

White Paper

Intel® EP80579
Integrated
Processor with
Intel® QuickAssist
Technology

Accelerating Secure Storage on FreeBSD*

Adrian Hoban, Brad Vrabete, Brendan Kennedy

*Performance Products Division
Embedded & Communications Group
Intel Corporation*

May 2009



Abstract

It goes without saying that Information Security is extremely important in today's connected world. Protecting the vast quantities of digital information stored by companies is critical to maintaining business integrity and reducing the risk related to the unintentional disclosure of private information. Storing data securely is one mechanism that can help reduce the risk of attackers gaining access to sensitive information. This paper examines some of the secure storage solutions that are available on the FreeBSD operating system and discusses options for the acceleration of processor-intensive cryptographic operations.

Contents

Abstract	2
Contents	2
Overview	3
Introduction	3
Acceleration	4
Results	5
Conclusion	7
References	7
Appendix	7

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit [Intel Performance Benchmark Limitations](#).

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting [Intel's Web Site](#).

GEOM Disclaimer

The following extract has been taken from the GEOM Based Disk Encryption man page [5]:

"...NOTICE: Please be aware that this code has not yet received much review and analysis by qualified cryptographers and therefore should be considered a slightly suspect experimental facility. We cannot at this point guarantee that the on-disk format will not change in response to reviews or bug-fixes, so potential users are advised to be prepared that dump(8)/restore(8) based migrations may be called for in the future..."

Intel and Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2009, Intel Corporation. All rights reserved.



Overview

This paper examines some of the secure storage solutions that are available on the FreeBSD operating system and discusses options for the acceleration of processor-intense cryptographic operations.

Two paradigms for securely storing data are introduced. The first describes the individual file description approach; the second describes the "transparent" encrypted partition model. Both have their advantages and it is left to the reader to decide which suits their usage model best.

Mechanisms for accelerating the cryptographic processing are also discussed. This section focuses on the use of the OpenSSL toolkit and on leveraging the OpenBSD Cryptographic Framework (OCF) for gaining easy access to hardware accelerators.

A test environment was set up to compare the performance of the different secure storage options using software- and hardware-based cryptography. The hardware platform used for testing was based on an Intel[®] EP80579 Integrated Processor with Intel[®] QuickAssist Technology. This processor has an integrated high-performance security acceleration engine for algorithms for bulk encryption, hashing, authentication, IKE/PKE, and true random number generation.

The test results show that in this particular test environment, the workload was more IO-bound than CPU-bound. The GDBE class in the GEOM framework produced by far the slowest times to encrypt the test file. The GEOM ELI (GELI) class using a software crypto implementation was quicker than an OpenSSL pure software implementation and resulted in 33% reduction in absolute CPU load. When the test results based on the Intel[®] QuickAssist Integrated Accelerator were examined, it was clear that with these IO-bound workloads the greatest benefit was in the reduction of CPU load. The GELI HW accelerated load was 14% less in absolute terms than the load experienced with pure software configuration (equivalent to a 65% percent reduction). Similarly, the OpenSSL HW acceleration showed a drop of 15% CPU load.

Introduction

At its core, information security may be described as having three key properties concerning the confidentiality, integrity and

availability of data [1]. Data confidentiality relates to maintaining privacy by preventing the information being accessed without authorization. Data integrity relates to the property of preventing data from being modified without authorization. Data availability relates to the information being available when it is required. This paper focuses on the software-related aspects of data confidentiality in the context of systems running the FreeBSD operating system.

Fundamentals

A cryptographic algorithm and cryptographic key is used to convert (encrypt) the plaintext data to a form that is unusable, known as ciphertext. A cryptographic key is required to convert (decrypt) the ciphertext back into plaintext.

In secure storage solutions, the algorithm used as the cipher is typically chosen from the symmetric key suite. The Advanced Encryption Standard (AES) and the triple variant of the Data Encryption Standard (3DES) are two examples of this type of cipher, although many more exist. Algorithms in this suite share the property that the cryptographic key used for encryption and decryption are either the same or simply related. The size of the key, the contents of the key, and the algorithm itself all have an effect on the ability of the cipher to resist attacks [2].

Secure Storage Options

There are many mechanisms that can be employed by a FreeBSD developer who wishes to add confidentiality to data stored on hard drives or removable storage devices while also hiding many of the complexities associated with cryptography. The first choice that needs to be made is between per-file based "individual" file encryption and partition-based "transparent" disk encryption.

Individual File Encryption

With individual file-based encryption, the user already has a copy of the file stored in plaintext. They can use a cryptographic library, such as the one included with the OpenSSL toolkit, to create a new ciphertext version of the file [3]. If acceleration is available for the cryptographic algorithm, then OpenSSL may access this acceleration via the native crypto framework [4].

This model works well when the user does not need to encrypt very many files and/or when the user does not desire security on the local non-



removable storage devices. A common usage model is when a user wants to transfer a plaintext file from their local hard drive onto a portable USB memory stick as a ciphertext file. If the memory stick is lost or stolen, the task facing an attacker to retrieve the data from the memory stick is made considerably more difficult by having the data stored in ciphertext form.

"Transparent" Encrypted Partition

With the transparent encrypted partition model, the user sets up an encrypted partition on their storage device. Once the encrypted partition has been set up, a write of a file to the encrypted partition will result in the file being encrypted automatically. On reads from the partition, the file will be decrypted automatically. There is much flexibility in how the partition may be set up, such as the ability to configure if the partition is applied to an entire storage device, part of a storage device, or on a particular file contained within another partition.

FreeBSD has a modular storage framework called GEOM. Encryption-related modules can be loaded on top of GEOM to provide transparent encrypted partitions. Two of these modules are GEOM Based Disk Encryption (GBDE) [6] and GEOM ELI (GELI) disk encryption [6]. Both GBDE and GELI provide the ability to encrypt a partition; however, GELI offers the advantage of utilizing the crypto framework which facilitates hardware acceleration. Another benefit of using GELI is that random one-time use keys may be used to encrypt a partition; this is particularly useful when swap space needs to be protected.

The transparent encrypted partition model works well when the user has many files that they want to store securely and also when they require that the data is stored in ciphertext form.

Acceleration

How an application can be accelerated

The common way of accelerating an application is to use a hardware device (hardware accelerator). Since the hardware accelerator is dedicated to executing only a certain task/function, typically it can be optimized to execute that function faster than a general purpose CPU.

For cryptographic functionality there are many hardware accelerator devices available. Since each one of them has a different API, it can be

hard for an application to support more than one such device. One solution to that problem was devised in OpenBSD: OpenBSD Cryptographic Framework (OCF).

OCF is an *asynchronous* service virtualization layer inside the kernel, which provides uniform access to hardware cryptographic accelerator cards. OCF implements two APIs for use by other kernel subsystems: one for use by *consumers* (other kernel subsystems) and another for use by *producers* (crypto-card device drivers). OCF supports two classes of algorithms: symmetric (e.g., DES, AES, MD5) and asymmetric (e.g., RSA). OCF works as a translation unit between the application request and hardware accelerator functionality.

In order to integrate with OCF, a device must implement a set of call-back functions which have to translate the OCF request into calls to the hardware accelerator's internal API. These will be run inside the kernel. The external API, used by consumers, enables an application to transparently use any hardware accelerator behind OCF.

OCF enables applications to use hardware accelerators without the need to have separate implementations for each device. OCF also makes it easier for hardware accelerator device manufacturers to have the device adopted by implementing an OCF driver for that device.

OpenSSL Acceleration

OpenSSL is a user space Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. [3]

OpenSSL on FreeBSD has user space OCF support and therefore can be accelerated. In order to accelerate OCF, the `-engine` command line option must be used.

The OpenSSL library is used by a large number of OpenSource applications such as OpenVPN, Apache, and Nautilus, among others. All these applications will transparently use accelerated OpenSSL if an OCF-compatible device is present in the system.

To encrypt a file using OpenSSL, the following command line can be used:

```
openssl <cypher> -a -salt -in <in_file_name>
-out <encrypted_file_name>
-engine cryptodev
```



The `-engine` command parameter tells OpenSSL to use a hardware acceleration engine and `cryptodev` is the name of the OCF engine.

Other Library Acceleration

Another approach is to call the OCF framework directly from user space. This eliminates the need of going through the OpenSSL library and has the potential for better performance for user space applications.

How to do it transparently

Instead of having to encrypt/decrypt each individual file, it may be more convenient to encrypt an entire partition and transparently save and retrieve files from there.

FreeBSD provides two partition encryption classes:

- The GEOM Based Disk Encryption (GDBE) class, and
- The GEOM ELI class [6].

Both classes transparently encrypt entire file systems and both classes are within the GEOM framework.

Both of these solutions work in a similar way. They present a file system that is transparently encrypted to the user. There is zero learning curve for the end user after the partition and file system have been initialized. The differences are in implementation and features: GELI uses OCF, can encrypt the root partition, and supports multiple cryptographic algorithms. Because of using OCF, GELI can be automatically accelerated when a cryptographic hardware is present in the system.

Other Tools

The approaches presented in this paper are not the only ones available. There are a large number of other tools, both open source and proprietary, that can be used to do either file encryption or entire partition encryption. An example of two of these tools are:

- Ncrypt (which does not require any external library) [11].
- GnuPG (which has GUI front end for file encryption) [12].

Results

Test Case Description

Test cases were set up to compare the equivalent file encryption and decryption methods using OpenSSL, GDBE and GELI. The encryption algorithm chosen was AES with 128-bit keys and Cipher-Block-Chaining mode. This cipher and mode represented a common denominator in terms of functionality between the different encryption methods.

For the transparent disk encryption tests, GDBE and GELI were set up to work with a hard drive partition configured on the local hard device. The test copied a large file to the partition (for encryption) and then copied the same file from the partition to an unencrypted partition (for decryption).

For the individual file tests, OpenSSL was used to encrypt and decrypt the large file directly.

The tests were run on two separate test environments: one used a SATA disk drive, the other used a Solid State Drive.

Test Environment

An Intel® EP80579 Customer Reference Board with the following specifications was used:

- 1GB RAM
- 160GB SATA Hard Disk Drive
- 80GB SSD Hard Drive
- Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology (1.2 GHz)
- FreeBSD 7.1 operating system

A 477 MB file was encrypted and decrypted on the SATA Drive using the different file encryption methods. Process time was monitored using the FreeBSD 'time' command. CPU load was an average gained through using the 'top' command, based on the '%CPU idle' reading.

Initial Setup

A 4 Gigabyte partition was created at OS install time to be used as the encrypted partition on the hard drive. It was formatted as the default FreeBSD partition type (UFS).

A 477 MB file was created. (See [Appendix](#)).



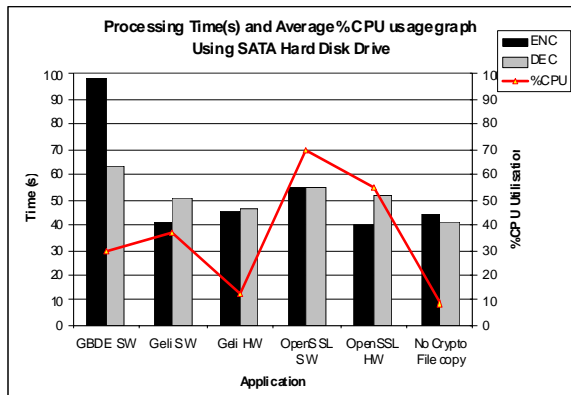
GBDE and GELI set ups may be found at [6]. Both should not be tested at the same time.

An OpenSSL guide may be found at [8]. FreeBSD 7.1 comes with OpenSSL 0.9.8e as standard.

Test Results

The results of the tests that were run using the SATA Hard Disk Drive have been captured in Figure 1. As a baseline, the time and CPU load required to perform a copy of the file without any encryption is also presented in the graph.

Figure 1. Cryptographic Performance Comparison between GEOM, GELI and OpenSSL using a SATA Hard Disk Drive



GBDE seems to be the least well optimized solution, as it required a high level of CPU load (30%) and also needed almost 100 seconds to complete the file encryption.

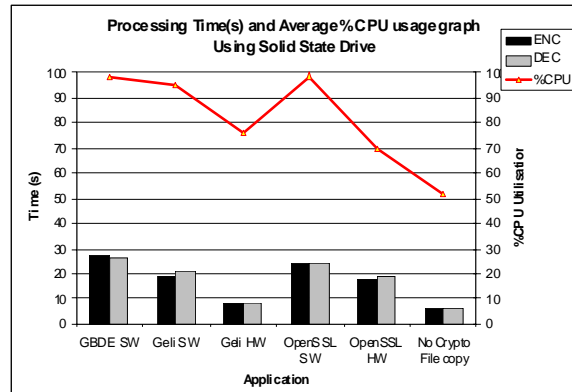
GELI Software took a similar amount of time to the basic file copy operation, but required four times the % CPU usage.

GELI Hardware results showed the positive effect of using Intel® QuickAssist Technology. Here the CPU load was only 4 percentage points higher than with the plain file copy scenario.

OpenSSL SW and HW operations on average took a similar amount of time and utilized far more CPU than the other methods. This is possibly due to a large number of copy operations between User and Kernel space. (OpenSSL is a user space program that links into kernel space hardware encryption support through its engine interface.)

The tests were repeated using the Solid State Drive (SSD) instead of the SATA Hard Disk Drive and the results are captured in Figure 2.

Figure 2. Cryptographic Performance Comparison between GEOM, GELI and OpenSSL using a Solid State Drive



The most striking differences between the two sets of results were that the overall time required to perform each of the tests has been dramatically reduced. With this reduction in time comes an expected increase in CPU loading. The CPU loading was consistently low during the first set of tests (Figure 1) as the CPU was forced to stall waiting on disk IO. The addition of the SSD to the system for the second set of tests significantly reduced the disk IO bottleneck, thereby leading to a reduction in the number of CPU stalls and an effective increase in CPU utilization. From an energy-efficiency perspective, this is a desired result; that is, the CPU should do the required task as quickly as possible (trend towards 100% utilization) and then be permitted to switch into an efficient low power state during idle times.

A comparison of the software-based encryption to the hardware-accelerated encryption test results in Figure 1 and Figure 2 shows clearly that the addition of hardware acceleration under these test conditions leads to reduction in both the time taken to encrypt the data and in the CPU loading required in order to complete the task.



Conclusion

The first and most obvious conclusion that can be drawn is that the system exhibits disk IO-bound characteristics when the tests are run with the SATA Hard Disk Drive. This observation is based on the relatively low CPU loading, the relatively long time required to complete the tests, and the comparison to the SSD-based results.

When considering the disk IO-bound environment, apart from GBDE, the main differentiator here between the different file encryption options was shown to be CPU Utilization. Using Intel® QuickAssist Technology, the CPU utilization was roughly halved in some scenarios and despite the bottleneck in these test cases being related to disk IO load and not CPU headroom, even when using OpenSSL, the CPU utilization was still lower by 15%.

When considering the SSD-based test results, the improved IO performance capability resulted in overall quicker task completion and higher, more efficient CPU utilization. The addition of hardware acceleration resulted in quicker task completion and reduced CPU loading (e.g. 29% reduction in the OpenSSL SW vs. HW results). When comparing GELI software encryption test results on the SATA Hard Disk Drive to the GELI hardware accelerated encryption test results on SSD, it is observed that by switching to SSD storage technology and utilizing hardware acceleration, 33 seconds of CPU processing power at 37% CPU load utilization was saved.

Another interesting observation on the SSD-based result is that the time required to complete the GELI hardware accelerated tests was close to the non-encrypted result (8 seconds vs. 6 seconds).

GELI accelerated with Intel® QuickAssist Technology is a compelling disk encryption solution for FreeBSD. It is easy to use and increases performance, while also leaving plenty of headroom for other tasks. The processing power-saving that is possible by leveraging Intel® QuickAssist Technology presents system designers with the opportunity to implement enriched feature sets that can help differentiate their product in the marketplace.

References

- [1] Allen, Julia H. (2001). The CERT Guide to System and Network Security Practices. Boston, MA: Addison-Wesley. ISBN 0-201-73723-X.
- [2] Mel, H.X, Baker, Doris M. (2001). Cryptography Decrypted. Addison-Wesley. ISBN 0-201-61647-5
- [3] OpenSSL Homepage: <http://www.openssl.org>
- [4] FreeBSD crypto interface man page: <http://www.freebsd.org/cgi/man.cgi?query=crypto&sektion=9&manpath=FreeBSD+7.1-RELEASE>
- [5] GEOM Based Disk Encryption man page: <http://www.freebsd.org/cgi/man.cgi?query=gbde&manpath=FreeBSD+7.1-RELEASE>
- [6] FreeBSD handbook chapter on Disk based encryption: <http://www.freebsd.org/doc/en/books/handbook/disks-encrypting.html>
- [7] The Design of the OpenBSD Cryptographic Framework: <http://www.openbsd.org/papers/ocf.pdf>
- [8] OpenSSL Guide: <http://www.madboa.com/geek/openssl/>
- [9] GEOM/GBDE related information: <http://phk.freebsd.dk/pubs/>
- [10] GEOM programming: <http://www.freebsd.org/doc/en/articles/geom-class/index.html>
- [11] NCrypt - NMRC File Encryptor / Decryptor / Wiper: <http://ncrypt.sourceforge.net/>
- [12] The GNU Privacy Guard: <http://www.gnupg.org/>

Appendix

Command to generate a 2Gbyte File:

```
TEST_MACHINE# dd if=/dev/random of=test.bin  
bs=500000000 count=1
```

