VIA PADLOCK : SECURITY AND USABILITY EVALUATION

Gordon Murphy
Steven Vasilakos

CSCI-E 170, Fall 2004
Final Project

ABSTRACT

VIA Technologies, Inc., a popular microchip supplier for the personal computer and related markets has recently announced a PC security initiative which integrates powerful and low-cost hardware-based security mechanisms into an x86 PC processor. Formally named VIA Padlock, this initiative also includes a software suite that utilizes the integrated hardware-based security building blocks to offer a broad set of security features. We present a thorough review, analysis, and test drive of the VIA Padlock initiative which addresses the benefits of adopting this initiative, and benchmarks certain features that VIA claims performance improvement over software-only counterparts. This also entailed assembling a PC with a VIA motherboard that is VIA Padlock fortified. Our goal is to provide the audience with an interest in this technology a practical "hands-on" review of its potential and details. At the very least, an educated decision can then be formed as to whether and how to advance one's own further investigation in the direction required.

---

1. INTRODUCTION

With the motivation of mitigating a broad range of security risks, the VIA Padlock Initiative combines a flexible array of low-cost hardware building blocks and software utilities to offer a variable policy based security umbrella. Architecturally, Padlock consists of the VIA Padlock Hardware Suite and VIA Padlock Software Suite. General benefits include a design that embraces open standards, like utilizing the Advanced Encryption Standard (AES) for encryption, diversifies its implementation across hardware and software, and provides a seamless layer of security protection by default.

VIA Padlock Hardware Suite features include the VIA Padlock Quantum Random Number Generator (RNG) and Advanced Cryptography Engine (ACE). True random numbers play a significant role in encryption and hashing, while the ACE supports industry standard algorithms.

VIA Padlock Software Suite features include secure Virtual Private Network (VPN) support, VIA Padlock Tru-Delete to wipe away sensitive files, a ZIP utility which offers speedy compression with encryption and a Software Development Kit (SDK) that presents third party developers with the ability to develop security features utilizing the RNG and ACE hardware building blocks.

In the course of this paper we will evaluate the VIA Padlock RNG, ACE, related SDK, and the Tru-Delete and Zip software utilities offered by VIA to showcase various aspects of the RNG and ACE building blocks. Each evaluation follows the same construct; its assumed purpose, a cognitive walk-through, applicable metrics tables

including a checklist of Saltzer and Schroeder Deign Principles, and finally, a general assessment. All of the testing was accomplished on the VIA EPIA MII 12000 Mini-ITX mainboard which features the 1.2GHz VIA C3 processor with the C5P Nehemiah core. Current prices are hovering just below 200 USD, with a complete system in the 500 USD range for the do-it-yourselfer. This relatively inexpensive and miniature board is well suited for many markets, and includes, along with the RNG and ACE, integrated network, graphics, and sound facilities.

It is also, we feel, important that the reader be aware that this evaluation was conducted without any direct contact with VIA Technologies, was approached from an as unbiased perspective as possible, and was limited to and completed in the course of a few weeks. By making use of free, trial, and low cost tools and components, as well as, making pertinent source code listings available to the reader, we have made every effort to allow for the intrigued reader to easily reproduce our results before making any final definitive conclusions for themselves (and we strongly recommend that applicable readers make this effort.) In a sense, this work was made by developers, for developers, however, its true intended audience is open to anyone wishing to learn more about the topic.

---

2. RNG EVALUATION

VIA Padlock Random Number Generator (RNG) – C5P Nehemiah core,

VIA Padlock SDK Version 1.0 (available for Linux and Microsoft Windows)

2.1 PURPOSE

The VIA RNG is an answer to the need for powerful low cost security mechanisms in general computing. Used as a building block, it can be applied to improve encryption, hashing, low level erasure of files, and is made conveniently available to developers through a thin SDK for any other desired purpose. To meet the performance and high entropy demand the RNG is designed into the hardware's processor where it is theoretically able to make use of quantum artifacts found in electrical noise.

2.2 WALK-THROUGH

Utilizing the leverage of the Padlock RNG in a direct manner requires only that one know how to execute its **xstore** processor instruction. Although using assembler is an option, we chose the route developers are more likely to approach and installed the available C based SDK (Windows XP platform). In relation to the RNG, the SDK is actually very light and the source code, which is available, exemplifies VIA's tendency toward simplicity and efficiency with the in-lined assembler calls appearing terse and option flexible. Specifically, there is one function to determine if the RNG is available, and another to request a buffer be filled with random bits. The buffer length is specified by the developer as a function parameter.

Before beginning, we chose to spend some time learning more about the intrinsic design of the RNG. A number of documents available at the manufacturer's website quickly revealed an entire universe related to random number theory and how the current design is believed to be based on sound principle in this regard. The current evolution includes two (2) entropy sources. The sources are claimed to be driven by quantum artifacts, specifically, electrical noise in the chip. It is believed, subatomic activity is by definition truly random, however, the process of measuring this noise may result in minimal bias and possible undesired bit to bit correlations. This and other reasons, such as variable performance and application demands, explain the inclusion of various options to the RNG instructions known as whiteners, filters, and DC bias settings which can improve perceived entropy, but, possibly reduce the throughput of random numbers the system can generate.

Pursuing the primary question of whether Padlock can efficiently and effectively generate strongly random numbers of its stated cryptographic caliber, we turned to the Diehard Random Number Test Suite (http://stat.fsu.edu/pub/diehard). This sensitive eighteen (18) test battery uses algorithmic scrutiny to glean non-random traces from a large sample of random numbers (we chose sample files of 3 million 32-bit integers). After writing a random number generation routine which accessed Padlock RNG as a source, we applied the data into Diehard for analysis. To make the evaluation interesting we also generated a similarly sized sample using a software source, the **Random** class in the Sun Java 1.5 JDK, to produce the random numbers. Technically, Java's facility is a Pseudo-Random Number Generator (PRNG) because it simulates entropy based on a variable seed value. We present the results in *Table 2*.

After 5 random sample generations each, it would appear that Padlock fared well, with fewer test failures than the Java samples (1 test failure with RNG compared to 15 with Java's Random class). Diehard produces probability, or "p", values, as well as, Kolmogorov-Smirnoff, or "K-S" values. Each test may use one or more of them differently, however, when they approach one or zero to six decimal places or more it is interpreted as a clear test failure. This means that some non-random trace was detected in this data sample, not necessarily that the RNG is poor. Given repeated failure, only then a case for judging against a particular RNG can begin to form.

Really good random generators are difficult to find because they are difficult to prove. Padlock's RNG is based on theoretically sound design principles with its physical entropy sources. It does what other generators aim to do, but more easily, and with higher availability and flexibility. A competitive PRNG, perhaps also employing SHA-1 hash mixing, must rely and depend on more variables and layers to maintain integrity, especially in assailable environments. This is not to say one could not also strengthen the Padlock RNG with algorithmic post-processing, however, in general is leaves less to disrupt its integrity, and is not theoretically susceptible to seed reproducibility attacks.

2.3 METRICS

Table 1: Saltzer and Schroeder Design Principles Checklist

Metric	Observation
Economy of mechanism	Small, efficient, self-contained, and well integrated into the core.
Fail-safe defaults	Default RNG options via the SDK seemed to result in respectably random data samples.
Complete mediation	This is a low level mechanism.
Open design	Open access for user applications, as well as, operating system processes.
Separation of privilege	Two entropy sources.
Least privilege	Self-contained.
Least common mechanism	Instructions to the RNG are claimed to be atomic and safe for multi-threaded environments.
Psychological acceptability	Straightforward SDK, and familiar/practical sample utilities to showcase the functionality. Also, Linux and OpenSSL reportedly already provide some level of RNG support, or plan to.

Table 2: Diehard Random Quality Test (*across 5 runs, with each of the 5 RNG and 5 Java samples randomly generated*)

Metric	RNG failures	Java failures	RNG min *ks*	Java min *ks*	RNG max *ks*	Java max *ks*	RNG min *p*	Java min *p*	RNG max *p*	Java max *p*
B-day	0	0	0.019667	0.013391	0.966485	0.323245	0.070415	0.016499	0.99433	0.980277
Overlap	0	0	N/A	N/A	N/A	N/A	0.0002	0.085686	0.584119	0.984023
Rank 31	0	0	N/A	N/A	N/A	N/A	0.382667	0.507448	0.948951	0.790079
Rank 32	0	0	N/A	N/A	N/A	N/A	0.32176	0.434447	0.96605	0.984763
Rank 6x8	0	0	0.075864	0.064926	0.996612	0.935926	0.000049	0.015182	0.999744	0.973959
Bit-stream	1	0	N/A	N/A	N/A	N/A	0.00221	0.00693	1	0.973
OPSO	0	5	N/A	N/A	N/A	N/A	0.0003	0	0.9978	0.9783
OQSO	0	5	N/A	N/A	N/A	N/A	0.0145	0	0.999	1
DNA	0	5	N/A	N/A	N/A	N/A	0.021	0	0.9945	1
Count 1 stream	0	0	N/A	N/A	N/A	N/A	0.021313	0.068981	0.999524	0.966642
Count 1 specific	0	0	N/A	N/A	N/A	N/A	0.009653	0.016767	0.998099	0.998314
Parking Lot	0	0	0.099592	0.435448	0.539567	0.9724	0.037471	0.022262	0.969407	0.944998
Min Dist	0	0	0.376844	0.006228	0.799079	0.958259	0.021483	0.003399	0.995986	0.995568
3D Spheres	0	0	0.373741	0.010414	0.852277	0.9515	0.00086	0.00833	0.99915	0.97936
Sqeeze	0	0	N/A	N/A	N/A	N/A	0.042509	0.059436	0.906727	0.892706
Overlap Sum	0	0	0.117689	0.083973	0.717538	0.905624	0.007729	0.013571	0.989307	0.947527
Runs	0	0	0.045889	0.000748	0.968779	0.908064	N/A	N/A	N/A	N/A
Craps	0	0	N/A	N/A	N/A	N/A	0.035787	0.130933	0.895806	0.999714

2.4 RNG ASSESSMENT

These results are promising for Padlock, however, would need to be followed up with many more, possibly thousands, of tests against various test suites for a highly definitive confirmation. VIA claims that this is in fact the case based on long term independent studies which even questioned the effects of atmospheric conditions on the RNG. Given more time we would look to manage automated test results in a centralized database and also attempt to find weaknesses using the various non-default RNG options, and possibly test against the National Institute of Standards and Technology (NIST) Statistical Test Suite (STS). For the time being, we can attest to the usability of this offering having found it easy to access and not having witnessed any negative Diehard test statistics to date.

---

3. ACE EVALUATION

VIA Padlock Advanced Cryptography Engine (ACE) – C5P Nehemiah core,

VIA Padlock SDK Version 1.0 (available for Linux and Microsoft Windows)

3.1 PURPOSE

The VIA ACE is a high performance hardware implementation of the Advanced Encryption Standard (AES) algorithm for encryption and decryption. Integrated into the processor core, it provides an accessible low level cryptography path open to the system and developers. A flexible thin layered SDK is provided to enhance usability.

3.2 WALK-THROUGH

Utilizing the leverage of the Padlock ACE in a direct manner requires only that one know how to execute its **xcrypt** processor instruction. Although using assembler is an option, we chose the route developers are more likely to approach and installed the available C based SDK (Windows XP platform). In relation to the ACE, the SDK is actually very light and the source code, which is available, exemplifies VIA's tendency toward simplicity and efficiency with the in-lined assembler calls appearing terse and option flexible.

ACE is a complete AES implementation which supports encryption and decryption for the four cipher modes Electronic Code Book (ECB), Cipher Block Chaining (CBC), 128-bit Cipher Feed Back (CFB), and 128-bit Output Feed Back (OFB). All three AES key sizes, 128, 192, and 256 bit, are supported, as well as, two (2) extended key generation options and includes the ability to customize the extended key sequence in the event that a non-standard algorithm is desired. Another flexible feature is the ability to analyze intermediate round results, making ACE more than simply a high-performance AES black-box. Pipelining is also employed to enhance performance and efficiency.

The ACE functions provided in the SDK are easy to use and provide for three separate usability flavors; *plain*, *aligned*, and *fast aligned*. *Plain* means that memory alignment is not guaranteed by the developer in the data pointers provided, *aligned* guarantees ACE that the data is memory aligned, and *fast aligned* is a simplified aligned function set. The reason for the distinction has to do with performance where pre-aligned data is faster in general for processors to access. ACE specifies a sixteen (16) byte block alignment, and data sizes must be divisible by 16 possibly requiring padding on files that are not.

We ran a test against the Sun Java 1.5 JDK's AES implementation and realized a performance gain factor of approximately 10 or higher in using ACE for encryption and decryption. The testing employed a 128-bit key, which is common, the CBC cipher mode, and it targeted the 45 megabyte Java 1.5 JDK setup executable file.

Note that the file was initially padded from 46,065,096 to 46,067,712 bytes using the MS-DOS "echo 00000 >>" command so that it would comply with ACE's 16 byte divisibility requirement. We chose to employ the *fast align* SDK function set because typically it is a simple matter to pre-align memory buffers before reading a file for encryption or decryption, and also because it is a simplified function set. The results are shown below in *Table 4*.

Finally, we reviewed the standalone Padlock AES Encryption Benchmark tool which allows for a hardware and software AES quick comparison. This is a simple configurable utility only requiring the user press "Start Test" to initiate with default settings of 8,192 byte samples looped 1000 times, and using a 128-bit key. The sample size, loop count, and key size are configurable. The results generally indicate performance increases 20 to 90 times over software across all four cipher modes. Because the source code is not included and it does not actually provide the user with any practical service outside of checking if the ACE is operational, we chose not to rely on this tool alone. It may be possible to attain this source code from VIA separately, however.

3.3 METRICS

Table 3: Saltzer and Schroeder Design Principles Checklist

Metric	Observation
Economy of mechanism	Small, efficient, self-contained and well integrated into the core.
Fail-safe defaults	Simple SDK provides for usability, reducing the chance of improperly accessing ACE.
Complete mediation	This is a low level mechanism.
Open design	Open access for user applications, as well as, operating system processes. Ability to inspect intermediary rounds, and control the extended key sequences.
Separation of privilege	This is a low level mechanism.
Least privilege	Self-contained.
Least common mechanism	Instructions to the ACE are claimed to be atomic and safe for multi-threaded environments.
Psychological acceptability	Straightforward SDK, and familiar/practical sample utilities to showcase the functionality. Also, Linux, OpenBSD, and OpenSSL reportedly already provide some level of ACE support, or plan to.

Table 4: Timing Benchmarks ACE vs. Java AES, 128-bit keys, jdk-1_5_0-windows-i586.exe

(All results in milliseconds)

Metric	ACE encrypt	Java encrypt	ACE decrypt	Java decrypt
Run 1	1322	12,058	1222	11,937
Run 2	1312	11,897	1222	11,908
Run 3	1322	11,877	1232	11,948
Run 4	2284	11,857	1412	11,917
Run5	1322	11,917	1211	11,977
Best	1312	11,857	1211	11,908
Worst	2284	12,058	1412	11,977

3.4 ACE ASSESSMENT

The testing here indicates that Padlock ACE is indeed a relatively fast AES implementation, especially considering it is free, only that the mainboard be acquired. Given its low cost, one could imagine a cost effective ACE grid providing significant cryptographic service potential. Other indicators, such at the Padlock ZIP evaluation and the Padlock benchmarking tool also reinforce this conclusion.

---

4. TRU-DELETE EVALUATION

VIA Tru-Delete Version 1.03 (available for Linux and Microsoft Windows)

4.1 PURPOSE

The purpose of this utility is to demonstrate the low level ability of the VIA hardware based Random Number Generator (RNG) to thoroughly wipe files, folders, and partitions of any trace of their original contents, even from digital forensic tools. This is achieved by completely overwriting specified data with random data making the possibility of recovering deleted data currently infeasible.

4.2 WALK-THROUGH

This evaluation is based on both the Linux and Microsoft Windows platform versions of Tru-Delete. The Linux platform has popular and free to download forensics tools to validate Tru-Delete. In particular, we chose the Sleuthkit and Autopsy (www.sleuthkit.org) combination toolkit to examine the data image artifacts. On the Windows side we chose a demo version of Access Data's (www.accessdata.com) The Forensic Tool Kit (FTK) and Restoration 2514, a free downloadable program. The FTK demo version is limited to 5000 files for

analysis, but identical to the full version in every other respect. Restoration is a minimal program that can identify and selectively restore deleted files. It can also wipe a disk clean with random data.

Once the Tru-Delete executable is run, the main Padlock Tru-Delete window appears *(Figure 1)*. The interface is simplistic and offers the user the ability to wipe the free space of a partition, an entire partition, or specific files and folders. There is also the option of whether the random overwrite data is generated by hardware or software (see *RNG Engine* menu item). This evaluation will examine the hardware only RNG option because Padlock as a whole is the overall interest of this paper.

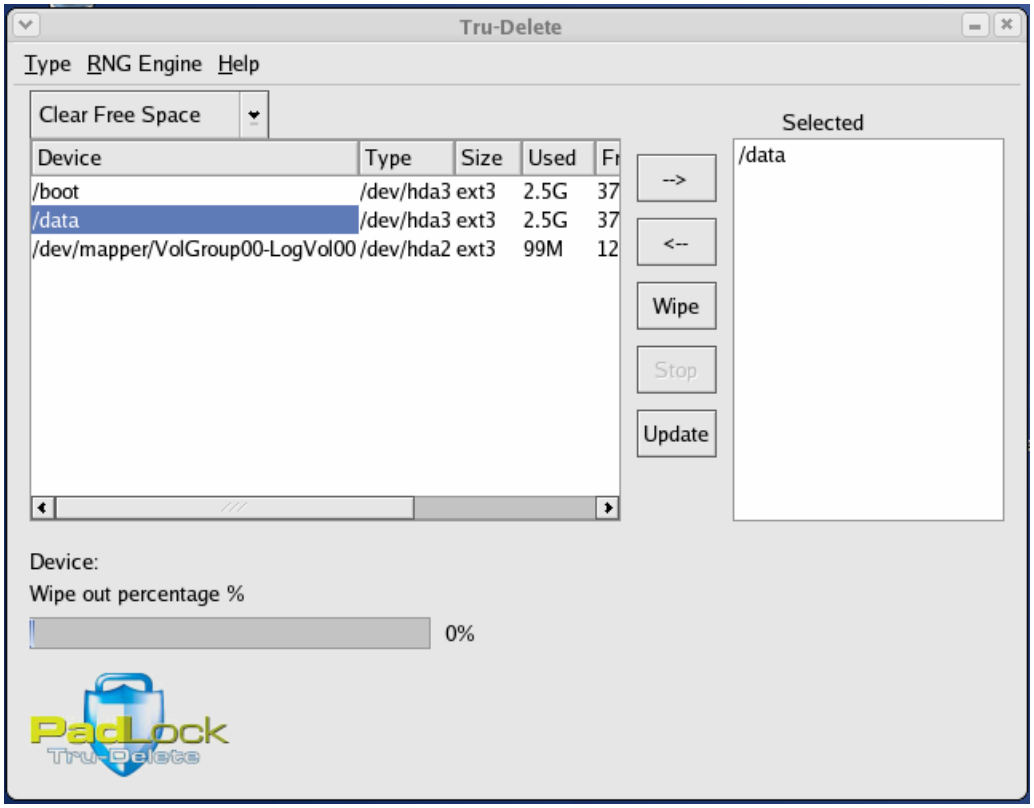
By default, the *Type* menu item is set to "clear or free partition space," as opposed to "clear file," and the partitions are listed on the left of the Tru-Delete window interface. Given that users would more often than not be running Tru-Delete to wipe a particular file or folder, it may be more convenient to default to "clear file" instead. On the right is the "Selected" list-box where the targeted partitions are identified once selected by the user. Above the left list-box of partitions is a choice drop down that offers the two options "Clear Free Space" and "Clear Partition Space." The former only wipes around existing files for each partition targeted, while the latter wipes the entire partition, existing files and all. When the user clicks the "Wipe" button the wipe process begins which includes a helpful progress bar. In our tests a 2.5 GB partition took approximately five (5) minutes to completely wipe. A number of times we experienced the error "mkdir: cannot create directory '\231/rng': No such file or directory" along with "In file VIA_RNG.cpp, line 206: Out of memory" on the first attempt to wipe a targeted partition, but no errors after resubmitting the wipe request. The tool offers a "Stop" button to abort a wipe in progress and an "Update" button to refresh the interface list boxes. The same tests were conducted on Windows XP, with the partition being a small 16MB USB key chain storage device. No errors were encountered and the program took about 31 seconds to wipe the storage device.

When the "clear file" *Type* menu item is selected the left list box no longer displays partitions, but lists the file directory tree instead - in this case starting at the Linux root "/" directory. Under Windows Tru-Delete displays drive mappings within its device column. In "clear file" mode the choice drop down that previously offered the options "Clear Free Space" and "Clear Partition Space" is disabled, and specific files and directories can now be selected for wiping in a manner similar to the partition mode.

Four tests are presented in *Table 6*, all based on a Linux ext3 data partition "/data" containing six (6) jpeg files and one (1) PDF file. Four tests were done. Test one is to simply examine the "/data" image *data1.dd* in Autopsy. Test two is to examine the "/data" image *data2.dd* after using the Linux "rm" command to delete three of the jpeg files. Test three, *data3.dd,* is similar to test two, but utilizing Tru-Delete to wipe the three jpegs. Finally, test four is to examine the "/data" image *data4.dd* after wiping the entire partition with Tru-Delete. As the results show, Tru-Delete prevented the data from being recovered by Sleuthkit and Autopsy by wiping the data, although the filenames themselves were still identifiable in the image.

Four tests for Windows XP are listed in *Table 7*. They are identical to the four tests for Linux and they were repeated for Windows XP using FTK and Restoration 2514. The tests on Windows XP are labeled *exam1*, *exam2*, *exam3* and *exam4*. The results show that Tru-Delete prevented data from being recovered by FTK or Restoration 2514 by wiping with random data. As in the test on Linux, the filenames were still identifiable, but the contents were not. The Windows XP "delete" command did not behave in the same manner as "rm" command in Linux, however, in that the Windows normally deleted files were recoverable using either FTK or Restoration 2514.

Figure 1: Padlock Tru-Delete *(Linux)*



4.3 METRICS

Table 5: Saltzer and Schroeder Design Principles Checklist *(Linux)*

Metric	Observation
Economy of mechanism	Clear and concise presentation, perhaps better to default to file/folder level wipe.
Fail-safe defaults	User must confirm Wipe and Stop Wipe actions.
Complete mediation	Simple checks and balances limited to confirmation message boxes and progress bars.
Open design	It's unclear how to interface with this utility to access its wiping capabilities programmatically, but the source is available.
Separation of privilege	N/A, failed to run as a basic non-root user – segmentation fault.
Least privilege	N/A, failed to run as a basic non-root user – segmentation fault.
Least common mechanism	N/A, failed to run as a basic non-root user – segmentation fault.

Psychological acceptability	Simple user friendly point and click operation and the confidence of low level data erasure.

Table 6: Forensic Tests *(Sleuthkit + Autopsy)*

Metric	Observation
data1.dd	Autopsy identifies all filenames and their contents in the starting image.
data2.dd	Autopsy identifies all filenames, but not the contents of those files deleted by Linux "rm" command.
data3.dd	Autopsy identifies all filenames, but not the contents of those files deleted by Tru-Delete.
data4.dd	Autopsy identifies all filenames, but no data contents as a result of the Tru-Delete partition wipe.

Table 7: Forensic Tests *(FTK + Restoration 2514)*

Metric	Observation
exam1	FTK and Restoration identify all filenames and their contents in the starting image.
exam2	FTK and Restoration identify all filenames, and the contents of those files deleted by the Window's "delete" command.
exam3	FTK and Restoration identify all filenames, but not the contents of those files deleted by Tru-Delete.
exam4	FTK and Restoration identify all filenames, but no data contents because of the Tru-Delete partition wipe.

4.4 TRU-DELETE ASSESSMENT

In conclusion, VIA Padlock Tru-Delete succeeds in wiping data contents from forensic recovery. The file/partition mode option offers flexibility toward varying data wiping policies, however, we look forward to future versions that might offer programmatic interfaces and operating system level user/group authorization views. A final note of interest is the Linux "rm" command also prevented any data recovery, as well, but this, although unexpected initially, may be somewhat misleading in suggesting that "rm" is as secure as Tru-Delete in the face of a more determined attacker. We eventually learned that the Linux ext3 partition type was designed to prevent the recovery of deleted files. In contrast, Window's XP "delete" command allows the deleted data file to be easily restored.

---

5. ZIP EVALUATION

VIA Padlock ZIP Utility Version 1.0 (available for Microsoft Windows)

5.1 PURPOSE

The purpose of this utility is to demonstrate the performance gains by utilizing hardware based encryption in file archiving. ZIP archives are frequently used by end users and therefore a practical avenue in which to showcase VIA Padlock's hardware based Advanced Cryptography Engine (ACE). Padlock ZIP also claims to be compatible with WinZip 9.0.

5.2 WALK-THROUGH

The main Padlock ZIP window appears when executed *(Figure 2 left)*. The appearance is similar to the popular WinZip utility's presentation, but more laconic.

Three main menu items are available; *Archive*, *Action*, and *Help*. Under *Archive* one can choose to create a new archive, open an existing one, or, more importantly, set the tool's options *(Figure 2 right)*. There are options for compression and encryption settings, as well as, disk spanning settings when archives are too big to store on the desired media as a single file. Help is sparse, and some selections are somewhat unfriendly, such as 0 and -1 as compression values "none" and "normal," respectively. Also, there is no password double entry – something which WinZip does offer. Passwords are critical here because, like WinZip, Padlock ZIP applies PKCS #5 (RFC 2898) to generate AES encryption/decryption keys from the password. The actual key size to generate is an optionally 128, 192, or 256 bits.

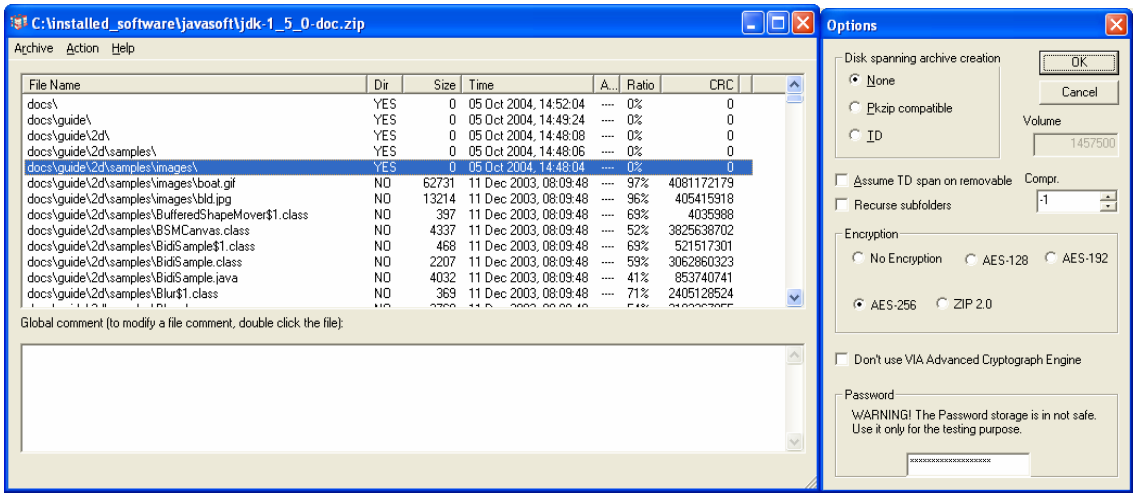
Once the desired options are set one can open an archive created in Padlock ZIP or WinZip 9.0, or create new archives. We found that archives *encrypted* in WinZip 9.0 (SR-1) do not open in Padlock ZIP because of a header version mismatch. This was confirmed in Microsoft Visual C++ 6.0 Debugger, however, using WinZip 9.0 we could open an archive originally *encrypted* in Padlock ZIP.

The *Action* menu allows some of the same functions seen in WinZip; Adding, Deleting, Extracting and Testing archive member files.

Our sense is that this utility needs to be polished further. Terse error messages required us to spend considerable time running through the source code in a debugger, and unrefined documentation required us to wonder and experiment more than we would have liked. For example, to learn that Padlock ZIP stores all the options including the encryption password in the Windows registry was surprising and time consuming. We also expected more guidance in selecting proper passwords for desired key lengths. Unwary users could gain a false sense of security if passwords are not properly constructed.

On a positive note, we were able to witness the hardware accelerated encryption performance boost. This was not true for all cases though, as the metrics below indicate. Archives of fewer, but larger monolithic files garnered higher performance gains from the ACE enabled Padlock ZIP. WinZip 9.0 was fastest when the archives contained an exceedingly large number of smaller files, such as the uncompressed Sun Java 1.5 JDK documentation files along with the JDK setup executable, as depicted in *Table 9*.

Figure 2: Padlock ZIP *(Windows XP)*



5.3 METRICS

Table 8: Saltzer and Schroeder Design Principles Checklist

Metric	Observation
Economy of mechanism	Padlock ZIP is too Spartan – more verbose option descriptions would be beneficial.
Fail-safe defaults	Encryption settings must be set before use. Compression is set by default.
Complete mediation	Not enough explanative assertions – invalid actions display ambiguous errors.
Open design	WinZip compatibility (almost), Zip file format, Advanced Encryption Standard (AES).
Separation of privilege	Relies on operating system for general file access, but also password enforces encrypted archives.
Least privilege	Relies on operating system (file attributes etc).
Least common mechanism	Storing encryption password and options in Windows registry violates this principle.
Psychological acceptability	WinZip look and feel and intended compatibility.

Table 9: Padlock ZIP Timing Benchmarks *(AES-256 Keys, 45 MB JDK 1.5 setup executable and 230 MB of the 10,779 JDK 1.5 documentation files)*

(Results from two separate PCs, each with the same VIA MII 12000 mainboard)

Metric	Padlock with ACE		Padlock without ACE		WinZip 9.0 SR-1	
AES + Normal Compression [JDK + Docs]	15m 52s	15m 48s	14m 37s	14m 39s	12m 18s	12m 21s
AES + No Compression [JDK + Docs]	13m 59s	13m 55s	14m 26s	14m 24s	11m 45s	11m 41s
AES + Normal Compression [JDK only]	25s	26s	31s	31s	35s	34s
AES + No Compression [JDK only]	7s	7s	11s	10s	11s	11s

Table 10: Padlock ZIP Timing Benchmarks *(monolithic files)*

Metric run #	Padlock w/ ACE	Padlock no ACE	WiZip 9.0 SR-1	Padlock w/ ACE	Padlock no ACE	WiZip 9.0 SR-1	Padlock w/ ACE	Padlock no ACE	WiZip 9.0 SR-1
	64 MB jpeg			45 MB JDK 1.5 setup exe			632 MB Fedora C3 ISO disk 1		
1	43s	47s	59s	26s	31s	38s	6m 9s	6m 52s	7m 38s
2	42s	47s	59s	27s	31s	38s	6m 6s	6m 54s	7m 37s
3	42s	47s	58s	27s	31s	38s	6m 5s	6m 51s	7m 35s
4	42s	47s	59s	26s	32s	38s	6m 5s	6m 52s	7m 36s
5	42s	48s	59s	27s	31s	38s	6m 6s	6m 50s	7m 38s

5.4 ZIP ASSESSMENT

We feel that a more polished ZIP tool would better serve users with the full advantages of VIA Padlock's hardware based encryption. The polished usability of WinZip 9.0 with the performance of Padlock ZIP would be closer to the mark. The timing benchmarks in *Table 9* and *Table 10* reveal that Padlock ZIP is fastest for archiving and encrypting large monolithic files, but not when the archives consist of a large number of smaller files.

6. CONCLUSIONS

We conclude that the value to performance and value to features ratios reflect well on VIA Padlock, and that the claims made by the manufacturer indeed have significant merit. The metrics garnered during this evaluation should offer the reader a frame of reference to the degree of improvement in performance and output quality over typical software-only solutions. Where competing hardware solutions exist, one should also consider the cost effectiveness of Padlock, and its potential to be deployed as a formidable embedded device in the field, or in a grid or array topology to multiply its service capacities. Clearly, the hardware suite of Padlock (RNG, ACE) is the heart and soul of the Padlock initiative. This along with the SDK is really what VIA is presenting, where as the software suite aspect we experimented with did not seem quite as polished. Padlock Tru-Delete and ZIP

are more like showcase pieces introduced in open source to inspire developers to utilize the power of the hardware suite. Moving forward we noted some indications that Linux, OpenBSD, and OpenSSL are already working to support Padlock on some level. VIA is also expected to release a follow up to the Nehemiah core early in 2005, named Esther, which will introduce hardware acceleration for the Secure Hash Algorithm (SHA-1, SHA-256) and Rivest, Shamir, Adleman (RSA) encryption algorithm as part of ACE. The wonderful potential of this technology is worth a closer look by anyone interested in contemporary computer security issues.

7. NOTES

1. Time constraints did not allow for an evaluation of Padlock Secure VPN, however, it relies on RNG and ACE, and is available for Linux only (2.6 kernel). Secure VPN is open source.

2. Centaur Technology, a subsidiary of VIA, is a primary design force behind RNG and ACE.

3. From the Padlock ZIP source and the Winzip website, it appears that the Padlock ZIP and Winzip 9.0 AES encryption routines are written by the same author, Dr. Brian Gladman.

4. Various tests in this paper were run on two separate VIA EPIA Mini-ITX MII 12000 PCs, each with a Windows XP and Linux 2.6 kernel dual boot. One machine's Linux was Fedora, the other, Red Hat Enterprise Workstation Beta 2. Both builds were done by the authors of this paper, and each build included 512 MB RAM.

8. REFERENCES

Centaur Technology, Inc. & VIA Technologies, Inc., *VIA C5P Security Application Note,* Centaur Technology, Inc., & VIA Technologies, Inc., 2003.

Centaur Technology, Inc. & VIA Technologies, Inc., *VIA Nehemiah Advanced Cryptography Engine,* Centaur Technology, Inc., & VIA Technologies, Inc., 2003.

Centaur Technology, Inc. & VIA Technologies, Inc., *VIA Nehemiah Random Number Generator Programming Guide*, Centaur Technology, Inc. & VIA Technologies, Inc., 2003.

Garms, Jess & Somerfield, *Daniel, Professional Java Security*, WROX, United Kingdom, 2001.

Grundy, Barry*, Law Enforcement and Forensic Examiner Introduction to Linux, A Beginner's Guide,* NASA, 2003.

Marsaglia, George, *The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness*, Florida State University, Florida, 1995.

McCullough, B., *Assessing the Reliability of Statistical Software: Part I*, The American Statistician, 1998, Volume 52, Issue 4, 358-366.

Daemen, Joan & Rijmen, Vincent, *The Design of Rijndael*, Springer, Germany, 2002.

VIA Technologies, Inc*, VIA Padlock SDK API & Programming Guide*, VIA Technologies, Inc., 2004.

VIA Technologies, Inc, *Padlock ZIP User Guide*, VIA Technologies, Inc., 2004.

VIA Technologies, Inc, *VIA Padlock Hardware Security Suite, Providing the Hardware Building Blocks for Optimizing Information Security*, VIA Technologies, Inc., 2004.

VIA Technologies, Inc, *VIA Padlock Tru-Delete User Guide, in Linux, Windows 2000/XP, and Windows CE,* VIA Technologies, Inc., 2004.