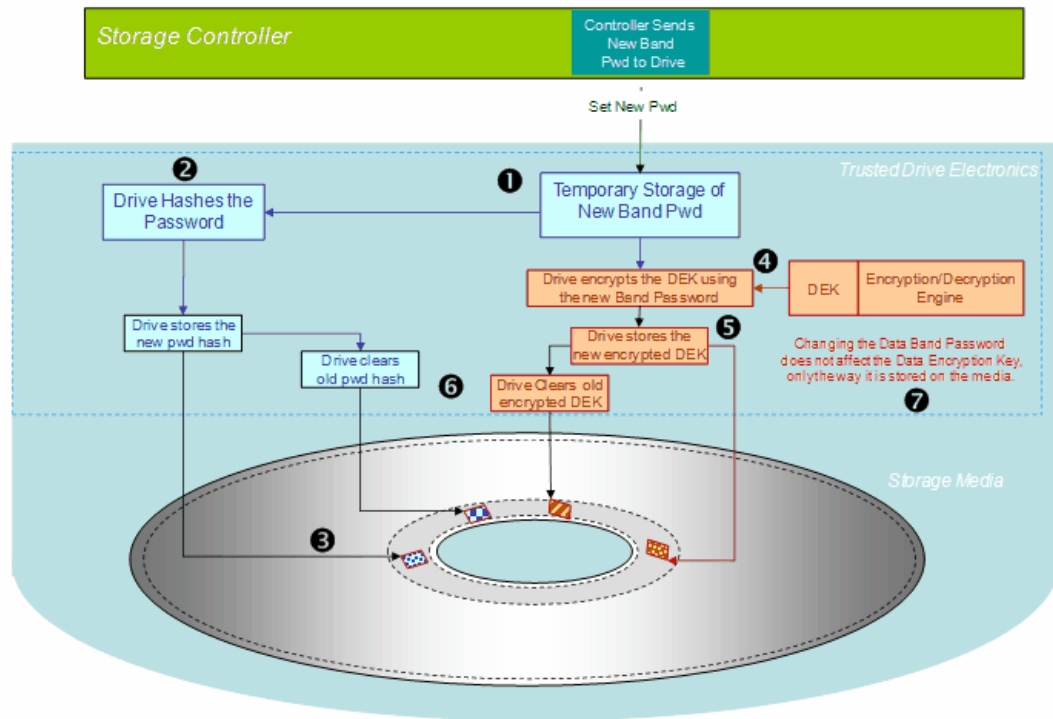


Figure 12. Changing the data band password

The initial condition is that the host has already authenticated to BandMasterX using the current password value then:

1. The host sends the new value for BandMasterX (new password) to the drive
2. The drive hashes the new password
3. The drive stores the new password digest on the media
4. The data encryption key (DEK) is retrieved from the encryption engine and encrypted with the new password
5. The drive stores the new encrypted DEK on the media
6. The old key values are erased from the media
7. The drive has a new BandMasterX password but the encryption key does not change

3.2 Summary

No keys are stored on the media in clear text format. All passwords are stored as hash digests and all data encryption keys are stored as encrypted objects using their respective data band passwords as the encryption key. If a data band is not conditioned to lock automatically on a power cycle, its DEK is encrypted using a secret key, known only to the drive, prior to storage on the media. The owner may change a data band password at any time without affecting the band's DEK.

4.0 User data bands

Unlike a standard drive, the user data area can be divided into separate data bands, each with its own password (BandMasterX) and set of locking parameters. The number of data bands available is product dependent but is never less than two.

When the drive leaves the factory all of its data bands are available but only band 0 is allocated. Band 0 is also known as the **global band**. The global band starts at LBA(0) and continues through LBA (max). In other words, at the time of shipment, the global band occupies the entire surface of the media minus the area allocated to the system band at the disk's inner radius. To allocate a data band to a physical address space on the disk, we need to know two things:

- The LBA that identifies the start of the band
- The number of LBAs contained in the band (the range)

If a band has not been allocated, the range is set to zero signifying that it does not provide any storage space. So let's go ahead and create a drive with the three data bands shown in Figure 13.

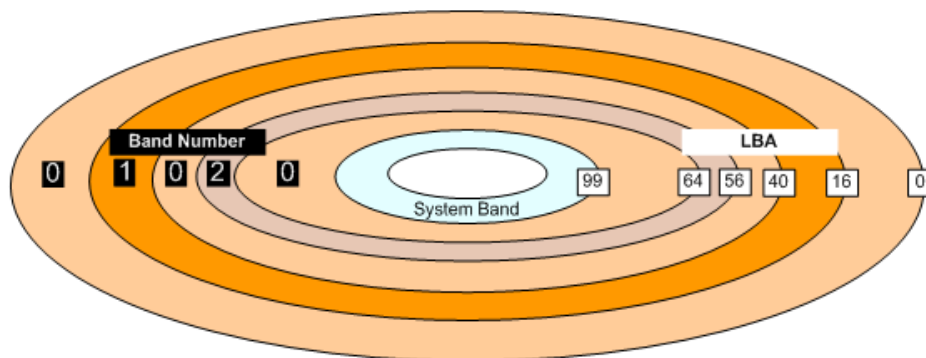


Figure 13. A disk with 100 LBAs and three data bands

For the sake of simplicity, let's say we have a total of 100 LBAs on the disk. The global band (band 0) will therefore start at LBA 0 and end at LBA 99.

To allocate band 1, we will set the start address to LBA 16 and the range to 24

To allocate band 2, we will set the start address to LBA 56 and the range to 8.

We do not have to (nor are we allowed to) manipulate the size of band 0. The drive will automatically give up storage space from band 0 to accommodate the new bands we allocate. If we wanted to de-allocate band 1, we would simply set its range to zero and the drive would return the space to the global band. Band 0 is the only band allowed to be discontinuous. In Figure 13 we see that band 0 occupies LBAs 0 - 15, 40 - 55, and 64 - 99. Allocated bands may be defined such that one ends at LBA (n) and the next starts at LBA (n+1), but they may not overlap. Bands must start at a 4 kilobyte sector boundary and since sectors (LBAs) are 512 bytes in length, the start LBA of a data band must be exactly divisible by 8.

A sample dialogue between the drive and the host to set up the data bands on a new drive is shown in Figure 16. Here BandMasterX is simply used as shorthand to represent any one of the data bands other than the global band.

For example, if the drive had 16 available data bands (including the global band 0), the 'X' would take any value from 1 through 15. To set up a data band, the first thing you have to do is to authenticate to the BandMaster authority using the current value of the password. If this is a new drive, we know that this defaults to MSID. With that done, the drive knows that you are a bona fide user and allows you to access the parameters associated with that band. This lets you set the band's start LBA (*RangeStart*) and its size (*RangeLength*) as well as the condition of its locking parameters *ReadLockEnabled*, *WriteLockEnabled*, and *LockOnReset*. If you were not intending to use the band and would prefer it to be locked at this time, you could set the *ReadLocked* and *WriteLocked* parameters to *TRUE*. Finally, you select a new password value for BandmasterX (replacing the value of MSID) to keep the band private to your use.

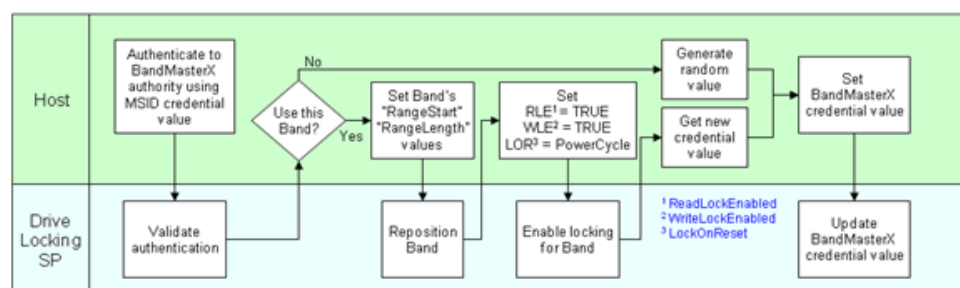


Figure 14. Setting up a data band

Notice early in the flow chart that if you decide not to use this band, it is still good practice to personalize the password. This prevents someone else from gaining access to the band (using MSID) and locking you out.

Now that we've covered the groundwork, let's summarize the rules and characteristics governing data bands.

4.1 Data band rules and guidelines

- At the time of shipment from Seagate:
 1. The global band (band 0) occupies all of the storage space on the media, LBA (0) through LBA (max). All other data bands have their range parameter set to zero. In other words, the bands exist but they are not allocated storage space
 2. All data bands have their own credential identified as BandMasterX where X is the band number. All credentials are set to the value of MSID
 3. All data bands have their own unique encryption key which is assigned during the manufacturing process
 4. All BandMaster Authorities, except the ones managing the Global Band and Band1 are disabled.
- The global band cannot be manipulated by the host—it is adjusted automatically by the drive when additional data bands are allocated.
- A new data band is allocated by assigning a start address and an LBA range. The real estate for the new band is taken from band 0.
- The Start LBA of a data band must be exactly divisible by 8—this preserves 4K sector alignment.
- If a data band is de-allocated by having its range parameter set to zero, its storage space is returned to band 0.
- Data bands may end and begin on consecutive LBAs but they cannot overlap. Band 0 is the only band which can be discontinuous.
- There is no limit to the size of a data band except that:
 1. It must be contained within the global band.
 2. It must not overlap any previously allocated band.
- Since each band has its own encryption key, it may be independently cryptographically erased.
- Changing the password on a band has no affect on the band's encryption key.
- Using EraseMaster to change the encryption key on a band has the following effects:
 1. The band is crypto erased
 2. The band is unlocked for reading and writing
 3. The band's password (BandmasterX) reverts to the value of MSID
- When a band is allocated, the storage space surrendered by the global band is effectively crypto erased. This is because the allocated band has its own pre-defined encryption key. If this new band is de-allocated prior to being written, the encryption key will revert to that used by the global band and, provided there has been no interim change to the global band's encryption key, the original data can be recovered.
- Under normal circumstances, Read and write operations are not allowed to cross band boundaries since doing so would involve an encryption key change and may involve an attempt to read data from a locked band. Of necessity, some initializing commands (e.g. a Media Format) are allowed to cross bands provided all bands are unlocked prior to the command being issued.
- If an owner decides against using any or all of the additional data bands, it is good practice to take ownership of the unused bands by personalizing their BandMaster passwords. Failure to do so could result in a Denial of Service attack.

5.0 Firmware download

A firmware download operation is not a normal occurrence in the lifecycle of a drive but is a useful way to update the drive code during the later stages of drive development when it is being integrated into new customer system designs. When the system hardware and the drive hardware come together for the first time, compatibility issues may come to light which are all but impossible to predict without an operational test.

On an SED, a firmware download operation is a serious event which could jeopardize the drive's security, so download operations require special attention to ensure that:

1. The code is genuine Seagate code with an appropriate digital signature (see Section 7.0 for a discussion on messaging and signatures).
2. The code has the correct security component and is SED family and model compatible
3. The new code is chronologically compatible with the old code.

Figure 15 shows the structure of the authorized firmware package. It is important to note that the host system will download the authenticated package to the drive in exactly the same manner as it would download a standard firmware file to a non SED drive. The firmware authentication and checking process is handled by the drive and is totally transparent to the system.

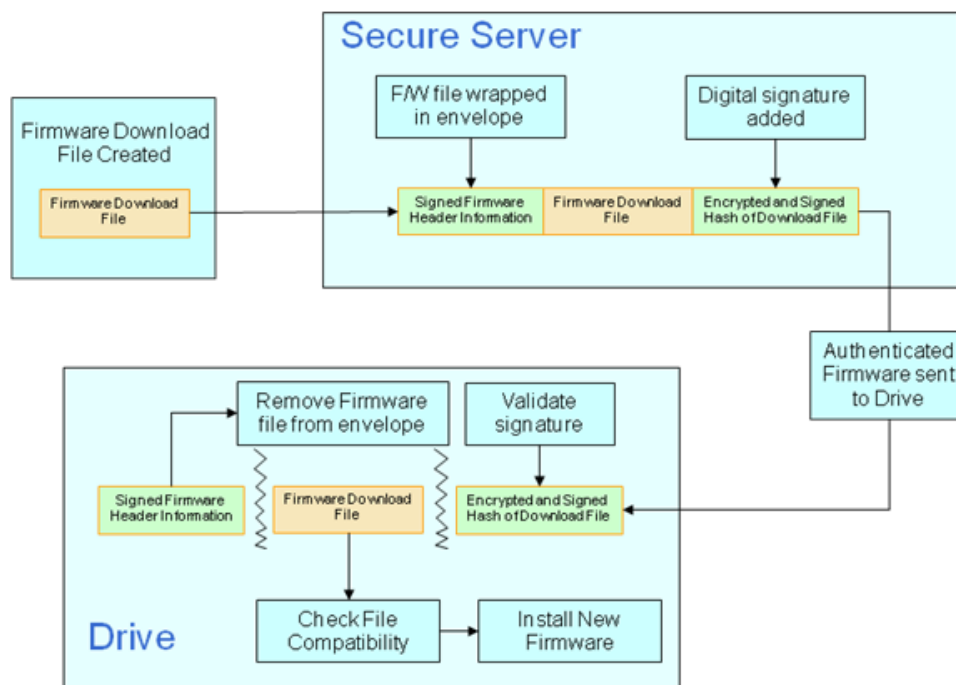


Figure 15. Authenticated firmware download

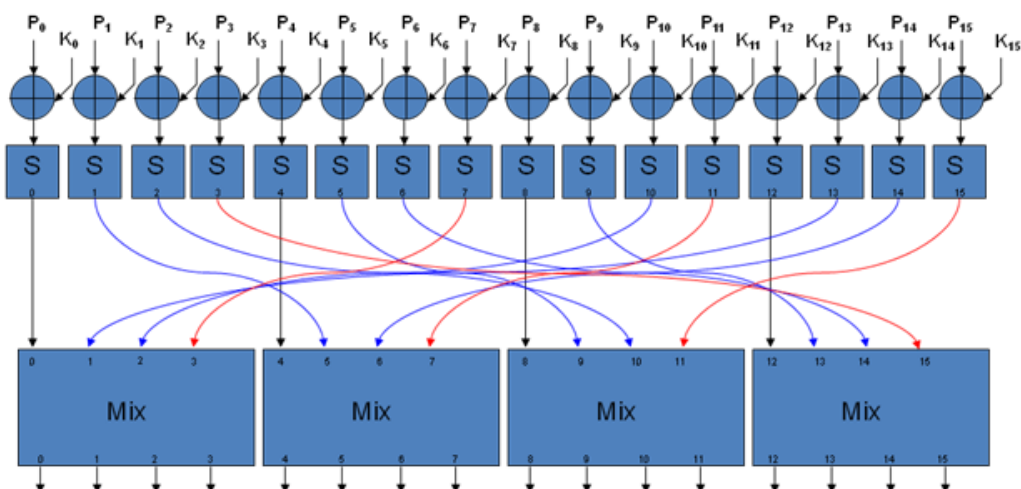
6.0 AES encryption algorithm

Figure 16 is a simplified pictorial representation of the AES128 algorithm. P_0 through P_{15} represent a block of 16 bytes (128 bits) of clear text message being input to the hardware. Each message byte is XOR'd with the corresponding byte of the encryption key (K_0 through K_{15}) and the results are fed to substitution boxes (which are just look-up tables) where each 8-bit input maps to an 8-bit output.

It may appear somewhat random, but the outputs of the substitution boxes are fed to a series of 4-byte Mixers in a particular cyclic pattern. Substitution output S_0 feeds Mixer input M_0 , S_1 feeds M_5 , S_2 feeds M_{10} and so on. For those purists interested in formulas, the connections can be represented as S_n feeds M_y where $y = (5n \text{ MOD } 16)$. The statement in parentheses is shorthand notation meaning "the remainder when $5n$ is divided by 16." For example, Substitution box 11 would feed Mixer input

$(5 \times 11 \text{ MOD } 16) = (\text{the remainder when } 55 \text{ is divided by } 16) = 7$, so S_{11} feeds M_7 .

The Mixers perform an XOR operation on the 4 input bytes and produce 4-byte outputs which are fed back to the inputs P_0 through P_{15} so that the whole operation can be repeated. After 10 or more repetitions, the mixer outputs will be stored as encrypted data and the whole operation starts over again with the next 128 bits of the clear text message. As can be seen in the discussion on **Cipher Block Chaining (CBC)**, the encryption key input (K_0 through K_{15}) is effectively modified every time a new 128-bit chunk of the clear text message is processed.



P_i = 8 bits of the 128-bit Plain Text input,
 K_i = 8 bits of the 128-bit encryption key.
 S = Substitution Box (Lookup Table) mapping 8 input bits to 8 output bits
Mix = 4-byte mixer (XOR function)

Figure 16. Block diagram of the AES algorithm

7.0 Asymmetric keys and digital signatures

We saw back in Section 1.2 how to create a *digest* by hashing the clear text message. If this digest is then appended to the message itself it can provide a convenient data integrity check after transmission. This process is reviewed in Figure 17.

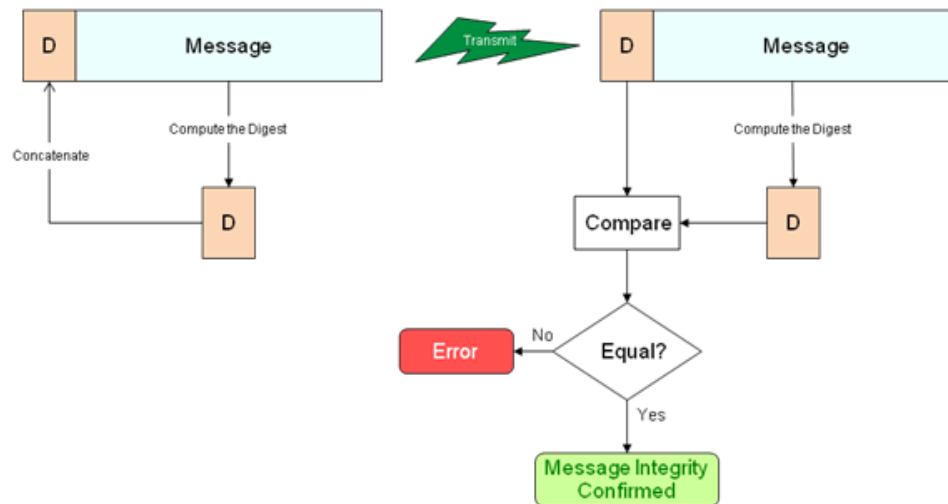


Figure 17. Using the hash digest to check message integrity

We can take this process one stage further, and use the digest to create a digital signature for the sender, but first we have to introduce the concept of asymmetrical encryption keys.

In our discussions on data encryption we have used the same key, a symmetrical key, to both encrypt and decrypt the plain text data, and we used the analogy that the same key is used to both lock and unlock the door to our house.

Now let's extend this analogy to a safety deposit box in a bank vault. To open the box, two different (asymmetrical) keys are required, one is held by the customer and one by the bank. This provides an added level of security by ensuring that boxes can only be opened in the presence of a bank employee who has verified the identity of the customer.

This concept would be very useful to protect messages from snoopers if the sender could encrypt the message with one key and relax in the knowledge that only the recipient, who has the second key of the matched pair, can decrypt it. In fact, this is how a secure messaging system works.

Let's consider Marge and Alan who want to send private messages to each other on their network. They each have a pair of keys identified as a private key and a public key. They keep their private keys to themselves and send their public keys to each other. Using these keys they can

- Sign their messages to prove authenticity, and
- Encrypt the message to ensure privacy.

Let's take a look at how Marge will sign her message to Alan.

In Figure 18, Marge has run her message through the hashing algorithm and obtained the digest. She now encrypts the digest with her private key and this becomes her digital signature which she appends to the original message and sends to Alan. Alan strips the signature from the message and decrypts it with Marge's public key to recover the message digest. He also hashes the message to create a second digest. A comparison of the two digests will prove the integrity of the message and if the message came from Marge.

This process verifies the sender of the message but it does nothing to protect the message from snoopers since it was sent as clear text.

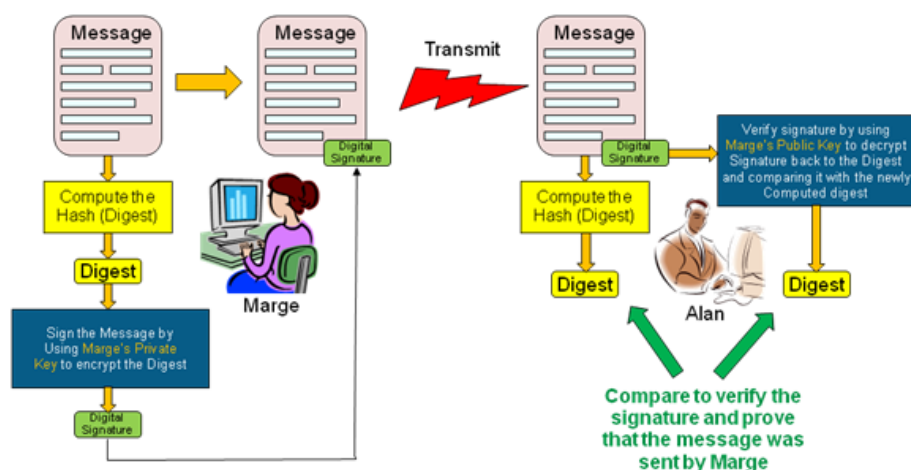


Figure 18. Using the hash digest as the sender's signature

To ensure privacy, we can take this process one stage further as shown in Figure 19. In this case, Marge signs the message as before, but before transmitting it to Alan, she encrypts both the message and the appended signature with Alan's public key.

When Alan receives the transmission, the first thing he does is decrypt the package with his private key. Once that is done he has the plain text message and the digital signature and he can continue to process the information as he did in Figure 18.

It has taken a fair amount of extra work, but at the end of it all, Marge can be sure that her message was secure and that Alan knows that it came from her and not some interceptor who is snooping on the network.

It is easy to get confused about what keys to use to encrypt messages and which to use for signatures, so here's a summary:

Marge's private key:	Used to sign messages to Alan and decrypt messages from Alan
Marge's public key ¹ :	Used by Alan to encrypt messages to Marge
Alan's private key:	Used to sign messages to Marge and decrypt messages from Marge
Alan's public key:	Used by Marge to encrypt messages to Alan.

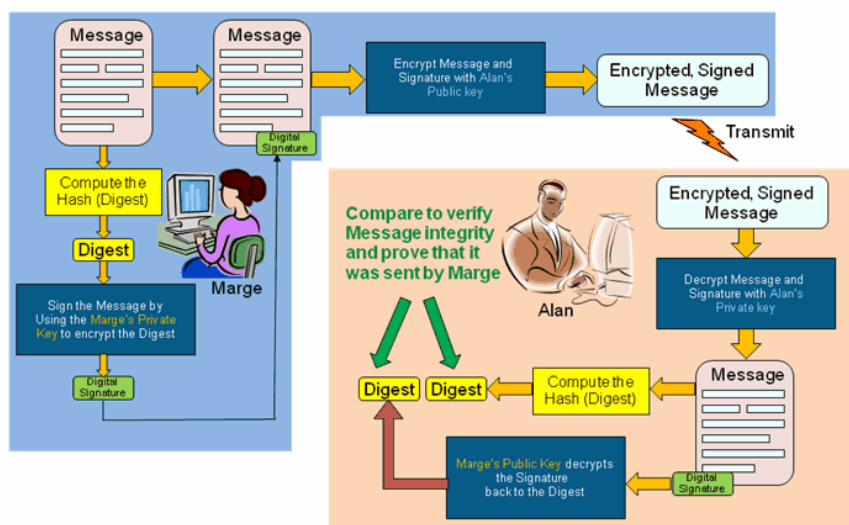


Figure 19. Signed messaging with privacy

¹ Marge would exchange public keys with anyone else on the network with whom she wants to exchange secure messages. A message encrypted with a public key can only be decrypted with the receiver's private key so there is no danger to privacy by distributing multiple copies of her public key.

Index

A

- Admin SP 8, 9, 12
- Advanced Encryption Standard 5, 9
- AES 5, 9
- AES128 5
- AES256 5
- algorithm 5
- authentication key 9
- Authentication keys 12
- authentication keys 13

B

- BandMaster 13, 15
- Block Ciphers 6

C

- CBC 6
- Cipher Block Chaining 6, 9
- cipher text 4, 8, 12
- ciphers 6
- clear text 8
- command restrictions 12
- Control commands 11
- CRC 7
- credentials 9, 10
- cryptographic erase 9
- Cyclic Redundancy Check 7

D

- data at rest 9
- data encryption 4
- Data Encryption Key 8
- data encryption key 9
- decrypted 9
- decryption 4
- DEK 8
- digest 7
- drive format 9
- Drive locking 8

E

- ECB 6
- Electronic Code Book 6
- encrypted text 4
- encryption 4, 9, 11, 18
- encryption engine 10
- encryption key 4, 5, 9, 11, 18, 19
- encryption keys 18
- erase 9

- EraseMaster 13
- erasure 15

F

- fingerprint 7
- firmware download port 13
- format 9

H

- hash 19
- Hash functions 7
- hashing algorithm 7
- host controller 11

I

- INCITS Technical Committee 11
- Initialization Vector 6
- IV 6

L

- Locking Security Partition 8
- Locking SP 8, 9, 12
- Logical ports 13

M

- Manufacturer's Secure ID 13
- MSID 13, 15

N

- National Security Agency 7
- non-encrypting 16
- NSA 7

O

- overview 10
- owner's credential 13
- owner's password 13

P

- password 15
- passwords 10, 12
- performance 9
- Physical SID 14
- protocol 3
- Protocol In 11
- Protocol Out 11
- PSID 14

R

- Read and write commands 11

- Read Buffer 12
- Read Long 12
- Read/Write Buffer 12
- Read/Write commands 18
- Read/Write Long 12
- repurposed 9

S

- scrapped 9
- SCSI commands 11
- Secure Hash Algorithm 7
- Secure ID 13
- security block 11
- Security commands 11
- security of data at rest 9
- security tables 10
- SED 8
- SEDs 4
- Self-Encrypting Drive 8
- Self-Encrypting Drives 4
- SHA 7
- SHA-1 9
- SHA256 7, 9
- SID 13
- standards 3
- symmetric encryption 4
- symmetric key 4
- system band 10

T

- T10 11
- TCG protocol 3
- Trusted Computing Group 3

U

- US government 5

W

- Write Buffer 12
- Write Long 12



Seagate Technology LLC

AMERICAS Seagate Technology LLC 10200 South De Anza Boulevard, Cupertino, California 95014, United States, 408-658-1000

ASIA/PACIFIC Seagate Singapore International Headquarters Pte. Ltd. 7000 Ang Mo Kio Avenue 5, Singapore 569877, 65-6485-3888

EUROPE, MIDDLE EAST AND AFRICA Seagate Technology SAS 16-18 rue du Dôme, 92100 Boulogne-Billancourt, France, 33 1-4186 10 00