Chapter 16

# PASSWORD CRACKING USING SONY PLAYSTATIONS

Hugo Kleinhans, Jonathan Butts and Sujeet Shenoi

**Abstract** Law enforcement agencies frequently encounter encrypted digital evidence for which the cryptographic keys are unknown or unavailable. Password cracking – whether it employs brute force or sophisticated cryptanalytic techniques – requires massive computational resources. This paper evaluates the benefits of using the Sony PlayStation 3 (PS3) to crack passwords. The PS3 offers massive computational power at relatively low cost. Moreover, multiple PS3 systems can be introduced easily to expand parallel processing when additional power is needed. This paper also describes a distributed framework designed to enable law enforcement agents to crack encrypted archives and applications in an efficient and cost-effective manner.

**Keywords:** Password cracking, cell architecture, Sony Playstation 3

## 1. Introduction

In 2006, Sebastien Boucher was arrested for possessing child pornography on his laptop while attempting to cross the Canadian–U.S. border [7]. When the FBI attempted to access his computer files at their laboratory, they discovered that the contents were password protected. The FBI requested the U.S. District Court in Vermont to order Boucher to reveal his password. However, the court ruled that Boucher was protected under the Fifth Amendment because a person cannot be compelled to "convey the contents of one's mind." According to John Miller, FBI Assistant Director of Public Affairs, "When the intent...is purely to hide evidence of a crime...there needs to be a logical and constitutionally sound way for the courts to allow law enforcement access to the evidence."

The technical solution in the Boucher case is to crack the password. However, password cracking involves advanced cryptanalytic techniques, and brute force cracking requires massive computational resources. Most law enforcement agencies neither have the technical resources nor the equipment to break passwords in a reasonable amount of time. In 2007, a major U.S. law enforcement agency began clustering computers at night in an attempt to crack passwords; these same computers were used for other purposes during the day. The approach represented an improvement over the existing situation, but it was still inadequate.

The Sony PlayStation 3 (PS3) system is an attractive platform for password cracking. The PS3 was designed for gaming, but it offers massive computational power for scientific applications at low cost. The architecture also permits the interconnection of multiple PS3 systems to enhance parallel processing for computationally-intensive problems.

This paper evaluates the potential benefits of using PS3 to crack passwords. An experimental evaluation of the PS3 is conducted along with an Intel Xeon 3.0 GHZ dual-core, dual-processor system and an AMD Phenom 2.5 GHz quad-core system. The results indicate that the PS3 and AMD systems are comparable in terms of computational power, and both outperform the Intel system. However, with respect to cost efficiency ("bang per buck") – arguably the most important metric for law enforcement agencies – the PS3, on the average, outperforms the AMD and Intel systems by factors of 3.7 and 10.1, respectively.

This paper also describes a distributed PS3-based framework for law enforcement agents in the field. The framework is designed to enable agents to crack encrypted archives and applications in an efficient and cost-effective manner.

## 2. Cell Broadband Engine Architecture

Kutaragi from Sony Corporation created the cell broadband engine (CBE) architecture in 1999 [1, 2, 12]. His design, inspired by cells in a biological system, led to a large-scale collaborative effort between Sony, Toshiba and IBM [4]. The efforts focused on developing a powerful, cost-effective architecture for video game consoles. Although the CBE was designed for gaming, the architecture translates to other high-performance computing applications.

### 2.1 Overview

Over the past decade, microprocessor design has increasingly focused on multi-core architectures [9]. A multi-core architecture, when used

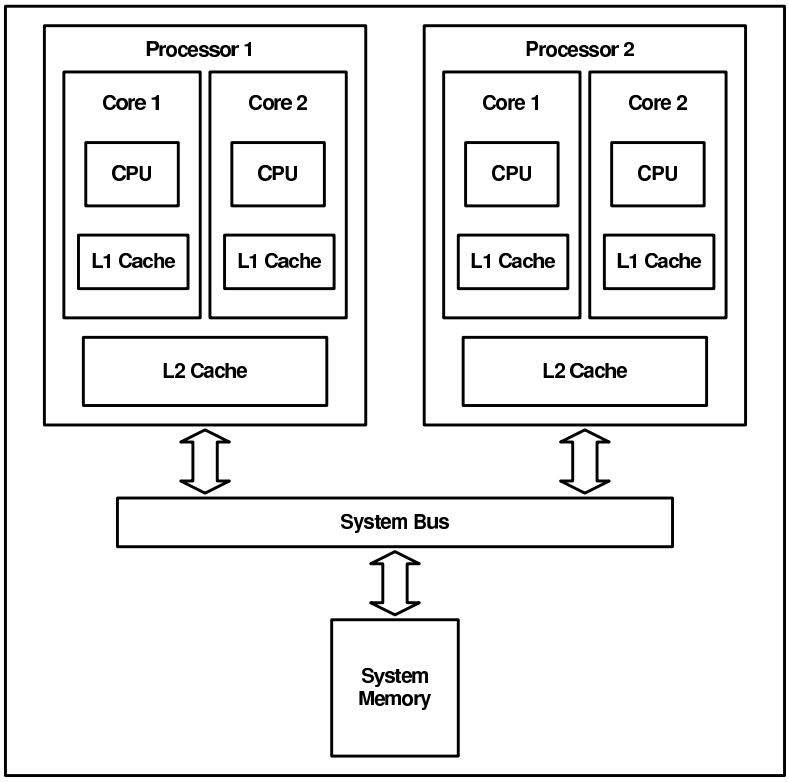



*Figure 1.* Example x86 dual-core, dual-processor system [14].

with multi-threaded applications, can keep pace with the performance demands of emerging consumer and scientific applications.

The multi-core architecture is the most popular conventional stored-program architecture (e.g., the x86 dual-core, dual-processor system in Figure 1). However, the architecture is limited by the von Neumann bottleneck – data transfer between processors and memory. Although processor speed has increased by more than two orders of magnitude over the last twenty years, application performance is still limited by memory latency rather than peak compute capability or peak bandwidth [6]. The industry response has been to move memory closer to the processor through on-board and local caches.

Similar to the conventional stored-program architecture, the CBE stores frequently-used code and data closer to the execution units. The implementation, however, is quite different [11]. Instead of the cache hierarchy implemented in traditional architectures, CBE uses direct mem-

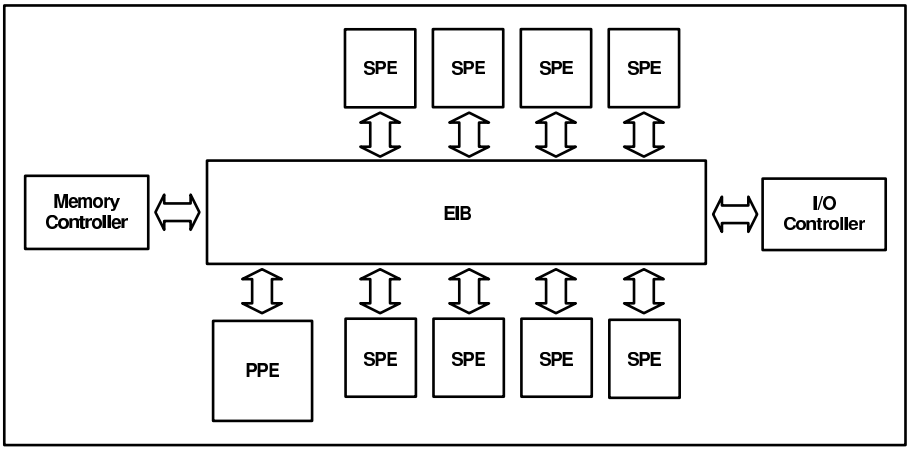



*Figure 2.* Cell Broadband Engine architecture.

ory access (DMA) commands for transfers between main storage and private local memory called "stores" [6].

Instruction fetches and load/store instructions access the private local store of each processor instead of the shared main memory. Computations and data/instruction transfers are explicitly parallelized using asynchronous DMA transfers between the local stores and main storage. This strategy is a significant departure from that used in conventional architectures that, at best, obtains minimal independent memory accesses. In the cell architecture, the DMA controller processes memory transfers asynchronously while the processor operates on previously-transferred data.

Another major difference is the use of single-instruction, multiple data (SIMD) computations. SIMD computations, first employed in large-scale supercomputers, support data-level parallelism [10]. The vector processors that implement SIMD perform mathematical operations on multiple data elements simultaneously. This differs from the majority of processor designs, which use scalar processors to process one data item at a time and rely on instruction pipelining and other techniques for speed-up. While scalar processors are primarily used for general purpose computing, SIMD-centered architectures are often used for data-intensive processing in scientific and multimedia applications [4].

## 2.2 Architectural Details

Figure 2 presents a block diagram of the CBE architecture. The CBE incorporates nine independent core processors: one PowerPC processor element (PPE) and eight synergistic processor elements (SPEs). The

PPE is a 64-bit PowerPC chip with 512 KB cache that serves as the controller for a cell [5]. The PPE runs the operating system and the top-level control thread of an application; most computing tasks are off-loaded to the SPEs [2].

Each SPE contains a processor that uses the DMA and SIMD schemes. An SPE can run individual applications or threads; depending on the PPE scheduling, it can work independently or collectively with other SPEs on computing tasks. Each SPE has a 256 KB local store, four single-precision floating point units and four integer units capable of operating at 4 GHz. According to IBM [2], given the right task, a single SPE can perform as well as a high-end desktop CPU.

The element interface bus (EIB) incorporates four 128-bit concentric rings for transferring data between the various units [5]. The EIB can support more than 100 simultaneous DMA requests between main memory and SPEs with an internal bandwidth of 96 bytes per cycle. The EIB was designed specifically for scalability – because data travels no more than the width of one SPE, additional SPEs can be incorporated without changing signal path lengths, thereby increasing only the maximum latency of the rings [15].

The memory controller interfaces the main storage to the EIB via two Rambus extreme data rate (XDR) memory channels that provide a maximum bandwidth of 25.6 GBps [2]. An I/O controller serves as the interface between external devices and the EIB. Two Rambus FlexIO external I/O channels are available that provide up to 76.8 GBps accumulated bandwidth [5]. One channel supports non-coherent links such as device interconnect; the other supports coherent transfers (or optionally an additional non-coherent link). The two channels enable seamless connections with additional cell systems.

The PS3 cell architecture is presented in Figure 3. Although it was designed for use in gaming systems, it offers massive computational power for scientific applications at low cost [2]. The architecture permits the interconnection of multiple systems for computationally-intensive problems. When additional computing power is required, additional systems are easily plugged in to expand parallel processing.

## 3. Password Cracking

We conducted a series of experiments involving password generation (using MD5 hashing) to evaluate the feasibility of using the PS3 for password cracking. The goal was to evaluate the PS3 as a viable, cost-effective option compared with the Intel and AMD systems that are currently used for password cracking.

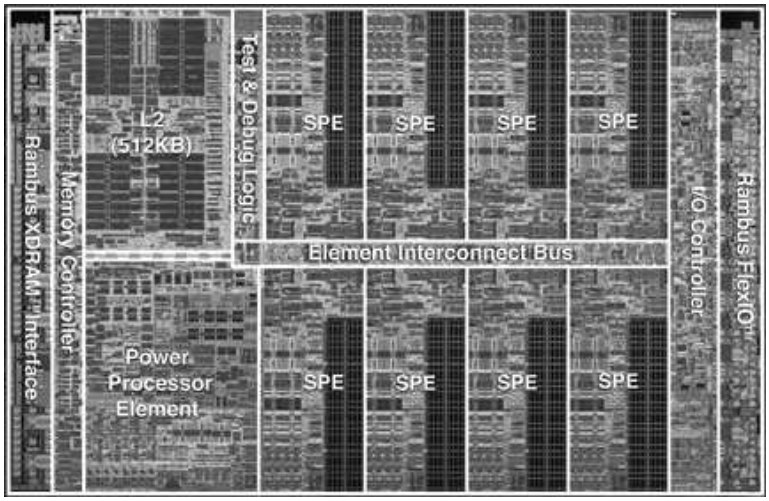



*Figure 3.* Die photograph of the PS3 cell architecture [2].

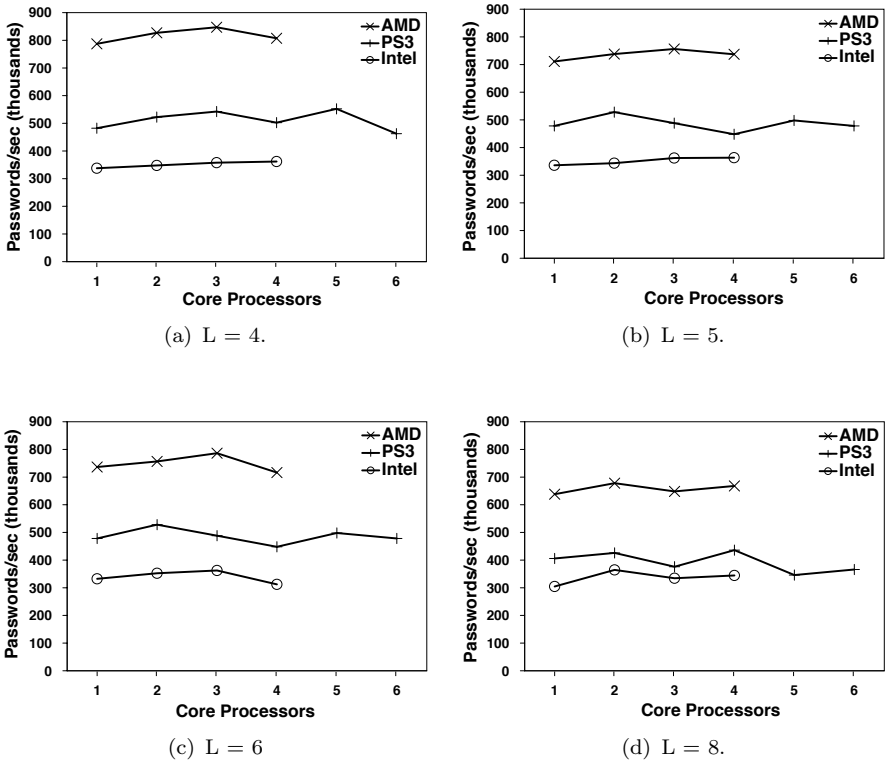
## 3.1 Experimental Setup

The experiments involved a PS3; an Intel Xeon 3.0 GHZ dual-core, dual-processor system with 2 GB RAM; and an AMD Phenom 2.5 GHz quad-core system with 4 GB RAM. All three systems ran the Fedora 9 Linux OS. MD5 code was written in C/C++ according to RFC 1321 [8] for use on the Intel and AMD systems. The algorithm was then ported to SIMD vector code for execution on the PS3. The CBE Software Development Kit [6] provided the necessary runtime libraries for compiling and executing code on the PS3.

Password generation involved two steps. The first step generated strings using a 72 character set (26 uppercase letters, 26 lowercase letters, 10 numbers and 10 special characters). The second step applied the MD5 cryptographic algorithm to create the passwords. The functionality of our MD5 code was verified by comparing generated passwords with the Linux MD5 generator. The password space was divided evenly among system processors and a strict brute-force implementation was used. Experiments were performed using password lengths of four, five, six and eight.

## 3.2 Experimental Results

Three metrics were used to evaluate password cracking performance: (i) passwords per second, (ii) computational time, and (iii) cost efficiency (i.e., "bang per buck"). The number of passwords per second was computed as the mean number of passwords generated during specific time




(a) L = 4. (b) L = 5.

(c) L = 6 (d) L = 8.

*Figure 4.* Number of passwords generated per second.

intervals (using a 95% confidence interval). The compute time was the estimated time required to generate the entire password space. The cost efficiency was calculated by dividing the number of passwords generated per second by the cost (in dollars) of the computing platform.

- **Passwords/Second:** The passwords per second metric was calculated for each independent processor. The Intel and AMD systems each use the equivalent of four processors while the PS3 uses six SPEs for computations (two SPEs are reserved for testing/overhead and are not easily accessible). Figure 4 presents the results obtained for various password lengths (L = 4, 5, 6 and 8). For the case where L = 4 (Figure 4(a)), the AMD system generated roughly 800K passwords/s for each processor; the PS3 generated approximately 500K passwords/s per processor and the Intel system 350K passwords/s per processor. As a system, AMD yielded close to 3.2M passwords/s, the PS3 generated 3.1M passwords/s, and Intel 1.4M passwords/s. For the four password lengths con-

*Table 1.* Estimated password space generation times.

| | L = 4 | L = 5 | L = 6 | L = 8 |
|---|---|---|---|---|
| **Intel** | 18.8 seconds | 1,377 seconds | 27.4 hours | 18.7 years |
| **AMD** | 8.3 seconds | 657 seconds | 12.8 hours | 8.9 years |
| **PS3** | 8.4 seconds | 649 seconds | 12.9 hours | 9.4 years |

sidered, the PS3 performed within 4% of the AMD system and significantly outperformed the Intel system.

- **Computational Time:** The computational time metric considers the worst-case scenario – every possible password from the character set has to be generated in order to guarantee that a password will be cracked. Table 1 presents the estimated times for generating all passwords of lengths 4, 5, 6 and 8. The PS3 and AMD have comparable performance; they outperform the Intel system by 47% on average.

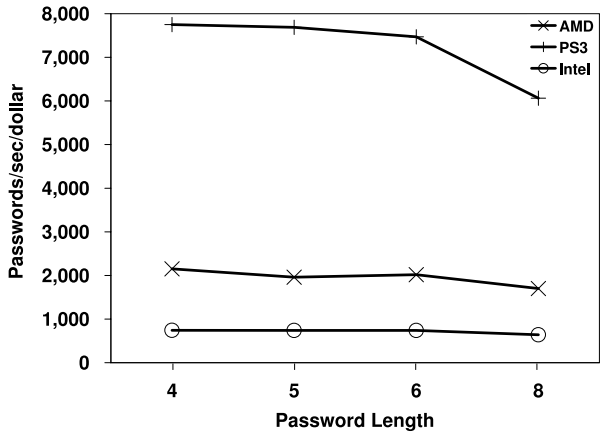



*Figure 5.* Cost efficiency results.

- **Cost Efficiency:** The cost efficiency metric compares the "bang per buck" obtained for the three password cracking systems (Figure 5). For passwords of length four, the PS3 computes 7,700 passwords/s/$ compared with 2,100 passwords/s/$ for AMD and 750 passwords/s/$ for Intel. The results are based on 2009 prices of $400 for the PS3, $1,500 for AMD and $1,900 for Intel. This metric clearly demonstrates the cost efficiency of the PS3, which,

on the average, outperforms the PS3 by a factor of 3.7 and Intel by a factor of 10.1.

### 3.3 Analysis of Results

The experimental results demonstrate that the PS3 is a viable option for password cracking. Techniques such as using separate SPEs for generating character strings and applying the MD5 algorithm, and implementing code optimization techniques (e.g., bit shifting operations) would certainly improve password cracking performance. Since the intent was not to demonstrate the full potential and computing power of the PS3, the SIMD vector code used in the experiments was not optimized. Significant computational gains can be achieved when the SIMD vector code is optimized for the PS3. Nevertheless, as the results demonstrate, the PS3 is very effective even without code optimization.

A 2007 study showed that a PS3 generates the key space for cryptographic algorithms at twenty times the speed of an Intel Core 2 Duo processor [3]. The ease of expanding computations across multiple PS3s make a multi-system framework an attractive option for large-scale password cracking efforts. Indeed, a dozen PS3 systems purchased for about $5,000 can provide the computing power of a cluster of conventional workstations costing in excess of $200,000.

## 4. Distributed Password Cracking Framework

Criminals are increasingly using encryption to hide evidence and other critical information. Most law enforcement agencies do not have the technical resources or the time to apply cryptanalytic techniques to recover vital evidence. The distributed framework presented in this section is intended to provide law enforcement agents in a large geographic region with high-performance computing resources to crack encrypted archives and applications.

The distributed password cracking framework shown in Figure 6 is designed for use by a major U.S. law enforcement agency. The design enables agents located in field offices to submit encrypted files to the Scheduler/Controller for password cracking. The Scheduler/Controller manages system overhead and assigns jobs to the PS3 Cluster based on job priority, evidence type and resource availability. The high-performance computing resources then go to work performing cryptanalysis on the submitted evidence.

Once a job is complete, details are recorded in the Repository and the submitting agent is notified about the results. The Repository serves as a vault for evidence and as a cross-referencing facility for evidence and

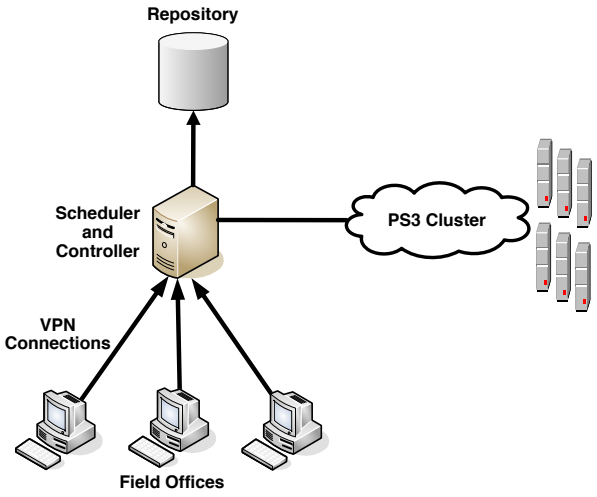



*Figure 6.* Distributed password cracking framework.

case files. Over time, common passwords and trends will emerge that can help accelerate password cracking.

The framework incorporates special security requirements that satisfy law enforcement constraints. Network channels and communications are protected using encryption and VPN tunnels. Strict identification, authentication and authorization are implemented to control access to evidence. Additionally, hashing algorithms are incorporated to ensure the integrity of the processed evidence.

Figure 7 presents a detailed view of the framework. Law enforcement agents interact with the system using the Client Module. The application programming interface (API) of the Client Module is designed to support GUI applications built on C++, Java or PHP/HTML. Using secure connections, law enforcement agents may submit new evidence, check the status of submitted evidence, and query the system for case data and statistical information.

The Command and Control Module manages system access and facilitates user queries and evidence submission. It runs on a dedicated, robust server that maintains secure access to the password cracking cluster. A Cipher Module running on each PS3 reports workload and resource availability data to the Job Scheduling Module. The Job Scheduling Module submits a job to an appropriate Cipher Module for execution. The Cipher Module manages cryptanalysis by distributing the password cracking effort between the SPEs. The key space is divided over the "allocated" or available cycles for a particular encrypted file. This could

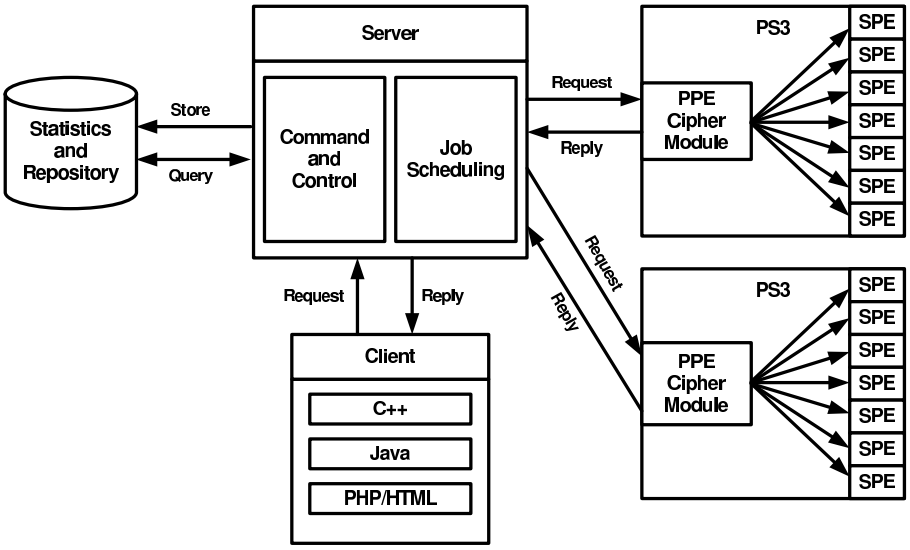



*Figure 7.* Framework components.

involve one SPE in one PS3, all the SPEs in one PS3, or multiple SPEs in multiple PS3s.

As an example, consider a Microsoft Excel 2003 spreadsheet that has been password protected. The law enforcement agent submits the file for processing. Based on the workload and the available cycles of the PS3s in the cluster, the Cipher Module allocates "chunks" of the key space for SPEs to start generating. As the keys are generated, they are encrypted using the appropriate algorithm (e.g., RC4 for Microsoft Excel 2003) and compared with the submitted evidence to locate a match. When the password is found, the results are returned to the law enforcement agent and the statistics and records are stored in the Repository.

To enable efficient password cracking, the Cipher Modules engage brute force methods augmented with dictionary word lists, previously encountered passwords and variations of commonly-used passwords. In addition, rainbow tables with pre-computed hash values and other cipher exploitation techniques may be leveraged [13]. The distributed framework alleviates the need for each field office to separately maintain expensive equipment for password cracking.

## 5. Conclusions

Law enforcement agencies can leverage the high-performance, extensibility and low cost of PS3 systems for password cracking in digital forensic investigations. The distributed PS3-based framework is designed to

enable law enforcement agents in the field to crack encrypted archives and applications in an efficient and cost-effective manner. Our future work will focus on refining the parallel PS3 architecture, optimizing SIMD vector code and continuing the implementation of the distributed password cracking framework.

## References

[1] E. Altman, P. Capek, M. Gschwind, H. Hofstee, J. Kahle, R. Nair, S. Sathaye, J. Wellman, M. Suzuoki and T. Yamazaki, U.S. Patent 6779049 – Symmetric multi-processing system with attached processing units being able to access a shared memory without being structurally configured with an address translation mechanism, U.S. Patent and Trademark Office, Alexandria, Virginia, August 17, 2004.

[2] N. Blachford, Cell architecture explained (Version 2) (www.blachford.info/computer/Cell/Cell0_v2.html), 2005.

[3] N. Breese, Crackstation – Optimized cryptography on the PlayStation 3, Security-Assessment.com, Auckland, New Zealand (www.security-assessment.com/files/presentations/crackstation-njb-bheu08-v2.pdf), 2007.

[4] M. Gschwind, The cell architecture, IBM Corporation, Armonk, New York (domino.research.ibm.com/comm/research.nsf/pages/r.arch.innovation.html), 2007.

[5] IBM Corporation, Cell Broadband Engine Architecture (Version 1.02), Armonk, New York, 2007.

[6] IBM Corporation, IBM Software Development Kit for Multicore Acceleration (Version 3), Armonk, New York, 2007.

[7] E. Nakashima, In child porn case, a digital dilemma, *The Washington Post*, January 16, 2008.

[8] R. Rivest, RFC 1321: The MD5 Message-Digest Algorithm, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts (www.ietf.org/rfc/rfc1321.txt), 1992.

[9] A. Shimpi, Understanding the cell microprocessor, AnandTech, Minden, Nevada (www.anandtech.com/cpuchipsets/showdoc.aspx?i=2379), 2005.

[10] J. Stokes, SIMD architecture, Ars Technica (arstechnica.com/articles/paedia/cpu/simd.ars), 2000.

[11] J. Stokes, Introducing the IBM/Sony/Toshiba cell processor Part I: The SIMD processing units, Ars Technica (arstechnica.com/articles/paedia/cpu/cell-1.ars), 2005.

[12] M. Suzuoki, T. Yamazaki, C. Johns, S. Asano, A. Kunimatsu and Y. Watanabe, U.S. Patent 6809734 – Resource dedication system and method for a computer architecture for broadband networks, U.S. Patent and Trademark Office, Alexandria, Virginia, October 26, 2004.

[13] C. Swenson, *Modern Cryptanalysis: Techniques for Advanced Code Breaking*, Wiley, Indianapolis, Indiana, 2008.

[14] T. Tian and C. Shih, Software techniques for shared-cache multi-core systems, Intel Corporation, Santa Clara, California (software.intel.com/en-us/articles/software-techniques-for-shared-cache-multi-core-systems), 2007.

[15] D. Wang, The cell microprocessor, Real World Technologies, Colton, California (www.realworldtech.com/page.cfm?ArticleID=RWT021005084318), 2005.