Comparing the Power of Full Disk Encryption Alternatives

Aaron Fujimoto, Peter Peterson, Peter Reiher
Computer Science Department
UCLA
Los Angeles, CA
aaronfuj@gmail.com, pahp@cs.ucla.edu, reiher@cs.ucla.edu

Abstract— **This paper examines the energy costs of different approaches to full disk encryption, including hardware encryption, software encryption, and "no encryption". Using the DEEP power measurement platform, we measured the energy consumed by each configuration under various workloads. We demonstrate that hardware encryption saves energy for many (though not all) workloads, but that the energy savings may not compensate for the hardware price at current rates.**

Keywords-component; energy use, full disk encryption, file system performance measurement

I. INTRODUCTION

As computers grow to process and store greater amounts of sensitive information, there is an increasing need for privacy and security. Encryption is perhaps the most effective approach, and can be done on individual files or on entire disks. While the piecemeal approach is simpler, the full disk encryption (FDE) approach is generally more foolproof. There are two main methods to perform FDE. Either the OS transparently encrypts and decrypts all disk data using the CPU, or special-purpose hardware performs the necessary operations without any runtime software support.

This paper investigates the issue of the relative power consumption of hardware and software FDE, using direct measurement of live workloads. In our tests, hardware FDE typically used only slightly more energy and time than an unencrypted disk. Furthermore, hardware FDE was faster and energy-cheaper than software FDE. However, the magnitude of the savings was not as great as may be expected.

II. EXPERIMENTAL METHODOLOGY

A. Measurement Technology

Our experiments were performed on a UCLA Atom DEEP [1] platform, which allows us to measure the instantaneous power and energy consumed by the system's processor, RAM, hard disk, USB, and power supply. DEEP instrumentation allows us to specify precisely when to start and stop measuring power use, and can synchronize data to particular code segments.

978-1-4673-2154-9/12/$31.00 ©2012 IEEE

DEEP's ability to simultaneously measure discrete components is important for this investigation because in addition to system-level energy effects due to runtime factors, software and hardware FDE will have different component-level power profiles. For example, DEEP allows us to measure how much more energy the CPU uses when performing software FDE and how much more energy the hardware FDE disk consumes.

B. Hard Disks

While we would have preferred to use a single test disk with FDE hardware that could be enabled or disabled on a per-test basis, such a disk was not available at the time of this experiment. Therefore, we attempted to identify two very similar disks --- one with encryption hardware and one without. For our standard disk, we chose the Seagate Momentus 7200.4 SATA 3Gb/s 250-GB Hard Drive (ST9250410AS) [2]. For our encrypting disk, we chose the Seagate Momentus 7200 FDE Laptop with FIPS 140-2 250-GB Hard Drive (ST9250412AS) [3] which performs AES-128 [4] in hardware. Both are 2.5 inch laptop-class hard disks with very similar specifications and performance ratings, making them ideal for comparison. The OS (Linux) was installed on a separate disk to minimize experimental noise.

Software encryption was performed by dm-crypt [5], which uses the Linux Crypto API. This standard approach enables FDE through the use of a cryptographic layer between the kernel and the disk's block device. Since the Momentus 7200 FDE performs AES-128 encryption [4], we also used AES-128 via dm-crypt along with AES-256 (to investigate the effect of key size) and "no encryption" as a baseline.

III. TEST DESIGN

A. General Issues

A hard disk is used to permanently store files and data in a computer. Therefore the tests we used were designed to access these files and data in various ways while limiting the amount of power consumed by unrelated tasks. Careful consideration was used when designing the tests to avoid possible caching of data both in the kernel's cache (by executing commands to flush the kernel's page cache) and in the buffer of the hard disk itself. One way of attempting to avoid caching files in the hard disk (without rebooting between each test) is to use multiple copies of test data instead of reusing a single source.

The tests exercised writing and reading directly to and from the disk, compressing and decompressing files on the disk, querying a database stored on the disk, searching through files on the disk, and playing back a video file from the disk. These various tasks exercised different behaviors of the disk, such as extended sequential access, read versus write, and random access to allow us to determine if there were energy differences based on these use cases. These tests are meant to be reasonably realistic and varied, but we do not claim that they are comprehensive or fully disjoint in their modes of use.

B. *File Read and Write Performance*

One direct way to measure the difference in power consumption over the different configurations was to directly write a file to the test disk and then read a file from the test disk. We tested multiple small (10 MB) files designed to fit into disk cache and a large file (1 GB) designed to overflow the hardware cache and force sustained operation. When writing, files were created by reading directly from /dev/zero. Subsequent reads were performed by copying the files from disk to /dev/null (discarding the output).

C. *Kernel Decompression and Compression*

A number of informal file system benchmarks and workloads appear frequently in the literature, such as the Modified Andrew Benchmark [16], decompressing and compiling the Linux kernel, and others. [17] These tests typically create a large number of files and directories and perform various operations on them. While these non-standardized benchmarks may be difficult to relate to one another, they can still be useful when one test is used in a head-to-head comparison between test systems. For this type of test, we copied the compressed source of the Linux Kernel to our test disk from the separate system disk, decompressed it, re-compressed it, and deleted it. This exercised a variety of disk operations, such as reading and writing of large and small files, creating and removing nested directories, compression, and decompression.

D. *SQL Query*

This test (from [6]) focused on database access. We created a MySQL database, loaded data into the tables, performed a few queries, and then removed the database. SQL queries provide a different method of accessing information than the sequential reads/writes that are exercised in the basic read/write and kernel extraction/compression tests.

E. *Kernel Grep*

Another common disk operation is searching. Searching may involve looking through several directories for a specific string. We used Linux's grep tool to perform a search on a directory containing an extracted Linux kernel. We recursively searched for all instances of the word "module" in all of the files. This exercised reads of several different files scattered through multiple directories over a fairly short period. We expected the results to be similar to the basic read/write tests.

F. *Video Player*

Computers are often used to play audio and video files. We measured the power to play a short video clip [7] using the Linux media player mplayer.

IV. Test Results

To handle statistical variation, we performed 10 to 20 runs of most tests, which allowed us to obtain good confidence intervals at the 95% level. For results that showed very close power use for different cases, we ran the tests several extra times to further shrink the confidence interval and better understand whether the alternatives displayed statistically significant differences.

We report various kinds of data corresponding to the individual tests. In some cases, we show the instantaneous average power (in watts) used by the components. This measurement gives a sense of how hard each component worked during that test. In other cases, we give the energy used by various components, which is the product of the power and the run time of the test. In other cases, we report the total system energy used.

A. *File Read and Write Performance Results*

This test was the most direct comparison of power consumption for both cached and sustained writes and reads. To avoid caching effects, we dropped the kernel page cache and synced the disk between each file creation. Ultimately, 10 MB writes consumed a statistically similar amount of energy on each component for all configurations (Figure 1), and all configurations took about the same amount of time. The average power per component was also similar across configurations, with only a slight difference found in the HDD and RAM.

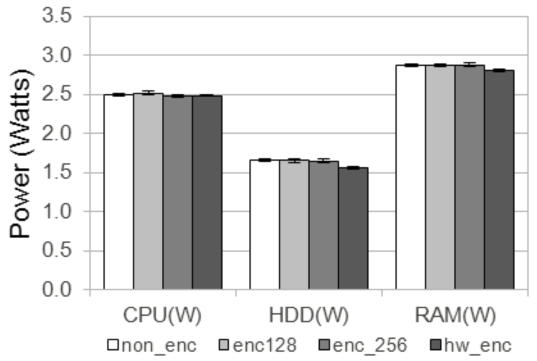

Figure 1. Power Used for 10MB File Writes

Reading 10 MB files showed more significant results. For the hardware FDE disk, the overall time was (0.154 ± 0.013s) which was only slightly longer than the time for the baseline disk without encryption (0.151 ± 0.005s). Software encryption with AES-128 and AES-256 took significantly longer at 0.445 ± 0.005s and 0.564 ± 0.003s respectively, resulting in a significant increase in energy consumption as seen in Figure 2. Figure 3 shows that while some of the difference in energy used is due to increased run time, there are also differences in the average amount of power used for the different alternatives.

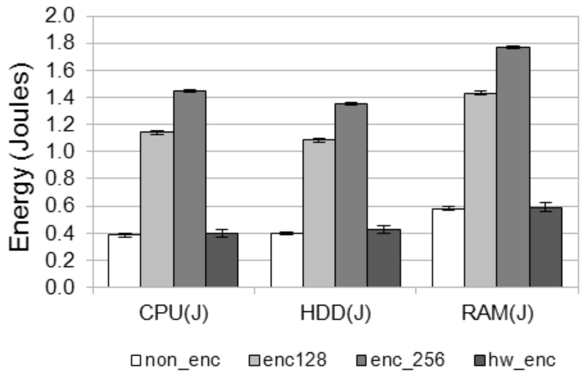

Figure 2. Energy Used for 10MB File Read

Average CPU power is similar in all configurations, while HDD and RAM have higher average power use in the non-encrypted and hardware FDE configurations versus software FDE as seen in Figure 3. This makes sense because the HDD and RAM are utilized less efficiently during the software FDE task (which spends most of its time performing encryption in the CPU). Also, we see that the hardware FDE disk has a higher average power for HDD than the standard disk without encryption, presumably due to the cryptoprocessor and related hardware. However, as Figure 2 shows, because software FDE takes longer to complete, the overall energy used to perform the operation is much higher than the energy used to perform the same operation without encryption or using hardware FDE.

Writing the 1 GB file further shows the difference between the software FDE process and no encryption or hardware FDE. Hardware FDE is only slightly slower than no encryption at 35.599 ± 0.999s vs. 34.172 ± 0.288s. Software FDE is significantly slower than both (58.824 ± 0.679s for AES-128 and 72.425 ± 1.085s for AES-256). In other words, the overall energy consumed follows the same trend as reading the 10 MB file.

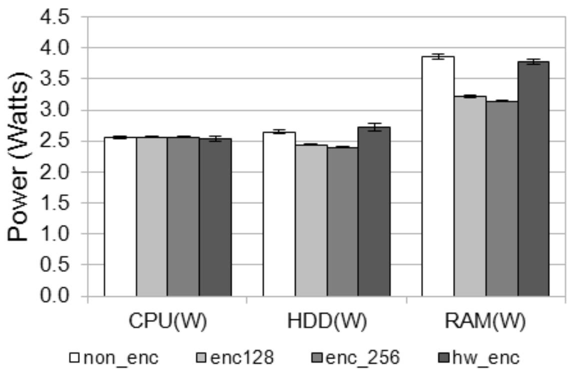

Figure 3. Power Used for 10MB File Read

B. Kernel Decompression and Compression Results

The most significant portions of this test were the decompression and compression operations. Decompressing the kernel source confirmed the results of the single file test, namely that software encryption took longer to execute a task than no encryption or encryption done in hardware. The standard disk was able to perform the task in 80.464 ± 0.903s on average, while the disk using hardware FDE was able to extract the files in an average of 80.946 ± 3.373s. Software FDE required an average of 100.279 ± 15.444s using AES-128 encryption and 107.093 ± 1.839s using AES-256 encryption. This large difference in time was reflected in the energy consumed by each component for the software configurations. The overall energy used by the system during this task also shows a correlation with the time required to perform the task. The standard disk consumed 2414.687 ± 30.313 J, hardware FDE consumed 2455.955 ± 17.425 J, the software FDE using AES-128 consumed 3033.972 ± 52.530 J, and the AES-256 software FDE configuration consumed 3229.965 ± 72.550 J.

We found the instantaneous average power per component to be fairly similar across configurations, except for the RAM (which followed what was seen in the Single File tests). Since the instantaneous power across configurations was similar, we were able to determine that the overall energy used for this task was based on the time spent, rather variations in component energy use.

Compressing the directory into a file produced the same ordering of costs for the various options, but the differences between software encryption and the other options was smaller which was reflected in the run times for the various encryption options.

Looking at the kernel decompression/compression test as a whole (Figure 4), the total energy used by all components reflects the results from the extraction and compression phases. The system with no encryption used the least amount of energy; the system using hardware encryption only consumed slightly more energy; and both software-encrypted systems consumed much more energy than the other two configurations.

An interesting anomaly in this test was that the energy cost of software FDE with AES-256 was often less than that for AES-128. One would not expect the larger key size to run faster and use less energy than the smaller, but it did (with statistical confidence). This question was not core to the purpose of our study, so we did not investigate it further.

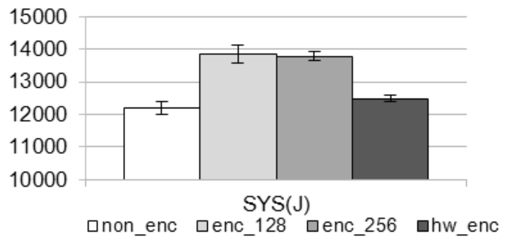

Figure 4. Total System Energy For Kernel Decompression/Compression (All Phases)

C. SQL Query Performance Results

Unlike the other tests, the SQL query performance results showed no significant difference in either the time required or the energy consumed to execute the query between any of the configurations. One explanation for the similar performance could be the small size of the database. This may have resulted in less disk IO and more computation. In other words, the energy used for decryption may have been small compared to the energy consumed by the MySQL process itself.

D. *Kernel Grep Performance Results*

We ran the Kernel Grep test 10 times on separate directories with each directory containing copies of the same structure and files, which proved more than enough to achieve our desired confidence interval.

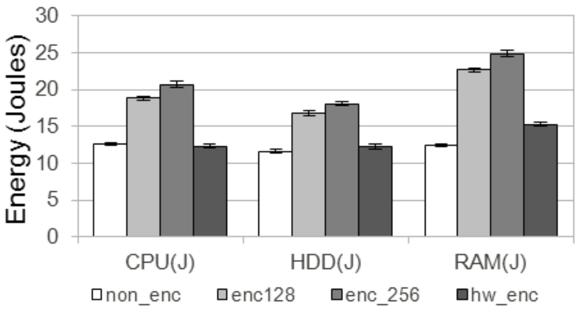

Figure 5. Energy Used in Kernel Grep Tests

Once again, the time required to perform this test was lower using the disk with no encryption and the disk with hardware FDE than either of the software configurations. The disk with no encryption took 5.175 ± 0.090s while the disk with hardware encryption took 5.055 ± 0.106s. Software FDE with AES-128 took 7.59 ± 0.104s while AES-256 took 8.346 ± 0.135s. Figure 5 shows the energy used by various systems components for this test using the different encryption options.

E. *Video Player Performance Results*

The video playing test was run a total of 60 times, resulting in a maximum margin of error found in a single component to be 2%. Accordingly, we show full system power in Figure 6. (Note that the scale on this figure starts at 2100 J.)

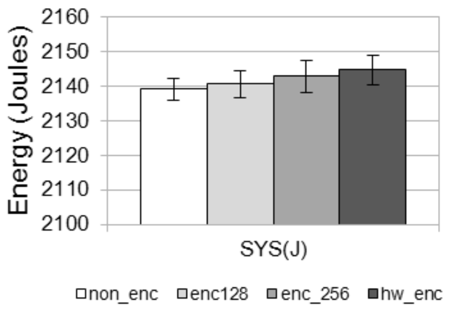

Figure 6. Total System Energy for Video Player Tests

One plausible explanation for why we see so little difference among the configurations is that the energy spent to play the video far exceeds the energy required to access and decrypt the video file. Another possible explanation is that since the timing of the task is tied to the video playback, faster options (e.g. "no encryption") still pay the cost of longer runtimes. With the use of DEEP's energy caliper capability, we could verify these hypotheses, but we have not done so yet.

V. DISCUSSION

Having seen the differences in power consumption between full disk encryption implemented in software and hardware, we can consider whether this difference is significant enough to warrant using one configuration over the other for efficiency's sake.

In most cases less energy is used when using hardware FDE than software FDE. The hardware solution is usually faster than the software solution, and the hardware-encrypting disk did not use considerably more energy than the standard disk; therefore, less overall energy is used with hardware versus software FDE. In fact, we saw that in most tests the hardware-encrypted disk performed at or near the level of the disk with no encryption at all, indicating that the cryptoprocessor is not a throughput bottleneck or energy-expensive. Thus, for the sole purposes of access time and energy consumed, the hardware-encrypted disk is more efficient than the software solution. This is not entirely surprising, since hardware implementations of algorithms are usually more efficient than equivalent software implementations.

However, there are additional concerns that may not make the hardware configuration the desired solution in all scenarios. While the hardware-encrypted disk typically consumes less energy, the software FDE solution costs less to acquire. Hardware FDE disks currently cost more than non-encrypting disks, and software encryption is essentially free in terms of hardware costs. At the time of this writing, the test disks in question cost around \$65 for the standard disk and \$90 for the hardware-encrypted disk. This is more than a 30% increase in disk cost. Since commodity hardware manufacturers care deeply about even small component costs, a difference of this magnitude is likely to keep hardware FDE as a premium option at current prices, even if it were offset with bulk purchases.

What about an individual pricing a new laptop? Suppose a user requires FDE and is interested in a hardware solution. How long will it take for the extra cost to be offset by power savings? Recent electricity costs in Los Angeles are around \$0.12/KWatt hour. There are 3,600,000 joules per kilowatt hour. As shown in Figure 4, using hardware full disk encryption saves around 1000 joules on one run of our kernel decompression/compression test. Therefore, to make up a \$25 difference in disk cost on the basis of power use, one would need to run this test almost 750,000 times over the lifetime of the machine in order to recoup the cost. This test takes a bit less than 2 minutes, so running it 750,000 times would take roughly 25,000 hours, which is almost 3 years --- about the lifetime of many laptop computers. Of course, nobody runs the equivalent of this test for nearly three solid years, and the relative benefit of other workloads is probably lower, on the whole. In total, this suggests that hardware FDE is not a very good investment in terms of simple economics.

There are other possible benefits of hardware encryption, of course. Battery lifetime is an important statistic for most portable devices. (The Atom DEEP uses an Intel Atom processor and is similar to a netbook or tablet in terms of hardware.) In this mode of use, decreased energy use is much more important, since it impacts how long one can work before the battery is exhausted. One can purchase batteries for laptop computers that advertise 50-60 watt hours, which is roughly 180,000 joules. If the entire power budget of such a battery was used to run the kernel decompression/compression test, it would run around 15 times with no encryption, 14 times with hardware FDE, and 13 times with AES-128 software FDE – a meaningful difference, but not tremendous.

These “back of the envelope” calculations are admittedly dependent on many limiting assumptions involving our experiments, workloads, hardware, and local energy costs. However, they do provide interesting food for thought and give a general sense of how much more battery life one can get out of a laptop using hardware full disk encryption versus software: some, but not a lot.

The results we report show significant differences in the power saved based on the workload. In essence, workloads consisting primarily of intense disk activity will benefit more from hardware FDE than those that have little disk I/O compared to computation. These workloads are useful as worst case scenarios, but may not be realistic use cases for mobile users.

One issue we have only briefly touched upon in this paper is access times. While the energy used by the overall system is important, a user of the disk may find it more crucial to have quick access times to the disk. In general, hardware FDE is more responsive versus software, so this could be a significant factor in some cases. On the other hand, caching will also tend to minimize the differences between configurations, since cached data will not exercise any form of encryption. We did not study caching explicitly, but the degree of care required to minimize caching effects in our tests suggest that effective caching could further reduce the energy value of hardware FDE.

Also, the security issues of encryption done in hardware versus software may justify the use of one solution over the other for certain users. Both configurations have their own vulnerabilities and neither is immune to all attacks, so issues of the style of use of the machine and the expected forms of attack could make a difference on which alternative is preferable.

The research reported here was designed primarily to discover the differences in energy used by software and hardware encryption, rather than to identify the reasons for these differences. There are a number of interesting future directions for research based on obtaining a deeper understanding of why various encryption alternatives performed as they did, and possible alterations of the system to improve their performance.

VI. Related Work

Energy measurement or characterization systems are not new [8][9], although they are often special systems built in-house for a specific experimental purpose. [9][14] [13][15] Many energy measurement projects measure only a single component or full system energy. Due to its ubiquity on laptops, the ACPI battery interface is a popular source of information, although its resolution is known to be poor [9].

In contrast, the DEEP project [1] intends to provide a kind of “standard platform” for energy experimentation. It is under active development, has a simple, easy-to-build design and provides novel features in terms of accuracy, synchronization, instrumentation, and more. DEEP enabled us to synchronize real code execution to component level energy data with very fine granularity to simultaneously capture energy effects across the CPU, RAM, and disk itself, in real time. DEEP's simple instrumentation makes it trivial to create and test new workloads in a commodity, off-the-shelf environment.

While our experiment was a head-to-head comparison of alternatives, it is in some sense related to the larger issue of energy accounting and management in operating systems. Large projects [9][10] have treated energy as a first-class resource to be managed, and in so doing have highlighted many issues and potential solutions for relevant problems. In particular, one issue that is important for comprehensive energy accounting in modern operating systems is accurately accounting the cost of delayed or asynchronous operations, such as network transmission following a user request. Resource Containers [11] are one method for attempting to capture these costs however this solution requires kernel modifications.

DEEP uses an off-the-shelf version of Linux, and our work on this experiment did not incorporate kernel modifications. As a result, our data describes the energy consumed between the start and the end of the user-initiated activity. Thus, it does not include the cost of delayed writes and other kinds of asynchronous operations may not be captured. However, because our experiments were head-to-head comparisons under controlled conditions, we are confident that the comparison is fair (although it likely underestimates the true cost of asynchronous operations such as delayed writes). Work on DEEP continues towards making it multi-process and preemption aware, as well as including more comprehensive energy accounting mechanisms.

To our knowledge, no one has performed a comparable head-to-head comparison of hardware and software cryptography for disks. However, this work falls into the larger body of work investigating the direct energy costs of security choices in general. Li and Xu [13] investigated the direct energy costs and benefits of offloading some security-related operations. Potlapally, et al. [12] investigated the varying direct energy costs of various software cryptosystems (and noted that hardware represented a path to greater efficiency). We previously [6] used the DEEP platform (then called Atom LEAP) in an undergraduate research course at UCLA investigating the energy costs of various security technologies.

Finally, the results of this project support the conclusions drawn by Dawson-Haggerty, et al. [14] in that, for many computer systems, frequency scaling and other efficiency strategies are often beaten handily by the so-called "race to sleep," where the best strategy (from an energy perspective) is the one that simply finishes fastest. This is due to the large static energy cost of most hardware, which can dwarf the combined costs of other system components. When this happens, the savings of "finishing first" typically is greater than the component level differences that may exist between various strategies. However, while our results are consistent with this concept, the idea itself does not offer insights into the relative costs of hardware versus software full disk encryption.

VII. CONCLUSION

Full disk encryption (FDE) provides security benefits, but has energy and associated economic costs. These costs can be statistically insignificant or very large, depending on the workload and the FDE implementation used.

Hardware FDE almost always uses less energy than software cryptography and in many cases is statistically indistinguishable from using no encryption whatsoever. In particular, hardware FDE tends to be significantly more efficient than software FDE under especially intense workloads. Conversely, workloads that use the disk but are time bound (such as video playback) see little benefit from hardware FDE.

While not rigorously tested here, caching effects are likely to reduce the observed differences in energy use between the hardware and software alternatives. As a result, the lower energy use of the hardware alternative might not lead to any benefits that would be meaningful to an individual user, especially in more realistic use cases.

This research studied primarily the effects of different full disk encryption alternatives, not the causes of those effects. A more detailed study (which the DEEP approach would allow) could reveal opportunities for making greater reductions in energy use when performing full disk encryption or when performing general file system operations. This possibility remains an interesting avenue for future research.

ACKNOWLEDGMENT

This work was supported by National Science Foundation Grant CNS-1116809.

REFERENCES

[1] D. Singh, P. Peterson, W. Kaiser, and P. Reiher, "DEEP: Decision Support for Energy Efficient Processing," submitted to IEEE Transactions on Computers, 2011.

[2] Seagate, "Momentus 7200.4 SATA Product Manual," http://www.seagate.com/staticfiles/support/disc/manuals/notebook/momentus/7200.4/100534376c.pdf, May 2011.

[3] Seagate, "Momentus 7200 FDE.2 SATA Product Manual", http://www.seagate.com/staticfiles/support/disc/manuals/notebook/momentus/7200%20FDE.2/100571983b.pdf, May 2011.

[4] Seagate, "Full Disk Encryption FAQs - Pre Sales", http://seagate.custkb.com/seagate/crm/selfservice/search.jsp?DocId=206011&NewLang=en&Hilite=aes#12, May 2011.

[5] C. Saout, "dm-crypt: a Device-mapper Crypto Target," http://www.saout.de/misc/dm-crypt/, May 2011.

[6] P. Peterson, D. Singh, W. J. Kaiser, and P. L. Reiher, "Investigating energy and security trade-offs in the classroom with the atom LEAP testbed," Proceedings of the 4th USENIX Workshop on Cyber Security Experimentation and Test, 2011.

[7] NASA, Apollo 11 Lunar Surface Journal, Journal Text 110:13:4, http://www.hq.nasa.gov/alsj/a11/a11v_1101342.mpg.

[8] J. Flinn and M. Satyanarayanan, "PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications," Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications, 1999.

[9] H. Zeng, C. Ellis, A. Lebeck, and A. Vahdat, "ECOSystem: Managing Energy as a First Class Operating System Resource," ACM SIGOPS Oper. Syst. Rev., 2002.

[10] R. Neugebauer, and D. McAuley, Derek, "Energy Is Just Another Resource: Energy Accounting and Energy Pricing in the Nemesis OS," In Proceedings of the Eighth USENIX Workshop on Hot Topics in Operating Systems, 2001.

[11] G. Banga, P. Druschel and J. Mogul, "Resource Containers: A New Facility for Resource Management in Server Systems," In Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation, 1999.

[12] N. Potlapally, S. Ravi, A. Raghunathan, N. K. Jha, "Analyzing the energy consumption of security protocols," In Proceedings of the international symposium on Low power electronics and design, ACM 2003.

[13] Z. Li and R. Xu, "Energy impact of secure computation on a handheld device," IEEE International Workshop on Workload Characterization, 2003.

[14] S. Dawson-Haggerty, A. Krioukov and D. Culler, "Power Optimization – A Reality Check," Technical Report Identifier: EECS-2009-140, University of California, Berkeley, 2009.

[15] Windows Internet Explorer Engineering Team, "Browser Power Consumption – Leading the Industry with Internet Explorer 9," http://blogs.msdn.com/b/ie/archive/2011/03/28/browser-power-consumption-leading-the-industry-with-internet-explorer-9.aspx, January 2012.

[16] J. Ousterhout, "Why Arent Operating Systems Getting Faster As Fast As Hardware?" USENIX Summer Conference, June 1990.

[17] V. Tarasov, S. Bhanage, and E. Zadok, "Benchmarking File System Benchmarking: It *IS* Rocket Science," USENIX 13th Workshop on Hot Topics in Operating Systems, May 2011.