CTWP016: Cactus Technologies® -240SE Series CryptoSSD Products

Covered Products: all -240SE series products

1 Introduction

SEDs (Self Encrypted Disk) are now widely available, most SSDs on the market today advertise as having hardware encryption built-in. However, not all SEDs are created the same. Data security is more than just having hardware encryption, good data security requires that the encryption engine must be properly designed with no bugs and/or loopholes to exploit. The encryption key must also be properly protected as it is the single point of weakness of the entire data security system. In this whitepaper, we will discuss how Cactus Technologies® CryptoSSD products address these issues.

2 Encryption Method

Encryption is a vast and complex topic, here we will only provide some basic information that is relevant to Cactus Technologies® CryptoSSD products.

The encryption algorithm used in Cactus Technologies® CryptoSSD products is AES256. AES is an encryption standard established by NIST (National Institute of Standards & Technology) in 2001. It is a subset of the Rijndael cipher and operates on blocks of 128bits with a choice of key sizes of 128, 192 or 256bits. AES was adopted by the US Government and is now widely used in many encryption software. In the US, AES is published by NIST as FIPS 197; it is also included in ISO/IEC 18033-3 standard. It is also the only public cipher approved by NSA for TOP SECRET information.

Numerous papers have been published that discussed various method of attacking AES encryption; as of today, there is no known method that can successfully crack a fully and properly implemented AES encryption in a reasonable amount of time.

There has also been concern about quantum computing potentially being able to crack AES encryption fairly quickly. However, that is not the case. Some encryption algorithms rely on factoring large numbers (e.g. RSA); there are known algorithms for attacking such type of encryption and quantum computing is well suited to running such algorithm. Thus, quantum computing is a threat to such encryption, which is commonly used, for example, in securing websites. AES encryption does not rely on factoring large numbers. In order to break AES256 encryption, one has to search through a large key space to find the key. There are some algorithms that speed up the search of this key space but even with the help of quantum computing, studies have shown that this would merely reduce the time to crack AES256 down to the equivalent of cracking AES128, which would still take an impractically long time to do. Therefore, at this time, AES256 is still widely considered to be resistant to quantum attack.

Due to the robustness of AES256 to resist various forms of attack, this is the generally recommended algorithm for securing stored data. Use of weaker strength keys, such as AES128, is not recommended.

2.1 AES Encryption Modes

When encrypting large chunks of data, AES can be operated in various modes. There are six modes approved by NIST and published in document SP800-38E (Recommendation for Block Cipher Modes of Operation). These six modes are:

- ECB (Electronic Code Book)
- CBC (Cipher Block Chaining)
- CFB (Cipher Feedback)
- OFB (Output Feedback)
- CTR (Counter)
- XTS (XEX mode w/ ciphertext stealing)

Each of these modes have strengths & weaknesses, we will not go into details here. Of these six modes, XTS mode is specifically suited for disk drive encryption. This mode originated as IEEE Standard 1619-2007 and was adopted by NIST in 2010. According to SP 800-38E, "In the absence of authentication or access control, XTS-AES provides more protection than the other approved confidentiality-only modes against unauthorized manipulation of the encrypted data." This is the preferred encryption mode for securing data in disk drives. While Cactus Technologies® CryptoSSD products can support any of the six modes listed above, XTS mode is the one that is currently being deployed.

3 Encryption Certification

When it comes to the security of the encryption engine, the questions that one has to ask are:

- how do I know the the encryption engine is designed correctly per AES specifications
- how do I know that there are no bugs/loopholes in the encryption engine that hackers can exploite

To address these issues, a certification process by independent 3rd party is necessary. In the U.S., NIST manages certification programs known as CAVP(Cryptographic Algorithm Validation Program) and CMVP (Cryptographic Module Validation Program). CAVP validates crytographic algorithms, such as AES, that are recommended by NIST and meets Federal Information Processing Standard (FIPS). For AES encryption, the relevant standard is FIPS-197. CMVP validates any cryptographic module that is designed to meet the requirements of FIPS 140-2 standard. FIPS 140-2 is a mandatory standard for the protection of sensitive data within U.S. Federal systems.

The validation is performed by certified, independent test labs. The CAMP validation process ensures that the encryption algorithm is properly designed and meets the AES specifications. The CMVP validation verifies that the hardware implementation faithfully reproduces what the algorithm requires and that there are no bugs/loopholes/backdoor, etc. that can be exploited. Once an algorithm or module has passed the validation, a certificate will be issued to the vendor and the algorithm or module will be added to a validated list maintained by NIST.

Most of the SEDs available on the market today are not FIPS-197 or FIPS-140-2 certified, thus, one cannot be certain that these products have implemented the AES algorithm correctly or that there are no loopholes or backdoors in their designs. In contrast, both the AES algorithm and the hardware implementation of the AES engine used in Cactus Technologies® CryptoSSD products have been validated by NIST. Cactus Technologies® uses X-WALL-MX+ encryption chip from eNOVA Technology Corp. in our CryptoSSD products. Their FIPS-140-2 certificate numbers are 3013, 3014 and FIPS-197 certificate number is 4013.

4 Encryption Key Exchange

In order to perform encryption/decryption, an encryption key is required. In the following discussion, we will refer to this as the DEK (Device Encryption Key). There are various ways in which the DEK can be provided to the device, we will discuss a few of these methods below.

4.1 Self Generated Permanent Key

In this method, the drive internally generates an encryption key and stores it in a reserved area on the drive. Data is automatically encrypted and decrypted, the user has no control of the DEK.

This method was used in some SSDs that are advertised as having AES encryption. However, this key handling method offers no protection to the data at all as the DEK is stored in the drive itself; anyone that has access to the drive will have access to the data. The encryption engine in this case is merely functioning as a data scrambler, which is a requirement for MLC/TLC NAND flash.

4.2 User Generated Session Key

In this method of key exchange, the user generates a DEK and sends it to the device, usually via an ATA Vendor Specific command. The DEK is not stored in the device, thus everytime the drive is power cycled, the user needs to resend the DEK to the device.

This method of key exchange is better than the Self Generated Permanent Key method just described because a 3rd party who has access to the drive cannot decrypt the data without providing the proper DEK. This is one of the key exchange method supported by Cactus Technologies® CryptoSSD products, we refer to this as the SetDEK method.

The weakness of this key exchange method is that the DEK is sent over to the device in plain text, thus it is possible for a 3rd party to capture this DEK through some sort of bus monitoring method. Furthermore, the DEK doe not change and the same DEK is send over to the device every time; thus, one a 3rd party has captured the DEK once, it can decrypt the data if it manages to get access to the drive.

4.3 Cactus Technologies® Single Factor Authentication w/ HMAC

To address the issue of key exchange security, Cactus Technologies® worked with eNOVA Technology Corp. to develop a secure key exchange method using HMAC protocol. HMAC stands for Key Hashed Message Authentication Code, this is a well known authentication protocol, commonly used in the industry.

This key exchange method uses FIPS 140 level 2 certified hardware crypto modules in eNOVA MX+ device to generate a DEK. A customer defined 256-bit shared key is used to generate a random Key Encryption Key (KEK), which is used to wrap the DEK in a MAC (Message Authentication Code). This MAC is then sent to the device. In the device, this MAC is authenticated in the MX+ device and the DEK is then unwrapped from the MAC.

The MAC is random and different every time and the DEK is not send over in plain text as in the SetDEK method; hence, this method of key changes is secure from 3rd party snooping. In order for a 3rd party to be able to decrypt the data on the drive, it has to have knowledge of the Unlock PIN, the shared key and the series of ATA commands for HMAC exchange; it is highly unlikely that a 3rd party will have access to all three items. Hence, this method of key exchange is highly secure.

As is the case with the SetDEK method, the DEK is stored internally in the MX+ device in volatile register only. This register is write only and cannot be read; the stored data is lost when power is removed, thus, key exchange must take place every time the device is power cycled.

This method of key exchange is very secure but is more difficult to integrate into a user's host system. There are numerous ATA Vendor Specific commands involved, the details of which are disclosed to the customer only after an NDA has been executed with eNOVA Technolgoy Corp. Cactus Technologies® will provide Windows and Linux HMAC utilities to manage the key exchange but for customers using other OSes or operating environments who need to write their own utilities, please contact Cactus Technologies® for further information and support.

5 Piggybacking on ATA Security Command

Some SEDs piggyback on the ATA Security Command to enable/disable encryption or as user access control to the encrypted drive. There are, however, some problems in using ATA Security Password this way:

1. There are two levels of ATA Security Password – User and Supervisor. A Supervisor Password will override the User Password; therefore, if the ATA Security Password is used only as authentication control, then a 3rd party who has Supervisor level password can access the encrypted drive.
2. ATA Specification allows for ATA Security Password to be disabled without affecting function of the drive. If a SED is using ATA Security Password as the DEK, then it is not

possible to disable the password without affecting device usage and the device would be violating ATA specifications in this case. If the ATA Security Password is used only as access control, then disabling it will allow 3rd party access to the encrypted data, which is a major security hole.

For these reasons, it is not advisable to piggyback on ATA Security feature as way to enable/disable encryption or to use it as user authentication for a SED.

6 CryptoErase

In applications that requires high level of data security, there is often a requirement that all the data on the drive be erased if needed to prevent a 3rd party from being able to access the data should they gain physical access to the drive. This erasure is often times carried out by performing one of several defined military erasure protocols, such as NSA 9-12, NSA 130-2, DoD 5220, US AF AFSSI 5020, US Navy NAVSO P-5239-26, etc. The problem with these erase procedures is that some of them take a long time to complete, particularly with large capacity drives. It can take tens of minutes to several hours to perform a military erasure on large capacity drives.

With the introduction of SEDs, however, the need to do such erasure routines becomes unnecessary. The data on SEDs is already scrambled during the encryption process, the encrypted data is virtually undecipherable without the encryption key. Therefore, to protect the encrypted data, all that is needed is to delete the encryption key and this process happens instantaneously; this process is commonly referred to as CryptoErase.

For SEDs that use Self Generated Permanent Key, the CryptoErase process is triggered by a ATA Vendor Specific command or via a jumper on the drive. Since Cactus Technologies® CryptoSSD products do not store the DEK in non-volatile memory on the drive, CryptoErase is as simple as simply cutting power to the drive. For added security, Cactus Technologies® CryptoSSD products that operates in HMAC mode can also delete the stored Unlock PIN via CryptoErase, this will effectively render the current DEK obsolete, thus, even if a 3rd party enters the Unlock PIN that was originally used, it will not be able to decrypt the drive data.

7 Summary

In this whitepaper, we provided a basic introduction to encryption technology as it relates to Cactus Technologies® CryptoSSD products. We also explored key exchange methods and introduced the user to Cactus Technologies® proprietary and secure HMAC key exchange method.

8 Version History

Version	***Date***	***Change***
1.0	April 30, 2018	Initial Version