Chapter 6

USING BOOT CONTROL TO PRESERVE THE INTEGRITY OF EVIDENCE

Keisuke Fujita, Yuki Ashino, Tetsutaro Uehara and Ryoichi Sasaki

Abstract This paper describes Dig-Force2, a system that securely logs and stores evidentiary data about the operation of a personal computer. The integrity of the logged data is guaranteed by using chained hysteresis signatures and a trusted platform module (TPM) that prevents unauthorized programs or tampered programs from executing. Experiments indicate that the Dig-Force2 system is both efficient and reliable.

Keywords: Evidence integrity, hysteresis signatures, boot control

1. Introduction

Personal computers are often used as instruments of electronic crime. This makes it important to securely log and store evidentiary data pertaining to computer operations for use in legal proceedings [8].

To address this issue, we have previously developed Dig-Force [3], a system that reliably records data about personal computer use on the computer itself. Dig-Force uses chained signatures to maintain the integrity of evidentiary data. Dig-Force is effective even when it is installed on a standalone computer located outside a protected network.

One problem with Dig-Force is that it is difficult to ensure that the personal computer user cannot alter programs and data on the computer. In particular, it is necessary to detect program or data tampering and to guarantee that only authorized programs are executed. This paper describes an enhanced version of the Dig-Force system (Dig-Force2) that securely logs and stores evidentiary data pertaining to computer use. The integrity of the logged data is preserved using chained hysteresis signatures and a trusted platform module (TPM) that prevents unau-

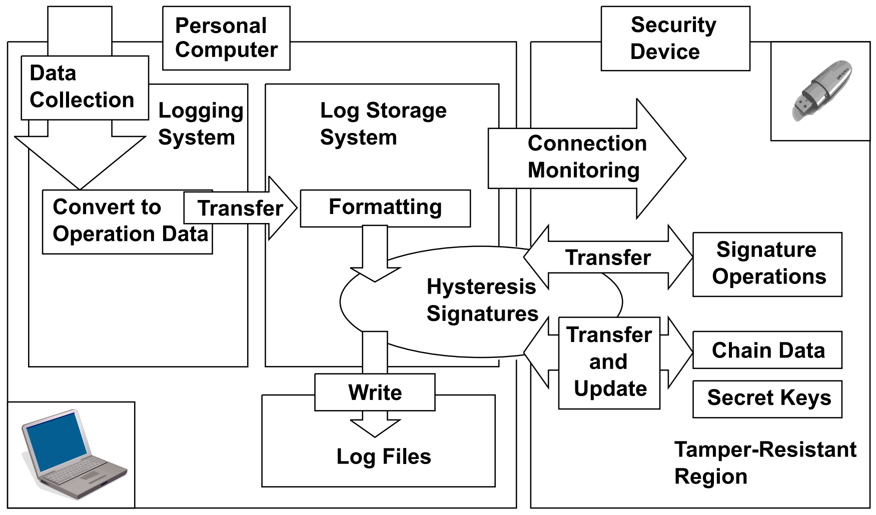

Figure 1. Dig-Force architecture.

thorized programs or tampered programs from executing. Experiments indicate that the new system is both efficient and reliable.

2. Dig-Force

This section describes the operating assumptions, architecture and processing flow of the Dig-Force system.

2.1 Operating Assumptions

The primary functions of Dig-Force are to log data about personal computer operations without any failure, and to detect tampering of the logged data even when it is done by the personal computer operator.

Dig-Force was designed to operate under three principal assumptions: (i) no programs or data on the personal computer should be modified by its operator, (ii) although the computer operator may perform unauthorized operations, neither the administrator nor the verifier ever perform unauthorized operations, and (iii) the computer operator never passes the security device (described below) to a third party.

2.2 Architecture

The Dig-Force architecture is presented in Figure 1. It consists of three principal subsystems:

- **Security Device:** The security device contains a tamper-resistant area to prevent the chained hysteresis signatures from being modified. The device also performs hysteresis signature operations.

- **Logging System:** The logging system collects operational information, which is preserved in a log storage system. The logging system also records that the security device is always installed on the personal computer. Note that the computer cannot be operated without the security device.

- **Log Storage System:** The log storage system communicates with the security device and stores the hysteresis signatures with the logged data. The logged data is intended to serve as evidence in legal proceedings. However, since the data is easily coped, erased or modified, chained hysteresis signatures are used instead of independent digital signatures to ensure the security and integrity of the logged data.

A hysteresis signature is a digital signature with a chained structure [6], i.e., each signature is dependent on the preceding signatures. A successful attack involving the alteration of data would require all the hysteresis signatures preceding and following the data to be adjusted. Thus, a hysteresis signature is more secure and reliable than a traditional digital signature.

The security device enables Dig-Force to defend against "restoration attacks," which are effective against hysteresis signatures [3]. Such an attack occurs when an intruder deletes the suffixes of log file data and performs a series of operations to update the log file.

2.3 Processing Flow

Dig-Force's processing flow has three phases: configuration, operation and verification. Three entities are involved: the system administrator who makes the initial settings, the personal computer operator, and the verifier who checks the logs stored on the computer.

- **Configuration Phase:** During this phase, the system administrator uses the security device to create public/private key pairs and stores the private keys in the tamper-resistant area of the security device. The administrator also determines the initial values of the chain data and stores them in the tamper-resistant area. Next, the administrator sends the public keys and the initial values of the chain data to the verifier. Finally, the administrator delivers the personal computer installed with the logging system, log storage system and security device to the operator.

- **Operation Phase:** The operator uses the personal computer received from the system administrator. The logging system confirms that the security device is installed, after which it collects operational data and passes it to the log storage system.

  The log storage system accumulates the logged data, communicates with the security device, applies the hysteresis signatures to the logged data, writes the chain values, signatures and logged data to the log file, and stores the chained hash values in the security device.

  Upon completing a session, the personal computer operator saves his documents and the log file on storage media, and submits the storage media and security device to the verifier.

- **Verification Phase:** The verifier applies the hash function to the final chain data contained in the submitted log file and computes the hash values. The verifier then compares these hash values with those contained in the security device. Next, the verifier checks the signatures using the initial values from the configuration phase and the chain values, signatures and logged data stored in the log file.

3. Implementation Issues

It is difficult to guarantee that users cannot modify programs or data on a personal computer. A malicious user with sufficient expertise could alter the Dig-Force program itself so that it does not detect tampering. To address this threat, Dig-Force should be tamperproof and computer operations should be monitored to ensure that only the "correct" versions of authorized programs execute. This can be implemented using a white list containing the digital signatures of approved programs, which are provided by a trusted third party.

Another important requirement is to implement boot control functionality that prevents unauthorized programs from executing. To accomplish this, we use features provided by Microsoft Windows XP, which is used as the development and operational environment. In particular, we leverage the multiple hierarchies that Windows XP provides from the hardware layer all the way up to the application layer (Figure 2).

In general, there are three ways to implement the reliable monitoring of programs on a personal computer. These involve using: (i) the operating system, (ii) a device driver located within the operating system kernel (Figure 2), and (iii) APIHook, a service program that hooks the Windows API, changes the processing and monitors application program start-up.

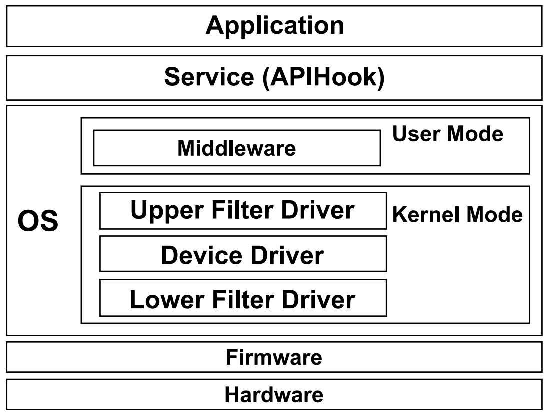

Figure 2. Windows XP hierarchy.

Although the operating system is the most desirable option for monitoring unauthorized programs, Windows XP does not offer the required functionality. Therefore, we considered the device driver and APIHook solutions. Ultimately, we selected APIHook because it was easier to implement.

APIHook is a service program that executes in the application layer; therefore, there is always the risk that it can be tampered with. It is difficult to tamper with or delete the APIHook program on a computer because APIHook is automatically set to execute first. However, if the hard drive is transferred to another computer, the APIHook program on the drive can be modified or deleted. To address this issue, we have developed an enhanced version of Dig-Force, called Dig-Force2, which engages a trusted platform module (TPM) as an additional safeguard.

TPMs are integrated circuit chips with security hardware that protect certain areas of the chips from being tampered with. A TPM mounted on the motherboard of a personal computer can function as a coprocessor accessible from the CPU via the low pin count bus. The Trusted Computing Group (TCG) [10] defines several functions for a TPM. These include: (i) creating, storing and conducting encryption/decryption and signature operations with RSA keys, (ii) performing hashing operations, (iii) generating random numbers, (iv) maintaining information on platform state, and (v) providing adequate non-volatile and volatile memory for storing data.

A TPM that is mounted in a personal computer is machine specific. Therefore, if a TPM in a computer is removed and replaced with another TPM, an authentication error results and the computer will not start up.

Thus, the TPM can be used to uniquely identify a particular computer. The encryption key in a TPM is also machine specific. Therefore, the hard drive data, which is enciphered using the TPM's encryption key, cannot be deciphered by another machine. Thus, even if the hard drive is moved to another personal computer, it is not possible to delete or otherwise tamper with the APIHook program.

There are two additional reasons for using a TPM. First, apart from the TPM, there is no need to incorporate a special device in the personal computer. Second, an auxiliary device (e.g., USB device) can be removed by mistake, which terminates data collection and storage.

4. Dig-Force2

We have designed the Dig-Force2 system to address the limitations of Dig-Force. Dig-Force2 employs a TPM as its security device rather than eToken [2], which is used by Dig-Force. This section describes the operating assumptions, architecture and processing flow of Dig-Force2.

4.1 Operating Assumptions

Dig-Force2 is designed to operate under two primary assumptions: (i) since the monitoring program is a service that operates under the administrator's authority, it cannot be halted by a computer user whose authority is lower than that of the administrator, and (ii) the BIOS, operating system and monitoring program software are reliable; since the monitoring program starts up right after the operating system, it is difficult for an unauthorized individual to alter the monitoring program, which features APIHook functions.

4.2 Architecture

Dig-Force2 has five main components (Figure 3):

- **Logging System:** The logging system collects operational information, which is maintained in a log storage system. The logging system also records that the auxiliary device is always installed on the personal computer.

- **Log Storage System:** The log storage system adds timestamps and formats the operational data received from the logging system. Then, it interacts with the TPM to apply hysteresis signatures to the formatted data and writes the data to a log file.

- **Auxiliary Device:** The auxiliary device must be inserted into the personal computer in order for the computer to operate. The

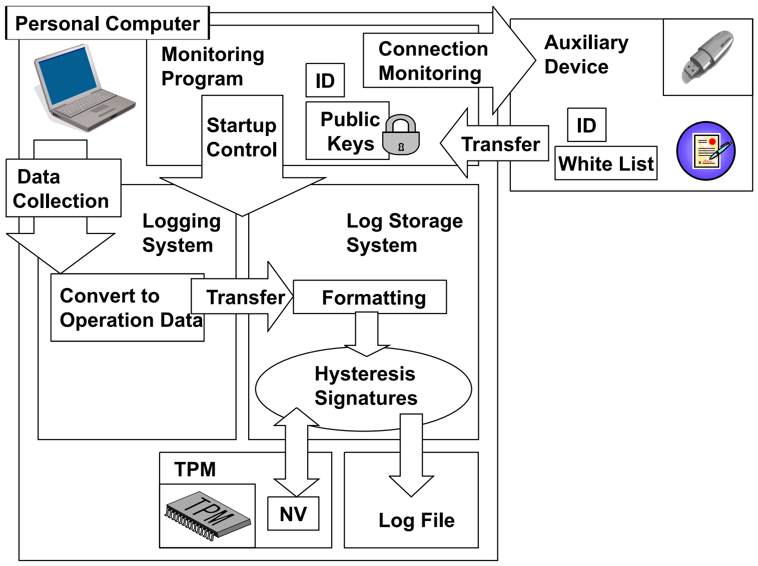

Figure 3. Dig-Force2 architecture.

auxiliary device stores the hash values ("white list") of authorized programs and IDs that identify users along with the digital signatures of the hash values and IDs (signed by the administrator).

- **Trusted Platform Module:** The TPM performs the hysteresis signature operations. It has a tamper-resistant area that protects the chain data that forms the signature keys and signature history.

- **Monitoring Program:** The monitoring program is started as a service at the uppermost authority of the operating system, which prevents the computer operator from terminating the program. The monitoring program checks that the auxiliary device is mounted on the computer; the computer cannot be operated without this device. The program then computes the hash values of the `.exe` files of the logging and log storage systems and compares them with the values in the white list; the computer is permitted to start only if the values match. Note that before the white list is used, the digital signature provided by the administrator or a trusted third party is verified. The monitoring program also reads the ID from the auxiliary device that identifies the computer operator and compares it with its pre-set value; the computer can be operated only if the IDs match.

After the computer has booted, the logging system and log storage system processes are monitored for unauthorized termination. Also, whenever the user attempts to start a new program, the monitoring program computes the hash value of the corresponding `.exe` file and compares it with the corresponding value in the white list; this ensures that only authorized programs are executed.

4.3 Processing Flow

Dig-Force2's processing flow has three phases: configuration, operation and verification.

- **Configuration Phase:** During this phase, the administrator configures the personal computer before passing it along with the auxiliary device to the operator. The following steps are involved in the configuration phase:

  - The administrator creates a storage root key (SRK) in the TPM; this root key secures the other TPM keys.
  - The administrator creates a secret key (S) in the TPM that is used for the hysteresis signatures and a public key (P) used for signature verification. These keys are encrypted using SRK and stored on the computer's hard drive.
  - The administrator specifies an arbitrary initial value (R1) for the chain data and stores it in the TPM's non-volatile memory.
  - The administrator sends the initial values that become the chain data to the verifier.
  - The administrator stores in the auxiliary device the white list and the IDs used by the monitoring program to identify individuals.
  - The administrator sets up the public key used for verifying the white list and the IDs that identify individuals in the monitoring program.
  - Finally, the administrator installs the monitoring system and the logging and log storage systems on the personal computer.

- **Operation Phase:** During this phase, the auxiliary device is mounted on the personal computer, and the monitoring program and the logging and log storage systems are started. Figure 4 illustrates the processing flow of the monitoring program. The following steps are involved:

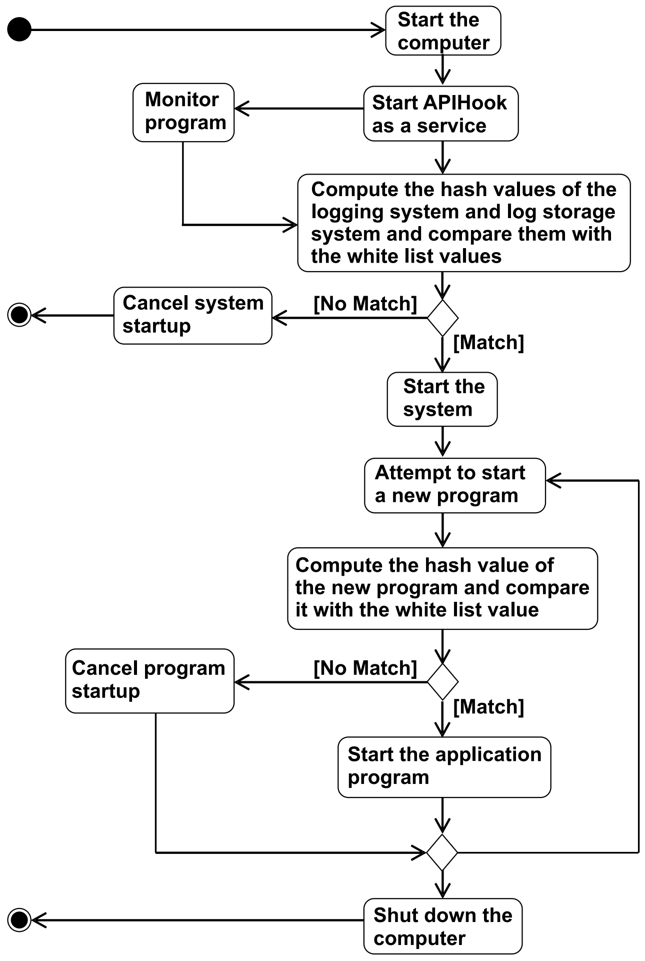
Start the computer
Start APIHook as a service
Monitor program
Compute the hash values of the logging system and log storage system and compare them with the white list values
[No Match]
Cancel system startup
[Match]
Start the system
Attempt to start a new program
Compute the hash value of the new program and compare it with the white list value
[No Match]
Cancel program startup
[Match]
Start the application program
Shut down the computer

Figure 4. Monitoring flow diagram.

– The operator mounts the auxiliary device on the computer.

– The monitoring program is started as a service program after the operating system has started.

– The monitoring program checks that the auxiliary device is mounted; if the device is not mounted, the monitoring program locks the computer.

– The monitoring program reads the IDs that identify individuals and their digital signatures from the auxiliary device. The monitoring program verifies the digital signatures; the computer works only if this verification is successful.

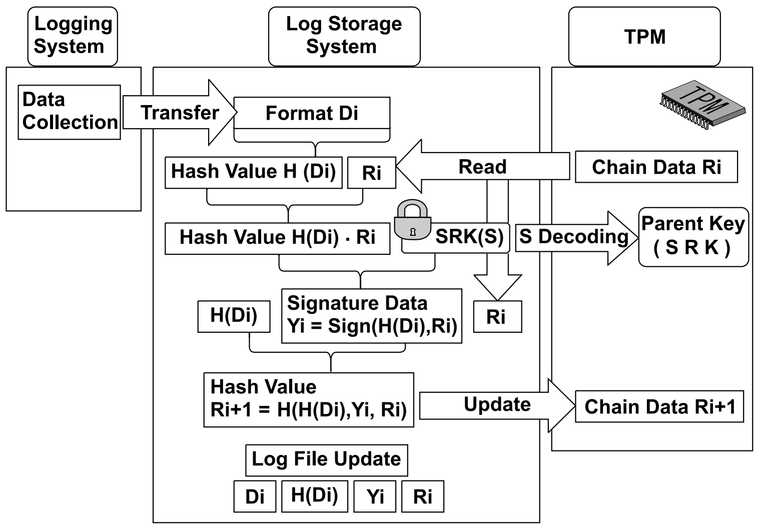

Figure 5. Hysteresis signatures using the TPM.

– The monitoring program reads the white list and the digital signatures from the auxiliary device; the signatures are verified using the public key.

– The monitoring program computes the hash values of the logging and log storage programs and compares them with those in the white list. If the hash values match, the monitoring program starts the programs and applies the hysteresis signatures to the logged data (Figure 5). The processing flow is the same as that of Dig-Force except that the TPM key (SRK(S)) is used for signature operations in the TPM and the chain data is stored in the TPM's non-volatile memory.

– The monitoring program checks the other programs (`.exe` files) to ensure that unauthorized programs do not start. Also, whenever the operator attempts to start a program, the monitoring program hooks the API and computes the hash value of the program; the program is permitted to execute only if this hash value matches its white list value.

- **Verification Phase:** The verifier receives the personal computer with the TPM and the auxiliary device from the operator. The verification of digital signatures in the TPM is similar to that for Dig-Force, except that the chain data stored in the TPM's non-volatile memory is used along with the TPM keys. This confirms

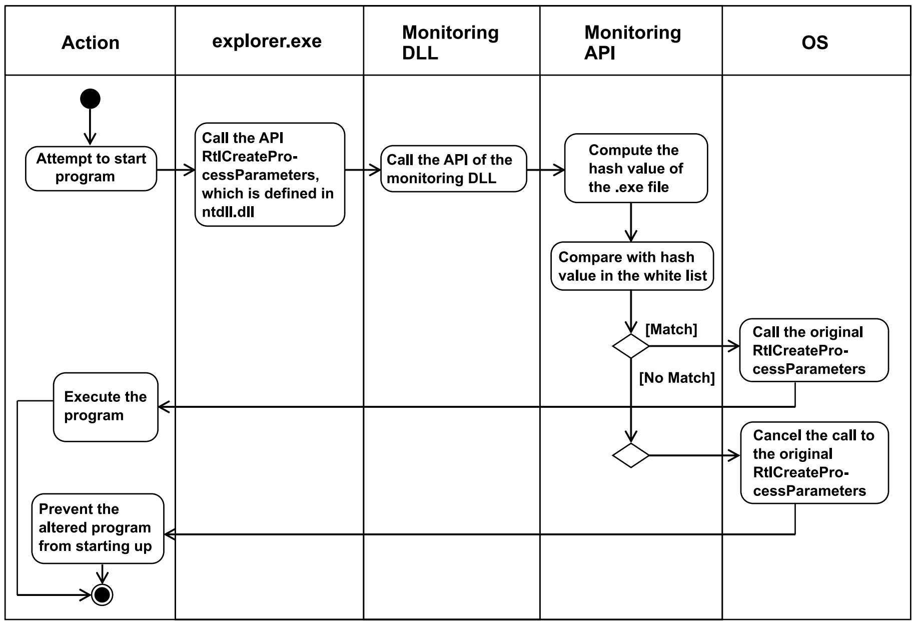
Action
explorer.exe
Monitoring DLL
Monitoring API
OS
Attempt to start program
Call the API RtlCreatePro-cessParameters, which is defined in ntdll.dll
Call the API of the monitoring DLL
Compute the hash value of the .exe file
Compare with hash value in the white list
[Match]
[No Match]
Call the original RtlCreatePro-cessParameters
Execute the program
Cancel the call to the original RtlCreatePro-cessParameters
Prevent the altered program from starting up

Figure 6. Boot control flow diagram.

that the monitoring program is operating in a reliable manner, ensuring that only authorized programs are executed.

5. Functional Experiments

Functional tests of the monitoring program and TPM were conducted to verify the effectiveness and practicality of Dig-Force2.

5.1 Monitoring Program

An experiment was conducted to test the program start-up control function provided by APIHook. APIMonitor [4] was used to identify the APIs invoked during program start up (we discovered that more than ten APIs are called when a program is started). We hooked some of these APIs and changed their processing flow. One of these APIs is `RtlCreateProcessParameters`, which is defined in `ntdll.dll`. Figure 6 illustrates how this particular API is hooked to control booting.

We wrote a prototype program that implemented the processing flow in Figure 6 based on information provided in [1]. The program was used to experiment with boot control. It was able to prevent unauthorized programs from starting up. However, it was unable to exert boot control on programs launched via a command prompt (`cmd.exe`). This is because hooking an API is not available with a command prompt, which is a DOS program. The problem was addressed by disabling program start-up using command prompts, which caused all DOS programs to be unavailable. Our future research will attempt to develop a boot control technique for DOS programs.

5.2 Trusted Platform Module

Experiments were conducted to evaluate the performance of the TPM for signature operations and its use of non-volatile memory. The development effort used the C++ programming language under Microsoft Visual Studio.NET 2003 and Infineon TPM Integration SDK; Windows XP Professional was used as the evaluation environment.

Experiments were conducted to measure the time taken for hysteresis signature processing and verification. The average times were 0.0764 seconds and 0.0295 seconds, respectively; these were deemed to be acceptable in operational environments.

Sufficient non-volatile memory is required in the TPM to store the chain data for the hysteresis signatures. Our experiments confirmed that 20 bytes of non-volatile memory were available in the TPM.

5.3 Possible Attacks

This section describes five attacks that can impact Dig-Force2 along with the corresponding countermeasures.

- **Attack 1:** This attack launches a malicious program that modifies the logged data and/or signatures. The attack is defeated by ensuring that the monitoring program checks every program and only permits authorized programs to execute.

- **Attack 2:** This attack tampers with or deletes the monitoring program after transferring the hard drive on which it resides to another computer. The hard drive is then returned to the original computer. Thus, the monitoring program is unable to prevent unauthorized programs from executing. The attack is defeated by enciphering the monitoring program using the TPM's encryption function and public key so that it cannot be decrypted on another computer.

- **Attack 3:** This attack halts the logging program and/or log storage program, preventing Dig-Force2 from collecting evidentiary data. The attack is defeated by having the monitoring program check the programs' execution status and automatically restart the programs if they are terminated.

- **Attack 4:** This attack modifies the white list so that the monitoring program permits an unauthorized program to execute. The attack is defeated by using digital signatures that employ the administrator's private key; the monitoring program uses the corresponding public key to verify the signatures and detect alterations.

- **Attack 5:** This attack alters the chain data stored in the non-volatile memory of the TPM. It is defeated by ensuring that the monitoring program prevents unauthorized programs (that access the non-volatile memory) from executing.

6. Conclusions

Dig-Force2 is an efficient and reliable system for collecting data about computer operations for use in legal proceedings. The integrity of the evidentiary data is ensured by using chained hysteresis signatures and a TPM that prevents unauthorized programs from executing.

Our future research will focus on enhancing the Dig-Force2 system. One issue is that the white list contains only the names of authorized programs and their hash values. However, it is also necessary to consider DLLs and plug-ins because tampering with these components can cause unauthorized programs to execute. We will attempt to augment the white list by incorporating the hash values of approved DLLs and plug-ins [9]. Another important issue is to harden the monitoring program against attacks. A promising approach is to use BitLocker in Windows Vista [5] to encrypt the hard drive that contains the monitoring program.

Acknowledgements

The authors wish to thank IBM Japan and Infineon Technologies, Japan for their assistance with TCG technology.

References

[1] K. Aiko, Kenji's Homepage (ruffnex.oc.to/kenji).

[2] Aladdin Knowledge Systems, Petach Tikva, Israel (www.aladdin .com).

[3] Y. Ashino and R. Sasaki, Proposal of digital forensic system using security device and hysteresis signature, *Proceedings of the Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 3–7, 2007.

[4] R. Batra, API Monitor (www.rohitab.com/apimonitor).

[5] Microsoft Corporation, BitLocker Drive Encryption, Redmond, Washington (www.microsoft.com/windows/products/windowsvista/features/details/bitlocker.mspx).

[6] K. Miyazaki, S. Susaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, Digital document sanitizing problem, *Institute of Electronics, Information and Communication Engineers Technical Reports*, vol. 103(195), pp. 61–67, 2003.

[7] J. Richter, *Advanced Windows*, Microsoft Press, Redmond, Washington, 1997.

[8] R. Sasaki, Y. Ashino and T. Masubuchi, A trial for systematization of digital forensics and proposal on the required technologies, *Japanese Society of Security Management Magazine*, April 2006.

[9] SignaCert, Independent IT Controls, Portland, Oregon (japan.signacert.com).

[10] Trusted Computing Group, Beaverton, Oregon (www.trustedcomputinggroup.org).